2021

Mobile Application for the School of Computer Science (EII)

EII. University of Oviedo

Software Engineering Degree.
Campus de los Catalanes (Oviedo).

AUTHOR:

Vaz Sánchez, Adrián

TUTORS:

Vinuesa Martínez, Luis Antonio
Álvarez García, Fernando

Mobile Application for the School of Computer Science

AUTHOR:	Vaz Sánchez, Adrián
TUTORS:	Vinuesa Martínez, Luis Antonio Álvarez García, Fernando
DATE:	01/11/2021
VERSION:	1.0.
UNIVERSITY:	School of Computer Science (EII). University of Oviedo

Aplicación móvil de la Escuela de Ingeniería Informática

AUTOR:	Vaz Sánchez, Adrián
TUTORES:	Vinuesa Martínez, Luis Antonio Álvarez García, Fernando
FECHA:	01/11/2021
VERSIÓN:	1.0.
UNIVERSIDAD:	Escuela de Ingeniería Informática (EII). Universidad de Oviedo

Abstract

EIIProject, is a **software development** project solicited by a **real client**, the **School of Computer Science** (EII, University of Oviedo), that consists of two clearly differentiated parts:

- **EIIAPP**: A School Management Application, used by the students at the School of Computer Science, that aims to centralize all information of the latter: schedules, events, information of interest, among others.

- **EIISERVER**: A server deployed on School premises, used by the staff of the latter, that serves the information to EIIAPP through a REST API and communicates with the students through Push Notifications.

Keywords: EII, University of Oviedo, School Management Application, Software Development Project, Push Notifications.

Resumen

EIIProject, es un **Proyecto de desarrollo software** solicitado por un **cliente real**, la **Escuela de Ingeniería Informática** (EII, Universidad de Oviedo), que consta de dos partes claramente diferenciadas:

- **EIIAPP**: Una Aplicación de Gestión Escolar, utilizada por los estudiantes de la Escuela de Ingeniería Informática, cuyo objetivo es centralizar toda la información de estos últimos: horarios, eventos, información de interés, entre otras funcionalidades.

- **EIISERVER**: Un servidor desplegado en las instalaciones de la Escuela, utilizado por el personal de esta última, que sirve información a EIIAPP por medio de una API REST y se comunica con los estudiantes mediante Notificaciones Push.

Palabras clave: EII, Universidad de Oviedo, Aplicación de Gestión Escolar, Proyecto de Desarrollo Software, Notificaciones Push.

This document has been created based on the template elaborated by **JOSÉ MANUEL REDONDO LÓPEZ** [1], [2]

Declaration of Originality

I, Adrián Vaz Sánchez, hereby certify that I am the sole author of this thesis and that no part of this last one has been published or submitted for publication, and that all references and sources of information used have been properly cited.

Acknowledgements

I would like to take advantage of this section to collect all who have made possible this work:

First, I would like to give thanks to my family for all their unconditional support and love through my entire life, particularly my mother, who has poured herself into my project success body and soul.

I would also like to acknowledge all the help received from my friends, including all the people that I have met in the School and that I already consider my friends, for bearing me and, honestly, for letting me use their devices to annoy them with floods of notifications, especially Covadonga Vega, the first beta-tester of the School.

I also want to thank María del Carmen Suárez Torrente, for having introduced me the School back in 2017 in one of the UniTour talks given by the School of Computer Science, causing me to enroll in this degree.

It goes without saying that I would also like to thank my tutors Luis Antonio Vinuesa Martínez and Fernando Álvarez García, for responding to each and every email I sent them throughout this project and guiding/advising me with all their knowledge, i.e., for helping me finish this degree that started with themselves giving me an introductory talk of the School and Software Engineering Degree on the Welcome Day of the School.

Table Of Contents

Table Of Figures

Table Of Tables

1 PLANNING OF THE INFORMATION SYSTEM

1.1 DEFINITION AND ORGANIZATION OF THE PSI

1.1.1 SPECIFICATION OF THE GOAL AND SCOPE

The goal and scope of both systems, EIISERVER and EIIAPP is detailed in Determination of the scope of the system.

1.1.2 ORGANIZATION OF THE PSI

The organization of the PSI is detailed in Table 1: Organization of the PSI. In addition, the **stakeholders** of the project, that also play a key role in this last, are detailed in Identification of the stakeholders.

Responsible	Professional profile	Role
DEVELOPMENT TEAM		
Vaz Sánchez, Adrián	Software Engineering Student	Creator, designer, analyst and engineering of the whole project and this document
PROJECT REPRESENTATIVES		
Vinuesa Martínez, Luis Antonio	Deputy Principal of Enterprise of the School	Client representative and Project Acceptor
Álvarez García, Fernando	Principal of the School	Client representative and Project Acceptor
BETA TESTERS		
Students of the Software for Mobile Devices Course.	Beta Testers	Testers of the system

Table 1: Organization of the PSI

1.2 STUDY OF RELEVANT INFORMATION

To **collect information** about the client that hires the project: School of Computer Science and due to the pandemic, that we are living by the time this is being documented, we use **the email as communication channel**. Note that the client has also provide a **document with a proposal that indicates what the School needs**.

1.2.1 ANALYSIS AND SELECTION OF ANTECEDENTS

The collected **antecedents** are summarized below:

- The School needs an **Application** developed for **Android** Operating Systems.
- The **Application** shall let a student of the Software Engineering Degree and the Master in Web Engineering **log in with its UO**.
- There shall be a **server**, behind the application, **deployed on the facilities of the School of Computer Science**.

- The server shall let log in an authorized member of the School: Secretary of Direction of the School, Janitors of the School, or other authorized Staff Members.
- Authorized users will have one or more **roles**:
  - **Administrators** can update the information stored on the server.
  - **Notifiers** can send notifications to students that are authenticated on the application.
- An authenticated student on the application shall be able to:
  - View its schedule.
  - Access to the stored information offline.
  - View its exams.
  - Receive notifications from the School.
  - Export its events to Google Calendar.
  - View the Links of Interest Page managed by the server.
  - Notify breakdowns of a computer on the School premises.
  - Update the stored information manually or periodically.
- An authenticated user with the role of an Administrator shall be able to update the information stored on the server by accessing the **official sources of the School** [3]–[5].
- The server shall authenticate the identity of a user with the **LDAP of the University of Oviedo**.

2 DEFINITION OF THE TECHNOLOGICAL ARCHITECTURE

In this section, we will discuss **the needs** behind the selection of the technological infrastructure in Identification of the needs of the technological infrastructure, **the selected infrastructure** in Selection of the technological infrastructure, and the reasons behind **the selected architecture** for EIIAPP and EIISERVER explaining the communications and relationships from an architectural point of view in Selection of the architecture.

2.1 IDENTIFICATION OF THE NEEDS OF THE TECHNOLOGICAL INFRASTRUCTURE

The needs of the technological architecture for both systems, EIIAPP and EIISERVER, are listed down below (more information that completes this section is detailed in System Definition):

- **EIIAPP** will be built for **Android Operating Systems**: mobile devices and tablets. This is a constraint given by the client: School of Computer Science since the purpose here is to release the system only on **Google Play Store**.
- **EIIAPP** shall communicate with **EIISERVER** to obtain the information of each student authenticated on EIIAPP.
    - **EIISERVER** shall be deployed on the infrastructures of the client.
        - On a machine with **Windows or Linux** Operating Systems.
    - **EIISERVER** shall be deployed on an accessible IP outside from the infrastructures of the client: School of Computer Science.
- **EIISERVER** shall send notifications to one or more instances of **EIIAPP** (mobile devices).
    - It is necessary for the notifications, to **persist across reboots**, **shutdowns** and **force stops** of the application.
    - It is necessary for the notifications to be **received** on the device **when EIIAPP** is being used (the application **is opened**) **and when** it is not (the application **is closed**).
    - It is necessary for the notifications to **persist after they are received** (the user must be able to access all received notifications).
- **EIISERVER** shall **authenticate** the identity of a user that is using the system. This is also a **constraint** imposed by the client: the School of Computer Science belongs to the net of universities of the University of Oviedo and the authentication method for all its member is **LDAP**.
- **EIISERVER** shall **obtain the information** related to the Software Engineering Degree and the Master in Web Engineering academic degrees **from the official sources available on the web** [3]–[5].
- **EIIAPP** shall let an authenticated user export its information (events stored on the system) to **Google Calendar** (imposed by the client: School of Computer Science).
    - **EIIAPP** shall **prevent** the export of events that were **already exported** in the past.
    - **EIIAPP** shall **delete** the events that were exported in the past and does **no longer exist** on the system (they were cancelled, removed, etc.).
    - **EIIAPP** shall let the user **decide in which Calendar** (of Google Calendar) he wishes to export the events.
- **EIIAPP** shall communicate with **EIISERVER** to update periodically the information stored on the first system.
    - It is necessary for the update process, to **persist across reboots**, **shutdowns** and **force stops** of **EIIAPP**.

- It is necessary for the update process to be **executed when EIIAPP** is being used (the application **is opened**) **and when** it is not (the application **is closed**).

2.2 SELECTION OF THE TECHNOLOGICAL INFRASTRUCTURE

In this section the selected technological infrastructure will be cited. To know the reasons behind every selection, see Selection of the architecture and Feasibility study of the system.

EIISERVER will be deployed on a **Windows Machine** on the infrastructures of the School of Computer Science. The reasons behind using Windows over Linux are simple and listed below:

- The **Development Team** (Adrián Vaz Sánchez) **has a Windows Machine** and is familiar with its environment.
- The **Development Team has no Linux Machine**.
- To prevent issues from deploying **EIISERVER** on a Linux Machine after having been developed on a Windows one.

EIISERVER will be built using **NodeJs** and **Typescript** technologies. It will follow a **Clean Architecture** along with an **API REST**, using **Express** for that purpose.

For heavy-computational tasks, **Worker Threads** will be used to lower the workload of **EIISERVER**: the process used for sending notifications and the process that updates the information stored on the database reading from **the official sources of the School available on the web** [3]–[5]. EIISERVER will communicate with **PostgreSQL** to persist information.

EIIAPP will be developed using **Flutter** and **Dart** technologies. To update the information stored on the system, it will communicate with **EIISERVER** through the exposed **API REST** of **EIISERVER**.

EIIAPP will be developed following a **Model-View-ViewModel Architecture** along with a **Provider Architecture**.

The **update process**, among other processes of **EIIAPP**, will run on **Dart Isolates** (Dart approach for "threads") to lower the workload of the main thread of the application. To schedule updates, **Android Alarm Manager** will be used. **EIIAPP** will communicate with **Hive** to persist information.

For the notifications we will use the **Firebase Messaging API** and for the events exportation from **EIIAPP**, **Google Calendar** along with **Google OAuth**.

2.3 SELECTION OF THE ARCHITECTURE

In this section we will detail the **architecture** used for both systems: **EIIAPP** and **EIISERVER**.

2.3.1 PROVIDER AND MODEL – VIEW – VIEW MODEL ARCHITECTURE FOR EIIAPP

As we have stated on the title, for **EIIAPP** we make use of **Provider Architecture** along with a **Model-View-View Model Architecture**.

2.3.1.1 MODEL – VIEW – VIEW MODEL ARCHITECTURE

To understand why these mentioned architectures fit well within our EIIAPP system, let's have a look into some Flutter **basic concepts** and start from here:

Flutter's building blocks are called "**widgets**". A **widget** is a piece of code that builds part of a view. For instance, a Scrolling View could be a widget (see Figure 1: Selection of the Architecture. EIIAPP. Building Blocks).

And **building blocks**, like when we are working on the construction of a building (to make a metaphor with the real world), represent the "bricks" that build our architecture.

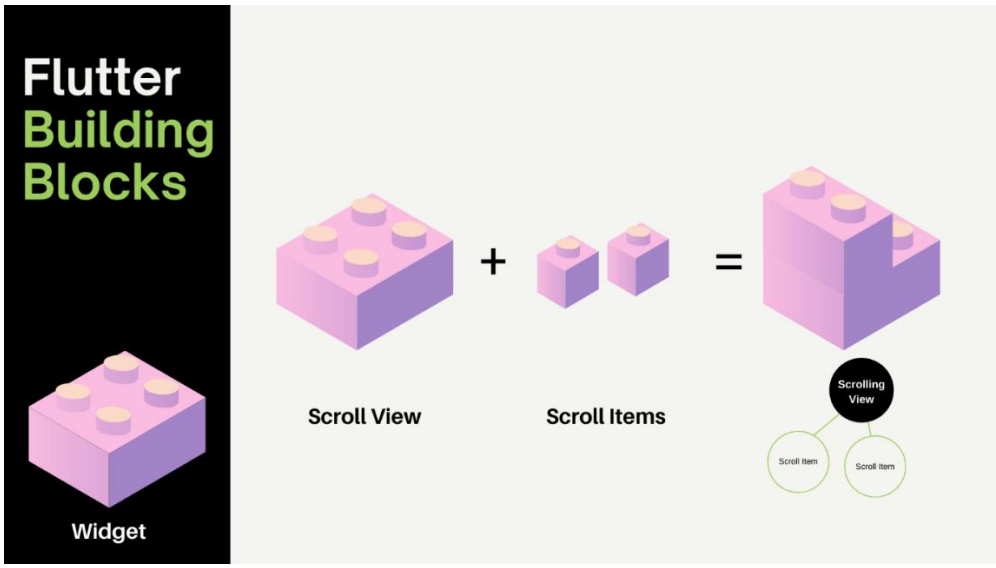


Figure 1: Selection of the Architecture. EIIAPP. Building Blocks

The composition of widgets that builds a View, is called the **widget tree**. For instance, on Figure 1: Selection of the Architecture. EIIAPP. Building Blocks, the widget tree is represented on the right: the root of the widget tree is the **Scrolling View**, and the nodes of the tree are each **Scroll Item**.

There are two types of widgets: **stateless widgets** and **stateful widgets**:

- **Stateless widgets** are **immutable** classes, in other words, elements of an interface whose content never changes.
- On the other hand, **stateful widgets** are pieces of code that **can change across time**, that changes are notified to the dependent widgets by calling **setState**.

A simple real example where a state change appears is the following: Whenever a user presses the right arrow in the month switcher displayed on the top of Calendar View (see Figure 2: Selection of the Architecture. EIIAPP. Stateful Widget), the selected month should be incremented in one unit. So "Month Switcher" could be modelled as a stateful widget with three methods: "getSelectedMonth", "incrementSelectedMonth" and "decrementSelectedMonth".

All the logic about getting the selected month, incrementing, or decrementing it and notifying about changes is **wrapped inside our widget** (Month Switcher). Nevertheless,

- what would happen if **another widget needed to access the selected month**?
- Could we pass the "selected month" **across the widget tree**?
- **Who is responsible** for managing all the **logic** if more than one widget changes the value of "selected month"?

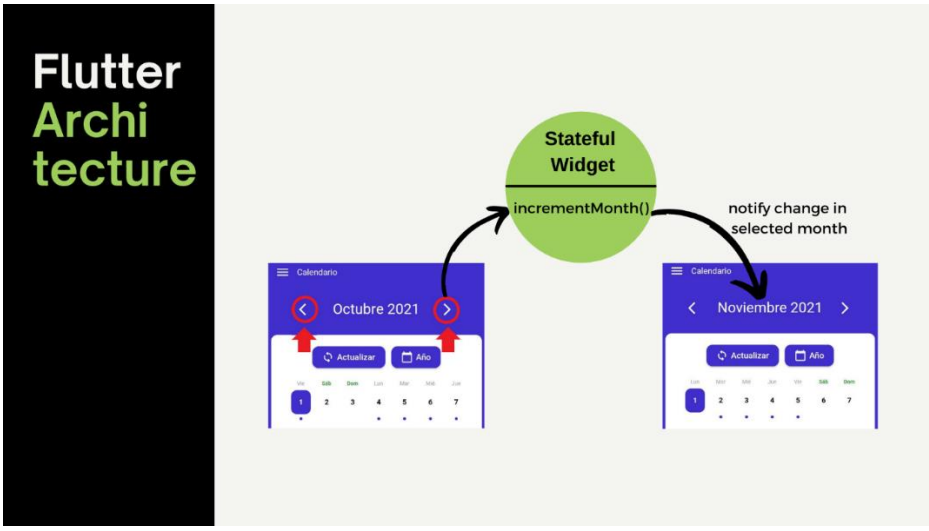


Figure 2: Selection of the Architecture. EIIAPP. Stateful Widget

To answer all these questions, we hereby introduce a concept well-known in other languages as React, among others: **Lifting State Up**. If data must be shared among multiples widgets, a widget placed **at the root of widget** tree can pass this information down to the previously mentioned widgets.

And here is where **View Models** get into action. **Each View has a View Model that is responsible for providing all information** to the user interface and notifying whenever some of that information changes.

All the logic about changing and notifying changes is now **wrapped inside the View Model** and the totality of the widgets placed across the widget tree, the view, will have access to the View Model and all its exposed methods. In that case, changes across the widget tree will not imply such a big problem, all widgets inside the Calendar View can get the value from "selected month", increment and decrement it.

With the use of the view models, we are **adapting** our views to future changes in the requirements, avoiding or **minimizing the redesign**, i.e., making our **interfaces flexible**.

2.3.1.2 PROVIDER ARCHITECTURE

However, the following constraint is going to break the architecture explained before and demolish our flexibility:

When a notification arrives, user should be informed about it

For instance, we could **show a badge next to the Notification Card** that appears in the Home View indicating the number of unread notifications.

We can tackle the problem just like we have discussed before: **Home View Model provides** the number of unread notifications by exposing a public method ("getUnreadNotifications()") **and notifies listeners** when this value changes (as shown on Figure 3: Selection of the Architecture. EIIAPP. Notification badge example).

If we **wanted to change the place where the number of unread notifications is displayed** across the widget tree of the Home View, we would have to move the call "model.getUnreadNotifications()" to the desired widget and that is all we would need.

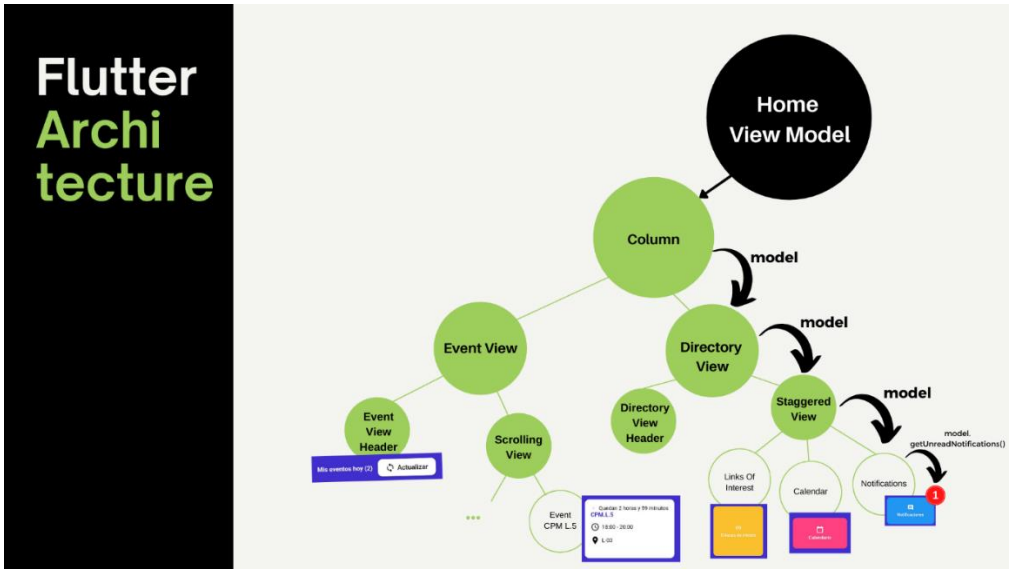


Figure 3: Selection of the Architecture. EIIAPP. Notification badge example

Imagine that now **EIIAPP shall display the number of unread notifications in more places apart from Home View**: in the Notification Tile of the **Navigation Drawer** and in the **Notification View** (see [Figure 4: Selection of the Architecture. EIIAPP. Architecture problem]).

Taking into account that the method "getUnreadNotifications()" **can only be accessed within Home View's widget tree**, how can this be achieved?

One approach could be to make Notification View Model also provide the number of unread notifications, exposing a method ("getUnreadNotifications()") and notify listeners when this value changes, i.e., do the same as Home View Model. And we would have to do the same for the Navigation Drawer. In other words, **we would need to duplicate code**.

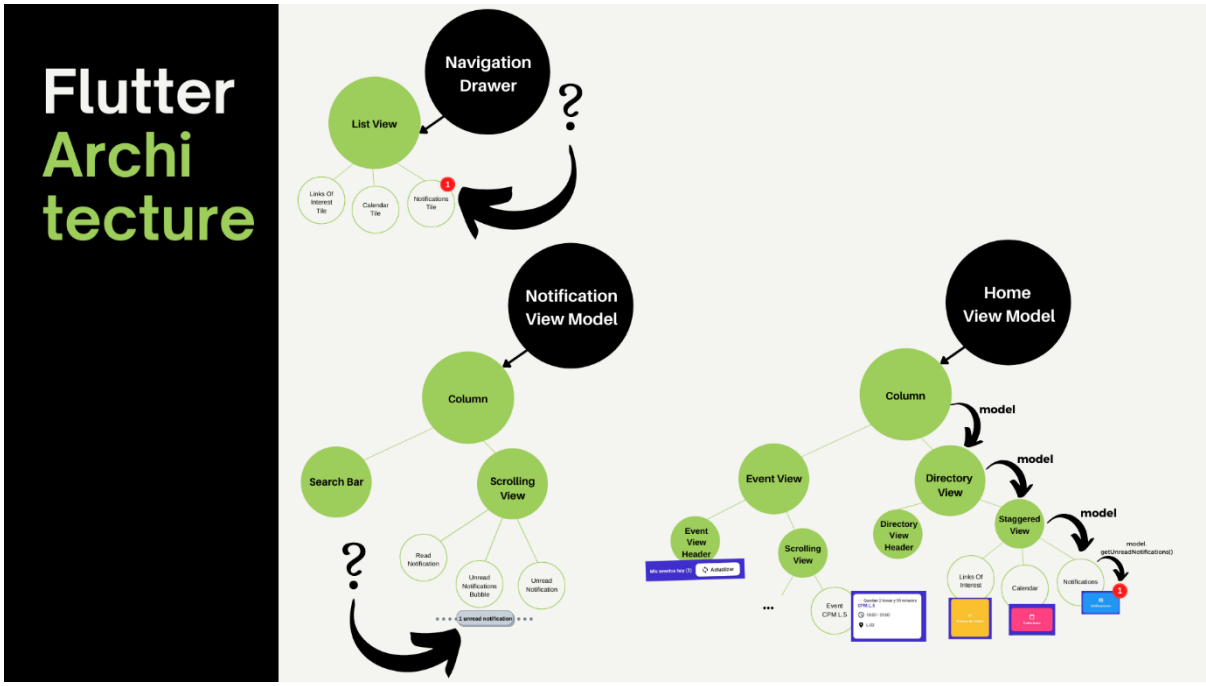


Figure 4: Selection of the Architecture. EIIAPP. Architecture problem

Another approach is to apply the same strategy as before (the "**Lifting State Up**" principle): we place a widget that is responsible for communicating state changes and exposing values as the root of our tree, the **Provider Manager** (see Figure 5: Selection of the Architecture. EIIAPP. Provider Architecture). The Views are now also part of our tree, so all exposed values can be accessed in all screens: Notification View, Home View, Navigation Drawer, and any other desired.

This **flexibility** makes us be able to **locate the notification badge wherever is required** with no need to redesign any part of our architecture.

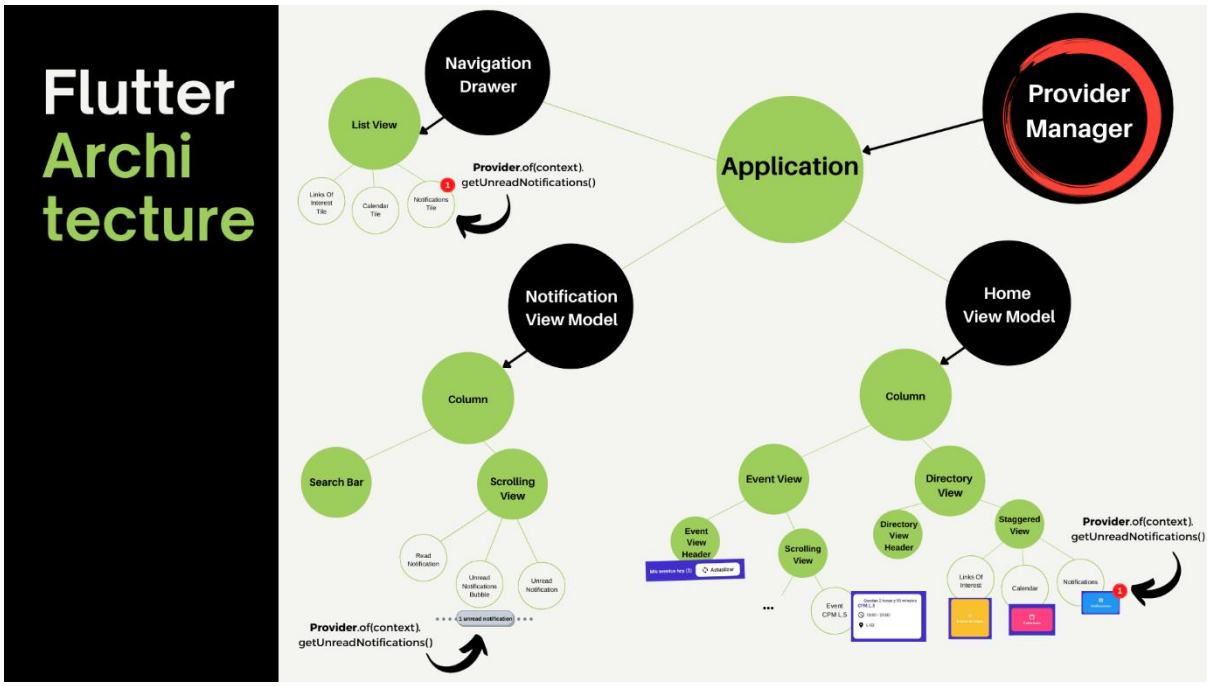


Figure 5: Selection of the Architecture. EIIAPP. Provider Architecture

2.3.2 CLEAN ARCHITECTURE FOR EIISERVER

On the other hand, for **EIISERVER**, we have used a **Clean Architecture** [6]. A sketch of the architecture is shown on Figure 6: Selection of the Architecture. EIISERVER. Clean Architecture.

With this architecture, we present to **separate the infrastructure from the business logic** and place the **inputs and outputs of EIISERVER on the edges**. It was also inspired from **Netflix's Hexagonal Architecture Approach** [7].

The architecture has evolved from a protype in a piece of paper as shown on Figure 7: Selection of the Architecture. EIISERVER. Evolution. The main concepts behind this architecture are the following ones:

- The **dependencies shall point inwards**, i.e., to the most important/stable elements.
  - **Datasources and Transport** depend on Repositories, Interactors and Entities.
  - **Entities** do not depend on any other elements than other Entities.
  - **Interactors** do only depend on Entities and Repositories.
- There are **three big layers** in the architecture:
  - Transport Layer
  - Core:
    - Entities

        - Interactors
        - Repositories
    - Datasource Layer

Transport Layer manages all the **inputs** of the system:

- the information provided by the Software Engineering Degree and the Master in Web Engineering endpoints [3]–[5]
- the API REST exposed to the EIIAPP
- the API for users of the EIISERVER

Datasource Layer, the outputs of the system:

- LDAP Authentication
- Database
- Push Notification Service

And the **Core Layer**, all the business logic of the EIISERVER.

In other words, the **Transport Layer** act as **ports** where information can be sent and the Datasource Layer as **adapters** that adapt the outer systems to **EIISERVER**.

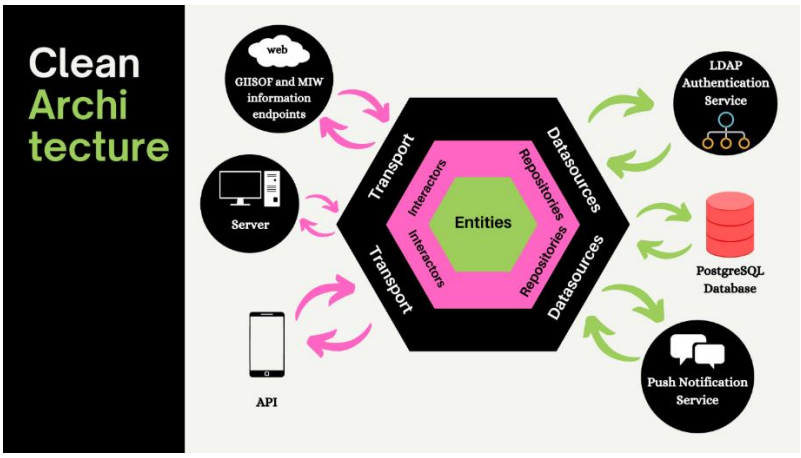


Figure 6: Selection of the Architecture. EIISERVER. Clean Architecture

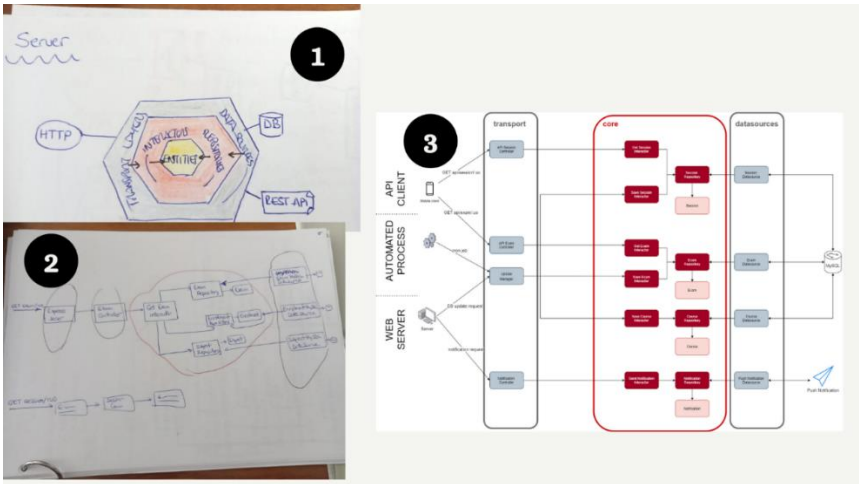


Figure 7: Selection of the Architecture. EIISERVER. Evolution

3 FEASIBILITY STUDY OF THE SYSTEM

3.1 STUDY AND EVALUATION OF THE SOLUTION ALTERNATIVES

In this section different alternatives relative to EIIAPP and EIISERVER will be discussed. The selected solution is detailed on Definition of the technological architecture.

3.1.1 MODEL – VIEW – CONTROLLER ARCHITECTURE FOR EIISERVER

The first idea for the architecture of the EIISERVER system was to apply a **Model-View-Controller**. This idea was discarded, even when it has remarkable advantages such as fast development, due to the following reasons:

- Controllers tend to contain business logic.
- This makes testing difficult: dependency injection, mocks, etc.
- And breaks design principles such as the Separation of Concerns for big projects.
- Difficult, in some cases, to decouple code (classes strongly dependent) and the Service Layer usually provides a wide range of methods (Separation of Concerns).

3.1.2 NODEJS WITH JAVASCRIPT

JavaScript is usually used along with NodeJS, the technology of EIISERVER. However, JavaScript was finally replaced with Typescript due to the following disadvantages:

- JavaScript is not strongly typed. This often leads to greater number of bugs during development, increasing the development time.
- Code management more difficult. With Typescript, many development environments provide accurate suggestions and warnings about errors related to types.

Figure 8: Study and evaluation of the solution alternatives. TypeScript evolution shows the number of downloads of Typescript in the last year, pointing out a non-stopping increase in the popularity of this language. Data is taken from **NPMTrends** [8].

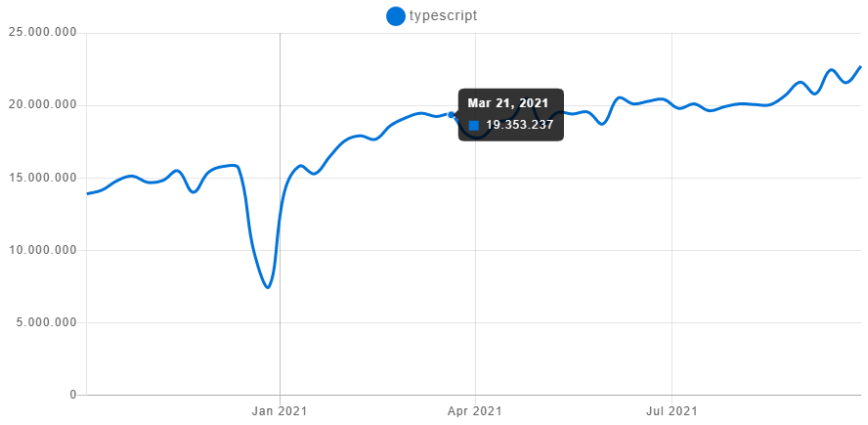


Figure 8: Study and evaluation of the solution alternatives. TypeScript evolution

3.1.3 USING JAVA OVER FLUTTER AND DART

The use of Java for the development of EIIAPP was taken in consideration due to the next reasons:

- **Familiarity**: Java is a programming language used in many courses of the Software Engineering Degree of the School of Computer Science.
- **Support**: Java has a big community of developers.
- All these reasons lead to **fast development times**.

But it was discarded in favour of Flutter and Dart because of:

- They are incipient technologies backed by **Google**.
- Flutter facilitates the creation of **flexible, modern and expressive interfaces**.
- **Performance**: aiming to provide **sixty frames per second** on most devices.

3.1.4 USING A MODEL – VIEW – CONTROLLER ARCHITECTURE FOR EIIAPP

Before deciding to Use a Model – View – ViewModel (MVVM) a Model – View – Controller (MVC) was taken in consideration but discarded due to the following reasons:

- With, MVVM, business logic is decoupled from the Views.
- MVVM fits the "event-driven" nature of Flutter (updating the View Models and notifying listeners, i.e., Views).
- With MVVM, View Models can be reused by multiple Views.

4 PLANNING AND MANAGEMENT OF THIS PROJECT

4.1 PLANNING OF THE PROJECT

4.1.1 IDENTIFICATION OF THE STAKEHOLDERS

We, hereby, present the identified stakeholders as well as sources of information to the reader:

4.1.1.1 STAKEHOLDERS

- School of Computer Science (Main Client)
  - Principal of the School: Fernando Álvarez García.
  - Deputy Principal of Enterprise of the School: Luis Antonio Vinuesa Martínez.
  - Deputy Principal of Infrastructures and Human Resources of the School: Jordán Pascual Espada.
  - Head of Studies and Academic Secretary of the School: Juan Ramón Pérez Pérez.
  - Coordinator of the Master in Web Engineering: Cristina Pelayo García-Bustelo.
  - Marketing and Social Networks Department: José Manuel Redondo López (Community Manager).
  - Scholars of the School.
- Authentication Methods:
  - LDAP Service (University of Oviedo).
  - Google OAuth.
- Google Calendar API.
- Firebase Messaging API.
- Final Users of the System:
  - Students of the School.
  - Secretary of Direction of the School, Janitors of the School, and other Staff Members of the School (Administrators and Notifiers).
- University of Oviedo.
- Developer Team:
  - Software Analyst, Project Manager, Developer, Tester, Software Architect, Software Engineer, Requirements Engineer, Test Engineer: Adrián Vaz Sánchez.
  - Beta-testers: Students of the Software for Mobile Devices Course.
- Hosting Provider.
- Domain Name Provider.
- Security Certificate Provider: School of Computer Science.
- Other School Management Systems.
- Other School Management Applications.
- School of Computer Science Investors.

4.1.1.2 SOURCES OF INFORMATION

- Internal Information about the Master in Web Engineering of the School.
- Information about the Software Engineering Degree.
- Information about the LDAP Authentication of the University of Oviedo.
- Google OAuth Guidelines and Conditions.
- Web Corporate Images of the School of Computer Science.
- Operation Manuals of the School.
- Métrica V3.

4.1.2 OBS AND PBS

4.1.2.1 OBS

The Organization Breakdown Structure (**OBS**) of the project is shown in Figure 9: OBS. What this structure represents is **the roles that** the student on charge of this project, **Adrián Vaz Sánchez**, **will play** throughout the previously mentioned project.

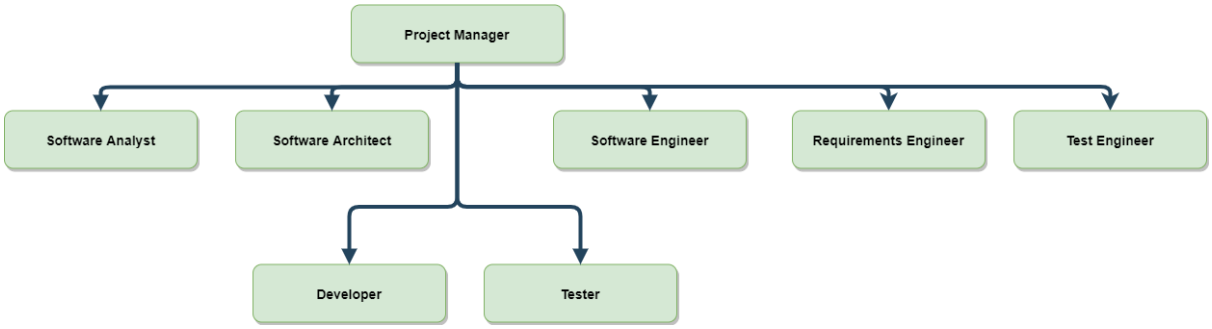


Figure 9: OBS

4.1.2.2 PBS

The Product Breakdown Structure (**PBS**) can be seen in Figure 144: From the architecture to WBS. EIIAPP. PBS , Figure 139: From the architecture to WBS. EIISERVER. PBS and Figure 146: From the architecture to WBS. PBS (Hardware). Traceability between the WBS and the PBS is shown on Table 2: Traceability between the WBS and the PBS.

WBS	PBS
EIIAPP	**EIIAPP**
Study of EIIAPP	Use Cases of EIIAPP
Development of Locators	**Locators**
Selection of Use Cases of Locators	List of Use Cases of Locators
Development of Locator	Locator
Development of Core	**Core**
Selection of Use Cases of Core	List of Use Cases of Core
Development of Models	**Models**
Selection of Use Cases of Models	List of Use Cases of Models
Development of Model	Model
Development of Services	**Services**
Selection of Use Cases of Services	List of Use Cases of Services
Development of Service	Service
Development of ViewModels	**ViewModels**
Selection of Use Cases of ViewModels	List of Use Cases of ViewModels
Development of ViewModel	ViewModel
Development of Datasources	**Datasources**

Selection of Use Cases of Datasources	List of Use Cases of Datasources
Development of Datasource	Datasource
Development of UI	**UI**
Selection of Use Cases of UI	List of Use Cases of UI
Development of Views	**Views**
Selection of Use Cases of Views	List of Use Cases of Views
Development of View	View
Development of Widgets	**Widgets**
Selection of Use Cases of Widgets	List of Use Cases of Widgets
Development of Widget	Widget
EIISERVER	**EIISERVER**
Study of EIISERVER	Use Cases of EIISERVER
Development of Logger	**Logger**
Selection of Use Cases of Logger	List of Use Cases of Logger
Development of LoggerManager	LoggerManager
Development of Core	**Core**
Selection of Use Cases of Core	List of Use Cases of Core
Development of Entities	**Entities**
Selection of Use Cases of Entities	List of Use Cases of Entities
Development of Entity	Entity
Development of Interactors	**Interactors**
Selection of Use Cases of Interactors	List of Use Cases of Interactors
Development of Interactor	Interactor
Development of Datasources	**Datasources**
Selection of Use Cases of Datasources	List of Use Cases of Datasources
Development of LDAPAuthenticationDatasource	**LDAPAuthenticationDatasource**
Selection of Use Cases of LDAPAuthenticationDatasource	List of Use Cases of LDAPAuthenticationDatasource
Development of PUSHNotificationDatasource	**PUSHNotificationDatasource**
Selection of Use Cases of PUSHNotificationDatasource	List of Use Cases of PUSHNotificationDatasource
Development of EntityDatasource	**EntityDatasource**
Selection of Use Cases of EntityDatasource	List of Use Cases of EntityDatasource
Development of Transport	**Transport**
Selection of Use Cases of Transport	List of Use Cases of Transport
Development of Loaders	**Loaders**
Selection of Use Cases of Loaders	List of Use Cases of Loaders
Development of Loader	Loader
Development of Middlewares	**Middlewares**
Selection of Use Cases of Middlewares	List of Use Cases of Middlewares
Development of Middleware	Middleware
Development of Readers	**Readers**
Selection of Use Cases of Readers	List of Use Cases of Readers
Development of GIISOFReader	**GIISOFReader**
Selection of Use Cases of GIISOFReader	List of Use Cases of GIISOFReader
Development of MIWReader	**MIWReader**
Selection of Use Cases of MIWReader	List of Use Cases of MIWReader
Development of Workers	**Workers**
Selection of Use Cases of Workers	List of Use Cases of Workers
Development of Worker	Worker
Development of Routes	**Routes**
Selection of Use Cases of Routes	List of Use Cases of Routes
Development of Route	Route
Development of Factories	**Factories**
Selection of Use Cases of Factories	List of Use Cases of Factories

Development of ErrorHandlerFactory	**ErrorHandlerFactory**
Selection of Use Cases of ErrorHandlerFactory	List of Use Cases of ErrorHandlerFactory
Development of LoaderFactory	**LoaderFactory**
Selection of Use Cases of LoaderFactory	List of Use Cases of LoaderFactory
Development of ReaderFactory	**ReaderFactory**
Selection of Use Cases of ReaderFactory	List of Use Cases of ReaderFactory
Development of WorkerFactory	**WorkerFactory**
Selection of Use Cases of WorkerFactory	List of Use Cases of WorkerFactory
Development of LoggerFactory	**LoggerFactory**
Selection of Use Cases of LoggerFactory	List of Use Cases of LoggerFactory
Development of InteractorFactory	**InteractorFactory**
Selection of Use Cases of InteractorFactory	List of Use Cases of InteractorFactory
Development of RepositoryFactory	**RepositoryFactory**
Selection of Use Cases of RepositoryFactory	List of Use Cases of RepositoryFactory
Project	**Project**
Study of the needs of the client	Specificaions
Installation of the server	**Server**
Server Configuration	Server Configured
Deployment of EIISERVER on the Server	EIISERVER Deployed on the Server

Table 2: Traceability between the WBS and the PBS

4.1.3 INITIAL PLANNING. WBS

To take a closer look at the elaboration of the Work Breakdown Structure, please see From the architecture to WBS. To know how the effort and size estimation process was done, please see Estimation of size and effort.

On the other hand, the activities shown in this section, follow **Métrica V3** [9], as stated in other sections of this document.

Note that for the construction and development of the system, percentages shown in Table 3: Initial Planning. WBS. Activity Percentages were applied.

Activity	Percentage
Development	**35%**
Project Management	**12%**
Requirements Analysis	**18%**
Design	**15%**
Tests	**10%**
User Manuals	**10%**

Table 3: Initial Planning. WBS. Activity Percentages

In that case, the following numbers are applied to our project, throughout its different activities:

Activity	Time
EIISERVER Development	**9 days**
EIISERVER Project Management	**3 days**
EIISERVER Requirements Analysis	**5 days**
EIISERVER Design	**4 days**
EIISERVER Tests	**3 days**
EIISERVER User Manuals	**3 days**
EIIAPP Development	**6 days**
EIIAPP Project Management	**2 days**
EIIAPP Requirements Analysis	**3 days**
EIIAPP Design	**2 days**

EIIAPP Tests	**2 days**
EIIAPP User Manuals	**2 days**

Table 4: Initial Planning. WBS. EII Project Activity Percentages

The initial planning of EIIProject was **intended to finish the 12th of May 2021**, starting the 2nd of February, and is structured in the next subsections.

Note that beta-testing activities are not included since they are performed and managed by the own client (its students of the Software for Mobile Devices Course).

4.1.3.1 PLANNING OF THE INFORMATION SYSTEM

EII Project	**1**	**70 days**	**Tue 02/02/21**	**Wed 12/05/21**		
Planning of the information system	**1.1**	**4 days**	**Tue 02/02/21**	**Fri 05/02/21**		
Definition and Organization of the PSI	1.1.1	2 days	Tue 02/02/21	Wed 03/02/21		
Study of relevant information	1.1.2	2 days	Thu 04/02/21	Fri 05/02/21	3	

Figure 10: Initial Planning. WBS. Planning of the information system

4.1.3.2 DEFINITION OF THE TECHNOLOGICAL ARCHITECTURE

Definition of the technological architecture	**1.2**	**5 days**	**Mon 08/02/21**	**Fri 12/02/21**	**2**
Identification of the needs of the technological infrastructure	1.2.1	1 day	Mon 08/02/21	Mon 08/02/21	
Selection of the architecture	1.2.2	4 days	Tue 09/02/21	Fri 12/02/21	6
Feasibility Study of the system	1.3	2 days	Mon 15/02/21	Tue 16/02/21	5

Figure 11: Initial Planning. WBS. Definition of the technological architecture

4.1.3.3 PLANNING AND MANAGEMENT OF THE PROJECT

Planning and management of the project	**1.4**	**8 days**	**Mon 15/02/21**	**Wed 24/02/21**	**5**
Identification of the stakeholders	1.4.1	1 day	Mon 15/02/21	Mon 15/02/21	
OBS	1.4.2	1 day	Tue 16/02/21	Tue 16/02/21	10
PBS	1.4.3	1 day	Tue 16/02/21	Tue 16/02/21	10
WBS	1.4.4	3 days	Tue 16/02/21	Thu 18/02/21	10
Risks	1.4.5	3 days	Tue 16/02/21	Thu 18/02/21	10
Budget	1.4.6	4 days	Fri 19/02/21	Wed 24/02/21	11;12;13;14
Project Closure	1.5	3 days	Mon 10/05/21	Wed 12/05/21	156

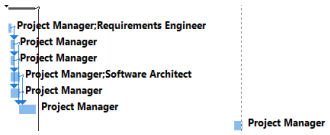


Figure 12: Initial Planning. WBS. Planning and Management of the Project

4.1.3.4 SYSTEM STUDY

System Study	**1.6**	**50 days**	**Thu 25/02/21**	**Fri 07/05/21**	
System Analysis	**1.6.1**	**14 days**	**Thu 25/02/21**	**Tue 16/03/21**	**9**
System Design	**1.6.2**	**11 days**	**Wed 17/03/21**	**Wed 31/03/21**	**18**
System Construction	**1.6.3**	**17 days**	**Mon 05/04/21**	**Tue 27/04/21**	**61**
System Introduction and Acceptance	**1.6.4**	**2 days**	**Thu 06/05/21**	**Fri 07/05/21**	**157**

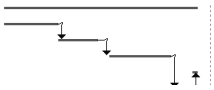


Figure 13: Initial Planning. WBS. System Study

4.1.3.4.1 SYSTEM ANALYSIS

System Analysis	**1.6.1**	**14 days**	**Thu 25/02/21**	**Tue 16/03/21**	**9**
System Definition	1.6.1.1	3 days	Thu 25/02/21	Mon 01/03/21	
Requirements establishment	1.6.1.2	5 days	Tue 02/03/21	Mon 08/03/21	19
EIISERVER module Identification	**1.6.1.3**	**5 days**	**Fri 05/03/21**	**Thu 11/03/21**	**57;10;58;59;6(**
EIIAPP module identification	**1.6.1.4**	**3 days**	**Fri 12/03/21**	**Tue 16/03/21**	**21**
Class Analysis	1.6.1.5	3 days	Thu 25/02/21	Mon 01/03/21	
Data modelling	1.6.1.6	6 days	Thu 25/02/21	Thu 04/03/21	
Defining User Interfaces	1.6.1.7	3 days	Thu 25/02/21	Mon 01/03/21	
Testing Plan Specification	1.6.1.8	5 days	Thu 25/02/21	Wed 03/03/21	

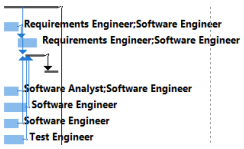


Figure 14: Initial Planning. WBS. System Analysis

4.1.3.4.1.1 EIISERVER MODULE IDENTIFICATION

EIISERVER module Identification	**1.6.1.3**	**5 days**	**Fri 05/03/21**	**Thu 11/03/21**	**57;10;58;59;60**
Logger module analysis	**1.6.1.3.1**	**2 days**	**Fri 05/03/21**	**Mon 08/03/21**	
LoggerManager module analysis	1.6.1.3.1.1	2 days	Fri 05/03/21	Mon 08/03/21	
Datasources module analysis	**1.6.1.3.2**	**5 days**	**Fri 05/03/21**	**Thu 11/03/21**	
LDAPAuthenticationDatasource module analysis	1.6.1.3.2.1	5 days	Fri 05/03/21	Thu 11/03/21	
EntityDatasource module analysis	1.6.1.3.2.2	5 days	Fri 05/03/21	Thu 11/03/21	
PUSHNotificationDatasource module analysis	1.6.1.3.2.3	5 days	Fri 05/03/21	Thu 11/03/21	
Core module analysis	**1.6.1.3.3**	**3 days**	**Fri 05/03/21**	**Tue 09/03/21**	
Entities module analysis	1.6.1.3.3.1	2 days	Fri 05/03/21	Mon 08/03/21	
Interactors module analysis	1.6.1.3.3.2	3 days	Fri 05/03/21	Tue 09/03/21	
Transport module analysis	**1.6.1.3.4**	**5 days**	**Fri 05/03/21**	**Thu 11/03/21**	
Loaders module analysis	1.6.1.3.4.1	2 days	Fri 05/03/21	Mon 08/03/21	
Middlewares module analysis	1.6.1.3.4.2	2 days	Fri 05/03/21	Mon 08/03/21	
Readers module analysis	**1.6.1.3.4.3**	**5 days**	**Fri 05/03/21**	**Thu 11/03/21**	
GIISOFReader module analysis	1.6.1.3.4.3.1	5 days	Fri 05/03/21	Thu 11/03/21	
MIWReader module analysis	1.6.1.3.4.3.2	5 days	Fri 05/03/21	Thu 11/03/21	
Workers module analysis	1.6.1.3.4.4	5 days	Fri 05/03/21	Thu 11/03/21	
Routes module analysis	1.6.1.3.4.5	5 days	Fri 05/03/21	Thu 11/03/21	
Factories module analysis	**1.6.1.3.5**	**2 days**	**Fri 05/03/21**	**Mon 08/03/21**	
ErrorHandlerFactory module analysis	1.6.1.3.5.1	2 days	Fri 05/03/21	Mon 08/03/21	
LoggerFactory module analysis	1.6.1.3.5.2	2 days	Fri 05/03/21	Mon 08/03/21	
LoaderFactory module analysis	1.6.1.3.5.3	2 days	Fri 05/03/21	Mon 08/03/21	
ReaderFactory module analysis	1.6.1.3.5.4	2 days	Fri 05/03/21	Mon 08/03/21	
InteractorFactory module analysis	1.6.1.3.5.5	2 days	Fri 05/03/21	Mon 08/03/21	
RepositoryFactory module analysis	1.6.1.3.5.6	2 days	Fri 05/03/21	Mon 08/03/21	
WorkerFactory module analysis	1.6.1.3.5.7	2 days	Fri 05/03/21	Mon 08/03/21	

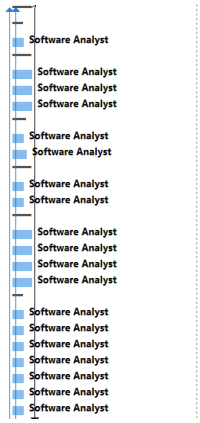


Figure 15: Initial Planning. WBS. EIISERVER module identification

4.1.3.4.1.2 EIIAPP MODULE IDENTIFICATION

EIIAPP module identification	**1.6.1.4**	**3 days**	**Fri 12/03/21**	**Tue 16/03/21**	**21**
Core module analysis	**1.6.1.4.1**	**2 days**	**Fri 12/03/21**	**Mon 15/03/21**	
Models module analysis	1.6.1.4.1.1	2 days	Fri 12/03/21	Mon 15/03/21	
Services module analysis	1.6.1.4.1.2	2 days	Fri 12/03/21	Mon 15/03/21	
ViewModels module analysis	1.6.1.4.1.3	2 days	Fri 12/03/21	Mon 15/03/21	
Datasources module analysis	1.6.1.4.2	3 days	Fri 12/03/21	Tue 16/03/21	
UI module analysis	**1.6.1.4.3**	**2 days**	**Fri 12/03/21**	**Mon 15/03/21**	
Views module analysis	1.6.1.4.3.1	2 days	Fri 12/03/21	Mon 15/03/21	
Widgets module analysis	1.6.1.4.3.2	2 days	Fri 12/03/21	Mon 15/03/21	
Locators module analysis	1.6.1.4.4	2 days	Fri 12/03/21	Mon 15/03/21	

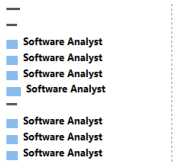


Figure 16: Initial Planning. WBS. EIIAPP module identification

4.1.3.4.2 SYSTEM DESIGN

System Study	**1.6**	**50 days**	**Thu 25/02/21**	**Fri 07/05/21**	
System Analysis	**1.6.1**	**14 days**	**Thu 25/02/21**	**Tue 16/03/21**	**9**
System Design	**1.6.2**	**11 days**	**Wed 17/03/21**	**Wed 31/03/21**	**18**
Class Design	1.6.2.1	1 day	Tue 23/03/21	Tue 23/03/21	103;104
EIISERVER Use Case Design	**1.6.2.2**	**4 days**	**Wed 24/03/21**	**Mon 29/03/21**	**62**
EIIAPP Use Case Design	**1.6.2.3**	**2 days**	**Tue 30/03/21**	**Wed 31/03/21**	**63**
System module architecture design	1.6.2.4	3 days	Wed 17/03/21	Fri 19/03/21	
Physical Data Design	1.6.2.5	4 days	Wed 17/03/21	Mon 22/03/21	
Technical Specification of the Testing Plan	1.6.2.6	4 days	Wed 17/03/21	Mon 22/03/21	

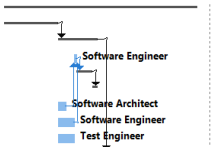


Figure 17: Initial Planning. WBS. System Design

4.1.3.4.2.1 EIISERVER USE CASE DESIGN

EIISERVER Use Case Design	**1.6.2.2**	**4 days**	**Wed 24/03/21**	**Mon 29/03/21**	**62**
Selection of Use Cases of Logger	**1.6.2.2.1**	**4 days**	**Wed 24/03/21**	**Mon 29/03/21**	
Selection of Use Cases of LoggerManager	1.6.2.2.1.1	4 days	Wed 24/03/21	Mon 29/03/21	
Selection of Use Cases of Core	**1.6.2.2.2**	**4 days**	**Wed 24/03/21**	**Mon 29/03/21**	
Selection of Use Cases of Entities	**1.6.2.2.2.1**	**4 days**	**Wed 24/03/21**	**Mon 29/03/21**	
Selection of Use Cases of Entity	1.6.2.2.2.1.4	days	Wed 24/03/21	Mon 29/03/21	
Selection of Use Cases of Interactors	**1.6.2.2.3**	**2 days**	**Wed 24/03/21**	**Thu 25/03/21**	
Selection of Use Cases of Interactor	1.6.2.2.3.1	2 days	Wed 24/03/21	Thu 25/03/21	
Selection of Use Cases of Datasources	**1.6.2.2.4**	**4 days**	**Wed 24/03/21**	**Mon 29/03/21**	
Selection of Use Cases of LDAPAuthenticationDatasource	1.6.2.2.4.1	4 days	Wed 24/03/21	Mon 29/03/21	
Selection of Use Cases of PUSHNotificationDatasource	1.6.2.2.4.2	4 days	Wed 24/03/21	Mon 29/03/21	
Selection of Use Cases of EntityDatasource	1.6.2.2.4.3	4 days	Wed 24/03/21	Mon 29/03/21	
Selection of Use Cases of Transport	**1.6.2.2.5**	**4 days**	**Wed 24/03/21**	**Mon 29/03/21**	
Selection of Use Cases of Loaders	1.6.2.2.5.1	2 days	Wed 24/03/21	Thu 25/03/21	
Selection of Use Cases of Middlewares	1.6.2.2.5.2	2 days	Wed 24/03/21	Thu 25/03/21	
Selection of Use Cases of Readers	**1.6.2.2.5.3**	**3 days**	**Wed 24/03/21**	**Fri 26/03/21**	
Selection of Use Cases of GIISOFReader	1.6.2.2.5.3.1	3 days	Wed 24/03/21	Fri 26/03/21	
Selection of Use Cases of MIWReader	1.6.2.2.5.3.2	3 days	Wed 24/03/21	Fri 26/03/21	
Selection of Use Cases of Workers	1.6.2.2.5.4	3 days	Wed 24/03/21	Fri 26/03/21	
Selection of Use Cases of Routes	1.6.2.2.5.5	4 days	Wed 24/03/21	Mon 29/03/21	
Selection of Use Cases of Factories	**1.6.2.2.6**	**2 days**	**Wed 24/03/21**	**Thu 25/03/21**	
Selection of Use Cases of ErrorHandlerFactory	1.6.2.2.6.1	2 days	Wed 24/03/21	Thu 25/03/21	
Selection of Use Cases of LoaderFactory	1.6.2.2.6.2	2 days	Wed 24/03/21	Thu 25/03/21	
Selection of Use Cases of ReaderFactory	1.6.2.2.6.3	2 days	Wed 24/03/21	Thu 25/03/21	
Selection of Use Cases of WorkerFactory	1.6.2.2.6.4	2 days	Wed 24/03/21	Thu 25/03/21	
Selection of Use Cases of LoggerFactory	1.6.2.2.6.5	2 days	Wed 24/03/21	Thu 25/03/21	
Selection of Use Cases of InteractorFactory	1.6.2.2.6.6	2 days	Wed 24/03/21	Thu 25/03/21	
Selection of Use Cases of RepositoryFactory	1.6.2.2.6.7	2 days	Wed 24/03/21	Thu 25/03/21	
Selection of Use Cases of ErrorHandlerFactory	1.6.2.2.6.8	2 days	Wed 24/03/21	Thu 25/03/21	

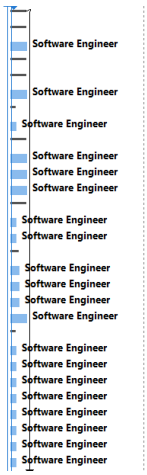


Figure 18: Initial Planning. WBS. EIISERVER Use Case Design

4.1.3.4.2.2 EIIAPP USE CASE DESIGN

EIIAPP Use Case Design	**1.6.2.3**	**2 days**	**Tue 30/03/21**	**Wed 31/03/21**	**63**
Selection of Use Cases of Locators	**1.6.2.3.1**	**2 days**	**Tue 30/03/21**	**Wed 31/03/21**	
Selection of Use Cases of Locator	1.6.2.3.1.1	2 days	Tue 30/03/21	Wed 31/03/21	
Selection of Use Cases of Core	**1.6.2.3.2**	**2 days**	**Tue 30/03/21**	**Wed 31/03/21**	
Selection of Use Cases of Models	1.6.2.3.2.1	2 days	Tue 30/03/21	Wed 31/03/21	
Selection of Use Cases of Services	1.6.2.3.2.2	2 days	Tue 30/03/21	Wed 31/03/21	
Selection of Use Cases of ViewModels	1.6.2.3.2.3	2 days	Tue 30/03/21	Wed 31/03/21	
Selection of Use Cases of Datasources	1.6.2.3.3	2 days	Tue 30/03/21	Wed 31/03/21	
Selection of Use Cases of UI	**1.6.2.3.4**	**2 days**	**Tue 30/03/21**	**Wed 31/03/21**	
Selection of Use Cases of Views	1.6.2.3.4.1	2 days	Tue 30/03/21	Wed 31/03/21	
Selection of Use Cases of Widgets	1.6.2.3.4.2	2 days	Tue 30/03/21	Wed 31/03/21	

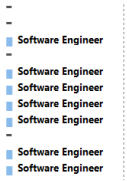


Figure 19: Initial Planning. WBS. EIIAPP Use Case Design

4.1.3.4.3 SYSTEM CONSTRUCTION

System Construction	**1.6.3**	**17 days**	**Mon 05/04/21**	**Tue 27/04/21**	**61**
Generation of the code of the components and procedures of EIISERVER	**1.6.3.1**	**9 days**	**Mon 05/04/21**	**Thu 15/04/21**	
Generation of the code of the components and procedures of EIIAPP	**1.6.3.2**	**6 days**	**Fri 16/04/21**	**Fri 23/04/21**	**107**
EIISERVER Tests Execution	**1.6.3.3**	**3 days**	**Fri 16/04/21**	**Tue 20/04/21**	**107**
EIISERVER Unit tests execution	1.6.3.3.1	3 days	Fri 16/04/21	Tue 20/04/21	
EIISERVER Acceptance tests execution	1.6.3.3.2	3 days	Fri 16/04/21	Tue 20/04/21	
EIIAPP Tests Execution	**1.6.3.4**	**2 days**	**Mon 26/04/21**	**Tue 27/04/21**	**133**
EIIAPP Unit Tests execution	1.6.3.4.1	2 days	Mon 26/04/21	Tue 27/04/21	
EIIAPP Widget Tests Execution	1.6.3.4.2	2 days	Mon 26/04/21	Tue 27/04/21	
EIIAPP Integration Tests execution	1.6.3.4.3	2 days	Mon 26/04/21	Tue 27/04/21	
User Manuals elaboration	1.6.3.5	5 days	Mon 05/04/21	Fri 09/04/21	

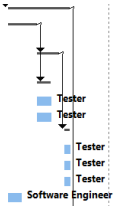


Figure 20: Initial Planning. WBS. System Construction

4.1.3.4.3.1 GENERATION OF THE CODE OF THE COMPONENTS AND PROCEDURES OF EIISERVER

Generation of the code of the components and procedures of EIISERVER	**1.6.3.1**	**9 days**	**Mon 05/04/21**	**Thu 15/04/21**	
Development of Logger	**1.6.3.1.1**	**5 days**	**Mon 05/04/21**	**Fri 09/04/21**	
Development of LoggerManager	1.6.3.1.1.1	5 days	Mon 05/04/21	Fri 09/04/21	
Development of Core	**1.6.3.1.2**	**9 days**	**Mon 05/04/21**	**Thu 15/04/21**	
Development of Entities	1.6.3.1.2.1	1 day	Mon 05/04/21	Mon 05/04/21	
Development of Interactors	1.6.3.1.2.2	9 days	Mon 05/04/21	Thu 15/04/21	
Development of Datasources	**1.6.3.1.3**	**7 days**	**Mon 05/04/21**	**Tue 13/04/21**	
Development of LDAPAuthenticationDatasource	1.6.3.1.3.1	7 days	Mon 05/04/21	Tue 13/04/21	
Development of EntityDatasource	1.6.3.1.3.2	7 days	Mon 05/04/21	Tue 13/04/21	
Development of PUSHNotificationDatasource	1.6.3.1.3.3	7 days	Mon 05/04/21	Tue 13/04/21	
Development of Transport	**1.6.3.1.4**	**9 days**	**Mon 05/04/21**	**Thu 15/04/21**	
Development of Loaders	1.6.3.1.4.1	4 days	Mon 05/04/21	Thu 08/04/21	
Development of Middlewares	1.6.3.1.4.2	5 days	Mon 05/04/21	Fri 09/04/21	
Development of Readers	**1.6.3.1.4.3**	**7 days**	**Mon 05/04/21**	**Tue 13/04/21**	
Development of GIISOFReaders	1.6.3.1.4.3.	7 days	Mon 05/04/21	Tue 13/04/21	
Development of MIWReaders	1.6.3.1.4.3.	7 days	Mon 05/04/21	Tue 13/04/21	
Development of Workers	1.6.3.1.4.4	1 day	Mon 05/04/21	Mon 05/04/21	
Development of Routes	1.6.3.1.4.5	9 days	Mon 05/04/21	Thu 15/04/21	
Development of Factories	**1.6.3.1.5**	**2 days**	**Mon 05/04/21**	**Tue 06/04/21**	
Development of ErrorHandlerFactory	1.6.3.1.5.1	2 days	Mon 05/04/21	Tue 06/04/21	
Development of LoaderFactory	1.6.3.1.5.2	2 days	Mon 05/04/21	Tue 06/04/21	
Development of ReaderFactory	1.6.3.1.5.3	2 days	Mon 05/04/21	Tue 06/04/21	
Development of WorkerFactory	1.6.3.1.5.4	2 days	Mon 05/04/21	Tue 06/04/21	
Development of LoggerFactory	1.6.3.1.5.5	2 days	Mon 05/04/21	Tue 06/04/21	
Development of InteractorFactory	1.6.3.1.5.6	2 days	Mon 05/04/21	Tue 06/04/21	
Development of RepositoryFactory	1.6.3.1.5.7	2 days	Mon 05/04/21	Tue 06/04/21	

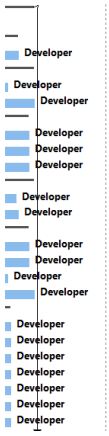


Figure 21: Initial Planning. WBS. Generation of the code of the components and procedures of EIISERVER

4.1.3.4.3.2 GENERATION OF THE CODE OF THE COMPONENTS AND PROCEDURES OF EIIAPP

Generation of the code of the components and procedures of EIIAPP	**1.6.3.2**	**6 days**	**Fri 16/04/21**	**Fri 23/04/21**	**107**
Development of Locators	**1.6.3.2.1**	**2 days**	**Fri 16/04/21**	**Mon 19/04/21**	
Development of Locator	1.6.3.2.1.1	2 days	Fri 16/04/21	Mon 19/04/21	
Development of Core	**1.6.3.2.2**	**6 days**	**Fri 16/04/21**	**Fri 23/04/21**	
Development of Models	1.6.3.2.2.1	2 days	Fri 16/04/21	Mon 19/04/21	
Development of Services	1.6.3.2.2.2	6 days	Fri 16/04/21	Fri 23/04/21	
Development of ViewModels	1.6.3.2.2.3	6 days	Fri 16/04/21	Fri 23/04/21	
Development of Datasources	**1.6.3.2.3**	**5 days**	**Fri 16/04/21**	**Thu 22/04/21**	
Development of Datasource	1.6.3.2.3.1	5 days	Fri 16/04/21	Thu 22/04/21	
Development of UI	**1.6.3.2.4**	**6 days**	**Fri 16/04/21**	**Fri 23/04/21**	
Development of Views	1.6.3.2.4.1	5 days	Fri 16/04/21	Thu 22/04/21	
Development of Widgets	1.6.3.2.4.2	6 days	Fri 16/04/21	Fri 23/04/21	

Figure 22: Initial Planning. WBS. Generation of the code of the components and procedures of EIIAPP

4.1.3.4.4 SYSTEM INTRODUCTION AND ACCEPTANCE

System Introduction and Acceptance	**1.6.4**	**2 days**	**Thu 06/05/21**	**Fri 07/05/21**	**157**	
Establishment of the Introduction Plan	1.6.4.1	2 days	Thu 06/05/21	Fri 07/05/21		
Uploading Data to the Operating Environment	1.6.4.2	2 days	Thu 06/05/21	Fri 07/05/21		
Presentation and approval of the system and going into productior	1.6.4.3	2 days	Thu 06/05/21	Fri 07/05/21		

Figure 23: Initial Planning. WBS. System Introduction and Acceptance

4.1.3.5 HARDWARE INSTALLATION

Hardware Installation	**1.7**	**6 days**	**Wed 28/04/21**	**Wed 05/05/21**	**106**	
Study of the needs of the client	1.7.1	6 days	Wed 28/04/21	Wed 05/05/21		
Installation of the server	**1.7.2**	**3 days**	**Wed 28/04/21**	**Fri 30/04/21**		
Server Configuration	1.7.2.1	2 days	Wed 28/04/21	Thu 29/04/21		
Deployment of EIISERVER on the Server	1.7.2.2	3 days	Wed 28/04/21	Fri 30/04/21		

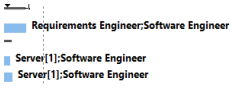


Figure 24: Initial Planning. WBS. Hardware Installation

4.1.4 RISKS

In this section the identified risks of the project are detailed, sorted by impact.

4.1.4.1 BUREAUCRATIC PROCESSES THAT DELAY THE PROJECT

The project might be delayed due to inherent bureaucratic processes: project acceptance, deployment environment preparation (by the client), external verification and acceptance (Google system verification), among others.

Category	Probability	Cost	Time	Scope	Quality	Impact
Organizational (Project Dependencies)	**Almost Certain**	**Extreme**	**Extreme**	**High**	**High**	**0.81**

- **Strategy**: Mitigate risk.
- **Response**: Each task prone to being delayed by any of these bureaucratic processes will suffer an increment in time, close to one or more weeks, on the whole, as a contingency.

4.1.4.2 DECISION TO USE TECHNOLOGIES INCOMPATIBLES

Some technologies, especially for the development of EIIAPP system, might be incompatible with the technology decided to use. For instance, the decision to use FirebaseMessaging API for notifications implies using Isolates and some persistence technologies (like **sqflite** [10]) are not compatible with Isolates.

Category	Probability	Cost	Time	Scope	Quality	Impact
Organizational (Project dependencies)	**Possible**	**High**	**High**	**High**	**High**	**0.39**

- **Strategy**: Eliminate risk.
- **Response**: Increase the time to study different technologies and its viability, and, thereby, root out possible incompatibilities between them.

4.1.4.3 REQUIREMENTS INFLATION

With the progress of the project, requirements that were not identified on the preliminary/initial phases start to emerge.

Category	Probability	Cost	Time	Scope	Quality	Impact
Technical (Requirements)	**Likely**	**High**	**High**	**High**	**High**	**0.39**

- **Strategy**: Mitigate risk.
- **Response**: Keep clients communicated about all progress made (via email communication channel and scheduling sessions to show the progress with tangible products) and ask them regularly about what they need to specify and refine all details.

4.1.4.4 GOOGLE DO NOT APPROVE EIIAPP SYSTEM

To use the "Export Events" module of EIIAPP system, Google needs to verify the system (check that EIIAPP uses Google API according to its Terms and Conditions) and approve it.

Category	Probability	Cost	Time	Scope	Quality	Impact
External (Subcontractors and Suppliers)	**Possible**	**Medium**	**High**	**High**	**Medium**	**0.28**

- **Strategy**: Mitigate risk.
- **Response**: To mitigate risk, a few weeks are used as contingency to have time to send Google a verification request more than once if required.

4.1.4.5 EIISERVER SOURCES OF INFORMATION FAIL OR ARE DOWN

EIISERVER uses the **official sources of the School of Computer Science** [3]–[5], to obtain the information of the system. Most of the sources are exposed via network. The problem appears if the websites are down since information cannot be reached/stored on the system.

Category	Probability	Cost	Time	Scope	Quality	Impact
External (Subcontractors and Suppliers)	**Possible**	**Low**	**Low**	**Low**	**High**	**0.28**

- **Strategy**: Transfer risk.
- **Response**: The School of Computer Science is the owner, responsible and provider of the sources/websites and project clients have agreed to use them.

4.1.4.6 WRONG PLANNING OF DATASOURCE MODULES

The development of the Datasources: "Entity Datasource", "PUSHNotificationDatasource", "LDAPAuthenticationDatasource" or "ExportCalendarDatasource" among others, implies dealing with different (and, in some cases, non-familiar) technologies. This could, in a wrong way, downplay the development time for this module.

Category	Probability	Cost	Time	Scope	Quality	Impact
Project Management (Planning)	**Possible**	**High**	**High**	**Medium**	**Medium**	**0.28**

- **Strategy**: Mitigate risk.
- **Response**: To mitigate risk, project planning for Datasource activities will consider this by adding more time.

4.1.4.7 DEVELOPMENT TEAM INEXPERIENCED IN THE TECHNOLOGIES USED IN EIIAPP AND EIISERVER

EIIAPP and EIISERVER uses technologies and environments non-familiar for the Development Team, such as Flutter, Dart or Typescript, among others.

Category	Probability	Cost	Time	Scope	Quality	Impact
Organizational (Resources)	**Likely**	**Medium**	**Medium**	**Medium**	**Medium**	**0.21**

- **Strategy**: Mitigate risk.
- **Response**: To mitigate risk, project planning for development activities will take in consideration this fact.

4.1.4.8 LATENCY/VELOCITY ISSUES ON THE SERVER WHERE EIISERVER IS DEPLOYED

EIISERVER is deployed on School premises. However, the server might experiment latency or velocity issues, making the user experience of both systems, specially EIIAPP, decrease.

Category	Probability	Cost	Time	Scope	Quality	Impact
Technical (Performance and Reliability)	**Possible**	**Medium**	**Medium**	**Low**	**Medium**	**0.15**

- **Strategy**: Transfer risk.
- **Response**: The School of Computer Science has agreed to deploy EIISERVER on such server and is responsible for the management of its infrastructure.

4.1.4.9 GOOGLE DECIDES TO STOP SUPPORTING FIREBASEMESSAGING

EIISERVER communicates with EIIAPP system via PUSH Notifications **using Firebase Messaging API**.

Category	Probability	Cost	Time	Scope	Quality	Impact
External (Subcontractors and Suppliers)	**Rare**	**Extreme**	**Extreme**	**High**	**High**	**0.09**

- **Strategy**: Assume risk.
- **Response**: It is not possible, for the scope of this project, to predict if Google is going to stop supporting Firebase Messaging.

4.1.4.10 LDAP SERVICE IS DOWN AND MAKES AUTHENTICATION ON EIISERVER IMPOSSIBLE

EIISERVER uses LDAP Authentication Service of the University of Oviedo to authenticate users on the system. The service is used throughout the net of universities of the University of Oviedo.

Category	Probability	Cost	Time	Scope	Quality	Impact
External (Subcontractors and Suppliers)	**Rare**	**Medium**	**Medium**	**Medium**	**Medium**	**0,03**

- **Strategy**: Transfer risk.
- **Response**: The School of Computer Science has agreed to use this authenticate method as unique point of entry to EIISERVER system.

4.1.5 INITIAL BUDGET

Cost Budget and Client Budget are shown in Cost Budget and Client Budget, respectively. To see the process followed, please see Budgeting.

4.1.5.1 COST BUDGET

Cost budget (53.981,25 € = 53.951,25 € + 30,00€) is shown in Table 5: Initial Budget. Cost Budget.

Entry	*Item*	*Description*	*Cost*	*Client Cost*
01		**Hardware (Installation)**	**2.210,00 €**	**2.764,04 €**
02		**System Planning**	**2.233,75 €**	**2.793,74 €**
03		**System Study**	**49.507,50 €**	**61.918,81 €**
	01	System Analysis	18.767,50 €	23.472,43 €
	02	System Design	15.550,00 €	19.448,31 €
	03	System Construction	15.060,00 €	18.835,47 €
	04	System Introduction and Acceptance	130,00 €	162,59 €
Total				**67.476,58 €**

Cost Budget	**Average value**
53.951,25 €	13.517,81 €
Other Costs	**Total**
30,00 €	53.921,25 €
Benefits (25%)	**Percentage**
13.487,81 €	0,250695459

Table 5: Initial Budget. Cost Budget

On the other hand, Summary Cost Budget is shown in Table 6: Initial Budget. Summary Cost Budget.

Cod.	Entry	Total
01	Hardware (Installation)	2.210,00 €
02	System Planning	2.233,75 €
03	System Study	49.507,50 €
03	Other Costs (Travel and Expense Allowance)	30,00 €
Total		53.981,25 €

Table 6: Initial Budget. Summary Cost Budget

4.1.5.2 CLIENT BUDGET

Detailed Client Budget (67.476,58 €) is shown in Table 7: Initial Budget. Detailed Client Budget.

Entry	*Item*	*Description*	*Price*	*Total*
01		Hardware (Installation)	2.764,04 €	2.764,04 €
02		System Planning	2.793,74 €	2.793,74 €
03		System Study		61.918,81 €
	01	System Analysis	23.472,43 €	
	02	System Design	19.448,31 €	
	03	System Construction	18.835,47 €	
	04	System Introduction and Acceptance	162,59 €	
Total				67.476,58 €

Table 7: Initial Budget. Detailed Client Budget

Summary Client Budget, on the other hand is shown in Table 8: Initial Budget. Summary Client Budget.

Code	*Description*	*Price*	*Total*
01	Hardware (Installation)	2.764,04 €	2.764,04 €
02	System Planning	2.793,74 €	2.793,74 €
03	System Study	61.918,81 €	61.918,81 €
Total			67.476,58 €

Table 8: Initial Budget. Summary Client Budget

4.2 EXECUTION OF THE PROJECT

4.2.1 PLANNING FOLLOW-UP PLAN

Three baselines were created throughout EIIProject to track and follow the previously mentioned project:

4.2.1.1 INITIAL BASELINE

The first baseline was set at the **beginning of EIIProject**, which started **the 2nd of February**.

4.2.1.2 HALFWAY BASELINE

The second baseline was set **after the construction of EIISERVER system** ("Generation of the code of the components and procedures of EIISERVER" activity), **the 25th of June**.

The project **followed the initial planning** to the letter excepting the activities included inside "Generation of the code of the components and procedures of EIISERVER". These activities are shown in Figure 25: Execution of the project. Halfway Baseline.

In the right part of the previously mentioned figure, **grey lines** represent the **activities scheduled** in the Initial Planning, and the **blue ones**, the **real execution** of EIIProject.

Generation of the code of the components and procedures of EIISERVER	**1.6.3.1**	**60 days**	**Mon 05/04/21**	**Fri 25/06/21**	
Development of Logger	**1.6.3.1.1**	**7 days**	**Mon 05/04/21**	**Tue 13/04/21**	
Development of LoggerManager	1.6.3.1.1.1	7 days	Mon 05/04/21	Tue 13/04/21	
Development of Core	**1.6.3.1.2**	**17 days**	**Wed 28/04/21**	**Thu 20/05/21**	**117**
Development of Entities	1.6.3.1.2.1	2 days	Wed 28/04/21	Thu 29/04/21	
Development of Interactors	1.6.3.1.2.2	17 days	Wed 28/04/21	Thu 20/05/21	
Development of Datasources	**1.6.3.1.3**	**21 days**	**Fri 21/05/21**	**Fri 18/06/21**	**110**
Development of LDAPAuthenticationDatasource	1.6.3.1.3.1	9 days	Fri 21/05/21	Wed 02/06/21	
Development of EntityDatasource	1.6.3.1.3.2	9 days	Fri 21/05/21	Wed 02/06/21	
Development of PUSHNotificationDatasource	1.6.3.1.3.3	21 days	Fri 21/05/21	Fri 18/06/21	
Development of Transport	**1.6.3.1.4**	**17 days**	**Mon 05/04/21**	**Tue 27/04/21**	
Development of Loaders	1.6.3.1.4.1	7 days	Mon 19/04/21	Tue 27/04/21	120
Development of Middlewares	1.6.3.1.4.2	10 days	Mon 05/04/21	Fri 16/04/21	
Development of Readers	**1.6.3.1.4.3**	**10 days**	**Mon 05/04/21**	**Fri 16/04/21**	
Development of GIISOFReaders	1.6.3.1.4.3.	10 days	Mon 05/04/21	Fri 16/04/21	
Development of MIWReaders	1.6.3.1.4.3.	10 days	Mon 05/04/21	Fri 16/04/21	
Development of Workers	1.6.3.1.4.4	3 days	Mon 05/04/21	Wed 07/04/21	
Development of Routes	1.6.3.1.4.5	11 days	Mon 05/04/21	Mon 19/04/21	
Development of Factories	**1.6.3.1.5**	**5 days**	**Mon 21/06/21**	**Fri 25/06/21**	**108;110;113;1**
Development of ErrorHandlerFactory	1.6.3.1.5.1	2 days	Mon 21/06/21	Tue 22/06/21	
Development of LoaderFactory	1.6.3.1.5.2	2 days	Mon 21/06/21	Tue 22/06/21	
Development of ReaderFactory	1.6.3.1.5.3	5 days	Mon 21/06/21	Fri 25/06/21	
Development of WorkerFactory	1.6.3.1.5.4	1 day	Mon 21/06/21	Mon 21/06/21	
Development of LoggerFactory	1.6.3.1.5.5	1 day	Mon 21/06/21	Mon 21/06/21	
Development of InteractorFactory	1.6.3.1.5.6	2 days	Mon 21/06/21	Tue 22/06/21	
Development of RepositoryFactory	1.6.3.1.5.7	1 day	Mon 21/06/21	Mon 21/06/21	

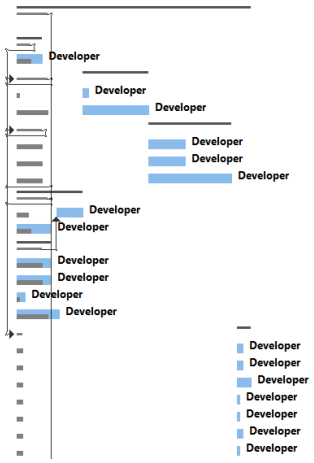


Figure 25: Execution of the project. Halfway Baseline

4.2.1.3 FINAL BASELINE

The final baseline **was set the 29th of October** (see Final Planning for more information).

4.2.2 PROJECT INCIDENTS LOG

A log of the main incidents that happened and threatened the project during its execution is listed down below:

- **05/04/2021**: LDAP Authentication Datasource required more certificates than the ones that were provided by the clients to communicate with the University of Oviedo's LDAP Authentication Service, specifically, an intermediate certificate that is now included in the project.
- **05/04/2021**: PUSH Notification Datasource construction was delayed due to the use of a non-suitable type of "notification" according to Firebase Guidelines. The notifications sent changed from "notifications" to "data messages" in order to make them persist across reboots and shutdowns. "notifications" are meant to be used for "marketing purposes" and the receival is not important, so notifications do not persist after reboots/shutdowns. On the other hand, "data messages" do persist after reboots/shutdowns.
- **20/04/2021**: A meeting via Microsoft Teams was scheduled for EIIServer acceptance and review. The clients were not satisfied with the product shown and requested some changes

in EIISERVER system. This implied a delay in the construction of the previously mentioned system.

- **27/07/2021**: EIIAPP was remodelled, as shown in Interface evolution, to represent the look and feel of the client in a better way and improve the interface. This was caused due to the lack of experience in Dart (see Development team inexperienced in the technologies used in EIIAPP and EIISERVER). So, after getting experience, EIIAPP system evolve **to improve the User Experience in detriment of time constraints** (as shown in Figure 26: Project Incidents Log. Project Triangle).

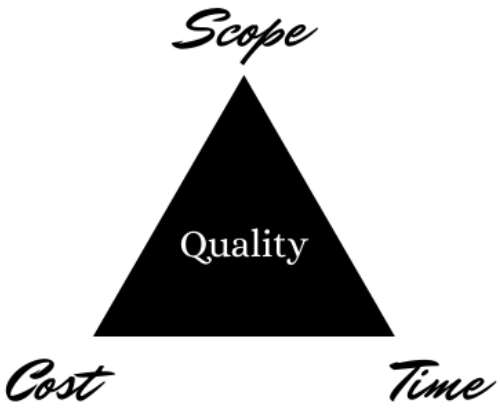


Figure 26: Project Incidents Log. Project Triangle

- **06/10/2021 – 19/10/2021**: The server where EIISERVER was deployed on, had a certificate that was expired, so a new one needed to be requested to the clients (another bureaucratic process that delayed EIIProject).
- **20/10/2021 - 30/10/2021**: Google verification process takes 8 working days (see Google do not Approve EIIAPP System). The verification request was delayed by the previous paragraph (we needed a valid certificate), "Domain Verification" leaded into a bureaucratic process (done by the School of Computer Science along with the University of Oviedo) and the verification was rejected several times.

4.2.3 RISKS

The risks that threatened EIIProject during its execution are detailed in this section (more information is detailed in Project incidents Log):

- On **20/04/2021**, the clients did not accept the project. This situation **was not fully contemplated** on the identified risks and leaded to a delay in the construction of EIIProject systems.
- **27/07/2021**: The risk "Development team inexperienced in the technologies used in EIIAPP and EIISERVER" leaded into the revamp of EIIAPP interface to apply all experience gained in this project.
- On **20/10/2021**, the clients needed to verify "uniovi.es" domain (DNS verification), which was another bureaucratic process (see Bureaucratic processes that delay the project) that delayed Google verification.

4.3 PROJECT CLOSURE

4.3.1 FINAL PLANNING

EIIProject **finished the 29th of October**, starting the 2nd of February. Its planning is structured in the next subsections:

4.3.1.1 PLANNING OF THE INFORMATION SYSTEM

2	**Planning of the information system**	**1.1**	**4 days**	**Tue 02/02/21**	**Fri 05/02/21**	
3	Definition and Organization of the PSI	1.1.1	2 days	Tue 02/02/21	Wed 03/02/21	
4	Study of relevant information	1.1.2	2 days	Thu 04/02/21	Fri 05/02/21	3

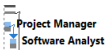


Figure 27: Final Planning. Planning of the information system

4.3.1.2 DEFINITION OF THE TECHNOLOGICAL ARCHITECTURE

5	**Definition of the technological architecture**	**1.2**	**5 days**	**Mon 08/02/21**	**Fri 12/02/21**	**2**
6	Identification of the needs of the technological infrastructure	1.2.1	1 day	Mon 08/02/21	Mon 08/02/21	
7	Selection of the architecture	1.2.2	4 days	Tue 09/02/21	Fri 12/02/21	6
8	Feasibility Study of the system	1.3	2 days	Mon 15/02/21	Tue 16/02/21	5

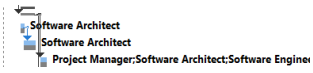


Figure 28: Final Planning. Definition of the technological architecture

4.3.1.3 PLANNING AND MANAGEMENT OF THE PROJECT

9	**Planning and management of the project**	**1.4**	**8 days**	**Mon 15/02/21**	**Wed 24/02/21**	**5**
10	Identification of the stakeholders	1.4.1	1 day	Mon 15/02/21	Mon 15/02/21	
11	OBS	1.4.2	1 day	Tue 16/02/21	Tue 16/02/21	10
12	PBS	1.4.3	1 day	Tue 16/02/21	Tue 16/02/21	10
13	WBS	1.4.4	3 days	Tue 16/02/21	Thu 18/02/21	10
14	Risks	1.4.5	3 days	Tue 16/02/21	Thu 18/02/21	10
15	Budget	1.4.6	4 days	Fri 19/02/21	Wed 24/02/21	11;12;13;14
16	Project Closure	1.5	3 days	Wed 27/10/21	Fri 29/10/21	156

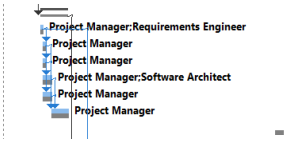


Figure 29: Final Planning. Planning and management of the project

4.3.1.4 SYSTEM STUDY

17	**System Study**	**1.6**	**172 days**	**Thu 25/02/21**	**Tue 26/10/21**	
18	**System Analysis**	**1.6.1**	**14 days**	**Thu 25/02/21**	**Tue 16/03/21**	**9**
61	**System Design**	**1.6.2**	**11 days**	**Wed 17/03/21**	**Wed 31/03/21**	**18**
106	**System Construction**	**1.6.3**	**129 days**	**Mon 05/04/21**	**Thu 30/09/21**	**61**
152	User Manuals elaboration	1.6.4	4 days	Wed 20/10/21	Mon 25/10/21	157
153	**System Introduction and Acceptance**	**1.6.5**	**15 days**	**Wed 06/10/21**	**Tue 26/10/21**	

Figure 30: Final Planning. System Study

4.3.1.4.1 SYSTEM ANALYSIS

	System Analysis	**1.6.1**	**14 days**	**Thu 25/02/21**	**Tue 16/03/21**	**9**
	System Definition	1.6.1.1	3 days	Thu 25/02/21	Mon 01/03/21	
	Requirements establishment	1.6.1.2	5 days	Tue 02/03/21	Mon 08/03/21	19
	EIISERVER module Identification	**1.6.1.3**	**5 days**	**Fri 05/03/21**	**Thu 11/03/21**	**57;10;58;59;60**
	EIIAPP module identification	**1.6.1.4**	**3 days**	**Fri 12/03/21**	**Tue 16/03/21**	**21**
	Class Analysis	1.6.1.5	3 days	Thu 25/02/21	Mon 01/03/21	
	Data modelling	1.6.1.6	6 days	Thu 25/02/21	Thu 04/03/21	
	Defining User Interfaces	1.6.1.7	3 days	Thu 25/02/21	Mon 01/03/21	
	Testing Plan Specification	1.6.1.8	5 days	Thu 25/02/21	Wed 03/03/21	

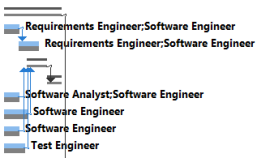


Figure 31: Final Planning. System Analysis

4.3.1.4.1.1 EIISERVER MODULE IDENTIFICATION

21	**EIISERVER module Identification**	**1.6.1.3**	**5 days**	**Fri 05/03/21**	**Thu 11/03/21**	**57;10;58;59;60**
22	**Logger module analysis**	**1.6.1.3.1**	**2 days**	**Fri 05/03/21**	**Mon 08/03/21**	
23	LoggerManager module analysis	1.6.1.3.1.1	2 days	Fri 05/03/21	Mon 08/03/21	
24	**Datasources module analysis**	**1.6.1.3.2**	**5 days**	**Fri 05/03/21**	**Thu 11/03/21**	
25	LDAPAuthenticationDatasource module analysis	1.6.1.3.2.1	5 days	Fri 05/03/21	Thu 11/03/21	
26	EntityDatasource module analysis	1.6.1.3.2.2	5 days	Fri 05/03/21	Thu 11/03/21	
27	PUSHNotificationDatasource module analysis	1.6.1.3.2.3	5 days	Fri 05/03/21	Thu 11/03/21	
28	**Core module analysis**	**1.6.1.3.3**	**3 days**	**Fri 05/03/21**	**Tue 09/03/21**	
29	Entities module analysis	1.6.1.3.3.1	2 days	Fri 05/03/21	Mon 08/03/21	
30	Interactors module analysis	1.6.1.3.3.2	3 days	Fri 05/03/21	Tue 09/03/21	
31	**Transport module analysis**	**1.6.1.3.4**	**5 days**	**Fri 05/03/21**	**Thu 11/03/21**	
32	Loaders module analysis	1.6.1.3.4.1	2 days	Fri 05/03/21	Mon 08/03/21	
33	Middlewares module analysis	1.6.1.3.4.2	2 days	Fri 05/03/21	Mon 08/03/21	
34	**Readers module analysis**	**1.6.1.3.4.3**	**5 days**	**Fri 05/03/21**	**Thu 11/03/21**	
35	GIISOFReader module analysis	1.6.1.3.4.3.1	5 days	Fri 05/03/21	Thu 11/03/21	
36	MIWReader module analysis	1.6.1.3.4.3.2	5 days	Fri 05/03/21	Thu 11/03/21	
37	Workers module analysis	1.6.1.3.4.4	5 days	Fri 05/03/21	Thu 11/03/21	
38	Routes module analysis	1.6.1.3.4.5	5 days	Fri 05/03/21	Thu 11/03/21	
39	**Factories module analysis**	**1.6.1.3.5**	**2 days**	**Fri 05/03/21**	**Mon 08/03/21**	
40	ErrorHandlerFactory module analysis	1.6.1.3.5.1	2 days	Fri 05/03/21	Mon 08/03/21	
41	LoggerFactory module analysis	1.6.1.3.5.2	2 days	Fri 05/03/21	Mon 08/03/21	
42	LoaderFactory module analysis	1.6.1.3.5.3	2 days	Fri 05/03/21	Mon 08/03/21	
43	ReaderFactory module analysis	1.6.1.3.5.4	2 days	Fri 05/03/21	Mon 08/03/21	
44	InteractorFactory module analysis	1.6.1.3.5.5	2 days	Fri 05/03/21	Mon 08/03/21	
45	RepositoryFactory module analysis	1.6.1.3.5.6	2 days	Fri 05/03/21	Mon 08/03/21	
46	WorkerFactory module analysis	1.6.1.3.5.7	2 days	Fri 05/03/21	Mon 08/03/21	

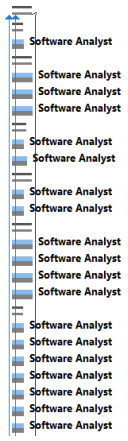


Figure 32: Final Planning. EIISERVER module identification

4.3.1.4.1.2 EIIAPP MODULE IDENTIFICATION

47	**EIIAPP module identification**	**1.6.1.4**	**3 days**	**Fri 12/03/21**	**Tue 16/03/21**	**21**
48	**Core module analysis**	**1.6.1.4.1**	**2 days**	**Fri 12/03/21**	**Mon 15/03/21**	
49	Models module analysis	1.6.1.4.1.1	2 days	Fri 12/03/21	Mon 15/03/21	
50	Services module analysis	1.6.1.4.1.2	2 days	Fri 12/03/21	Mon 15/03/21	
51	ViewModels module analysis	1.6.1.4.1.3	2 days	Fri 12/03/21	Mon 15/03/21	
52	Datasources module analysis	1.6.1.4.2	3 days	Fri 12/03/21	Tue 16/03/21	
53	**UI module analysis**	**1.6.1.4.3**	**2 days**	**Fri 12/03/21**	**Mon 15/03/21**	
54	Views module analysis	1.6.1.4.3.1	2 days	Fri 12/03/21	Mon 15/03/21	
55	Widgets module analysis	1.6.1.4.3.2	2 days	Fri 12/03/21	Mon 15/03/21	
56	Locators module analysis	1.6.1.4.4	2 days	Fri 12/03/21	Mon 15/03/21	

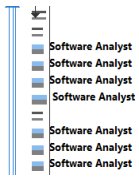


Figure 33: Final Planning. EIIAPP module identification

4.3.1.4.2 SYSTEM DESIGN

61	**System Design**	**1.6.2**	**11 days**	**Wed 17/03/21**	**Wed 31/03/21**	**18**
62	Class Design	1.6.2.1	1 day	Tue 23/03/21	Tue 23/03/21	103;104
63	**EIISERVER Use Case Design**	**1.6.2.2**	**4 days**	**Wed 24/03/21**	**Mon 29/03/21**	**62**
92	**EIIAPP Use Case Design**	**1.6.2.3**	**2 days**	**Tue 30/03/21**	**Wed 31/03/21**	**63**
103	System module architecture design	1.6.2.4	3 days	Wed 17/03/21	Fri 19/03/21	
104	Physical Data Design	1.6.2.5	4 days	Wed 17/03/21	Mon 22/03/21	
105	Technical Specification of the Testing Plan	1.6.2.6	4 days	Wed 17/03/21	Mon 22/03/21	

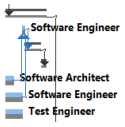


Figure 34: Final Planning. System Design

4.3.1.4.2.1 EIISERVER USE CASE DESIGN

63	**EIISERVER Use Case Design**	**1.6.2.2**	**4 days**	**Wed 24/03/21**	**Mon 29/03/21**	**62**
64	**Selection of Use Cases of Logger**	**1.6.2.2.1**	**4 days**	**Wed 24/03/21**	**Mon 29/03/21**	
65	Selection of Use Cases of LoggerManager	1.6.2.2.1.1	4 days	Wed 24/03/21	Mon 29/03/21	
66	**Selection of Use Cases of Core**	**1.6.2.2.2**	**4 days**	**Wed 24/03/21**	**Mon 29/03/21**	
67	**Selection of Use Cases of Entities**	**1.6.2.2.2.1**	**4 days**	**Wed 24/03/21**	**Mon 29/03/21**	
68	Selection of Use Cases of Entity	1.6.2.2.2.1.1	4 days	Wed 24/03/21	Mon 29/03/21	
69	**Selection of Use Cases of Interactors**	**1.6.2.2.3**	**2 days**	**Wed 24/03/21**	**Thu 25/03/21**	
70	Selection of Use Cases of Interactor	1.6.2.2.3.1	2 days	Wed 24/03/21	Thu 25/03/21	
71	**Selection of Use Cases of Datasources**	**1.6.2.2.4**	**4 days**	**Wed 24/03/21**	**Mon 29/03/21**	
72	Selection of Use Cases of LDAPAuthenticationDatasource	1.6.2.2.4.1	4 days	Wed 24/03/21	Mon 29/03/21	
73	Selection of Use Cases of PUSHNotificationDatasource	1.6.2.2.4.2	4 days	Wed 24/03/21	Mon 29/03/21	
74	Selection of Use Cases of EntityDatasource	1.6.2.2.4.3	4 days	Wed 24/03/21	Mon 29/03/21	
75	**Selection of Use Cases of Transport**	**1.6.2.2.5**	**4 days**	**Wed 24/03/21**	**Mon 29/03/21**	
76	Selection of Use Cases of Loaders	1.6.2.2.5.1	2 days	Wed 24/03/21	Thu 25/03/21	
77	Selection of Use Cases of Middlewares	1.6.2.2.5.2	2 days	Wed 24/03/21	Thu 25/03/21	
78	**Selection of Use Cases of Readers**	**1.6.2.2.5.3**	**3 days**	**Wed 24/03/21**	**Fri 26/03/21**	
79	Selection of Use Cases of GIISOFReader	1.6.2.2.5.3.1	3 days	Wed 24/03/21	Fri 26/03/21	
80	Selection of Use Cases of MIWReader	1.6.2.2.5.3.2	3 days	Wed 24/03/21	Fri 26/03/21	
81	Selection of Use Cases of Workers	1.6.2.2.5.4	3 days	Wed 24/03/21	Fri 26/03/21	
82	Selection of Use Cases of Routes	1.6.2.2.5.5	4 days	Wed 24/03/21	Mon 29/03/21	
83	**Selection of Use Cases of Factories**	**1.6.2.2.6**	**2 days**	**Wed 24/03/21**	**Thu 25/03/21**	
84	Selection of Use Cases of ErrorHandlerFactory	1.6.2.2.6.1	2 days	Wed 24/03/21	Thu 25/03/21	
85	Selection of Use Cases of LoaderFactory	1.6.2.2.6.2	2 days	Wed 24/03/21	Thu 25/03/21	
86	Selection of Use Cases of ReaderFactory	1.6.2.2.6.3	2 days	Wed 24/03/21	Thu 25/03/21	
87	Selection of Use Cases of WorkerFactory	1.6.2.2.6.4	2 days	Wed 24/03/21	Thu 25/03/21	
88	Selection of Use Cases of LoggerFactory	1.6.2.2.6.5	2 days	Wed 24/03/21	Thu 25/03/21	
89	Selection of Use Cases of InteractorFactory	1.6.2.2.6.6	2 days	Wed 24/03/21	Thu 25/03/21	
90	Selection of Use Cases of RepositoryFactory	1.6.2.2.6.7	2 days	Wed 24/03/21	Thu 25/03/21	
91	Selection of Use Cases of ErrorHandlerFactory	1.6.2.2.6.8	2 days	Wed 24/03/21	Thu 25/03/21	

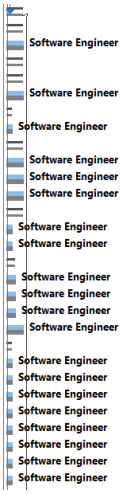


Figure 35: Final Planning. EIISERVER Use Case Design

4.3.1.4.2.2 EIIAPP USE CASE DESIGN

92	**EIIAPP Use Case Design**	**1.6.2.3**	**2 days**	**Tue 30/03/21**	**Wed 31/03/21**	**63**
93	**Selection of Use Cases of Locators**	**1.6.2.3.1**	**2 days**	**Tue 30/03/21**	**Wed 31/03/21**	
94	Selection of Use Cases of Locator	1.6.2.3.1.1	2 days	Tue 30/03/21	Wed 31/03/21	
95	**Selection of Use Cases of Core**	**1.6.2.3.2**	**2 days**	**Tue 30/03/21**	**Wed 31/03/21**	
96	Selection of Use Cases of Models	1.6.2.3.2.1	2 days	Tue 30/03/21	Wed 31/03/21	
97	Selection of Use Cases of Services	1.6.2.3.2.2	2 days	Tue 30/03/21	Wed 31/03/21	
98	Selection of Use Cases of ViewModels	1.6.2.3.2.3	2 days	Tue 30/03/21	Wed 31/03/21	
99	Selection of Use Cases of Datasources	1.6.2.3.3	2 days	Tue 30/03/21	Wed 31/03/21	
100	**Selection of Use Cases of UI**	**1.6.2.3.4**	**2 days**	**Tue 30/03/21**	**Wed 31/03/21**	
101	Selection of Use Cases of Views	1.6.2.3.4.1	2 days	Tue 30/03/21	Wed 31/03/21	
102	Selection of Use Cases of Widgets	1.6.2.3.4.2	2 days	Tue 30/03/21	Wed 31/03/21	

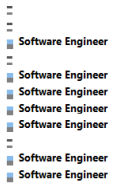


Figure 36: Final Planning. EIIAPP Use Case Design

4.3.1.4.3 SYSTEM CONSTRUCTION

106	**System Construction**	**1.6.3**	**129 days**	**Mon 05/04/21**	**Thu 30/09/21**	**61**
107	**Generation of the code of the components and procedures of EIISERVER**	**1.6.3.1**	**60 days**	**Mon 05/04/21**	**Fri 25/06/21**	
133	**Generation of the code of the components and procedures of EIIAPP**	**1.6.3.2**	**65 days**	**Mon 28/06/21**	**Fri 24/09/21**	**107**
145	**EIISERVER Tests Execution**	**1.6.3.3**	**5 days**	**Mon 28/06/21**	**Fri 02/07/21**	**107**
146	EIISERVER Unit tests execution	1.6.3.3.1	2 days	Mon 28/06/21	Tue 29/06/21	
147	EIISERVER Acceptance tests execution	1.6.3.3.2	5 days	Mon 28/06/21	Fri 02/07/21	
148	**EIIAPP Tests Execution**	**1.6.3.4**	**4 days**	**Mon 27/09/21**	**Thu 30/09/21**	**133**
149	EIIAPP Unit Tests execution	1.6.3.4.1	2 days	Mon 27/09/21	Tue 28/09/21	
150	EIIAPP Widget Tests Execution	1.6.3.4.2	2 days	Mon 27/09/21	Tue 28/09/21	
151	EIIAPP Integration Tests execution	1.6.3.4.3	4 days	Mon 27/09/21	Thu 30/09/21	
152	User Manuals elaboration	1.6.4	4 days	Wed 20/10/21	Mon 25/10/21	157

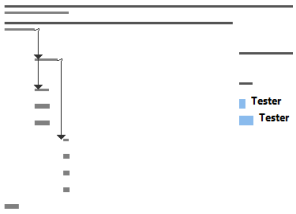


Figure 37: Final Planning. System Construction

4.3.1.4.3.1 GENERATION OF THE CODE OF THE COMPONENTS AND PROCEDURES OF EIISERVER

107	**Generation of the code of the components and procedures of EIISERVER**	**1.6.3.1**	**60 days**	**Mon 05/04/21**	**Fri 25/06/21**	
108	**Development of Logger**	**1.6.3.1.1**	**7 days**	**Mon 05/04/21**	**Tue 13/04/21**	
109	Development of LoggerManager	1.6.3.1.1.1	7 days	Mon 05/04/21	Tue 13/04/21	
110	**Development of Core**	**1.6.3.1.2**	**17 days**	**Wed 28/04/21**	**Thu 20/05/21**	**117**
111	Development of Entities	1.6.3.1.2.1	2 days	Wed 28/04/21	Thu 29/04/21	
112	Development of Interactors	1.6.3.1.2.2	17 days	Wed 28/04/21	Thu 20/05/21	
113	**Development of Datasources**	**1.6.3.1.3**	**21 days**	**Fri 21/05/21**	**Fri 18/06/21**	**110**
114	Development of LDAPAuthenticationDatasource	1.6.3.1.3.1	9 days	Fri 21/05/21	Wed 02/06/21	
115	Development of EntityDatasource	1.6.3.1.3.2	9 days	Fri 21/05/21	Wed 02/06/21	
116	Development of PUSHNotificationDatasource	1.6.3.1.3.3	21 days	Fri 21/05/21	Fri 18/06/21	
117	**Development of Transport**	**1.6.3.1.4**	**17 days**	**Mon 05/04/21**	**Tue 27/04/21**	
118	Development of Loaders	1.6.3.1.4.1	7 days	Mon 19/04/21	Tue 27/04/21	120
119	Development of Middlewares	1.6.3.1.4.2	10 days	Mon 05/04/21	Fri 16/04/21	
120	**Development of Readers**	**1.6.3.1.4.3**	**10 days**	**Mon 05/04/21**	**Fri 16/04/21**	
121	Development of GIISOFReaders	1.6.3.1.4.3.	10 days	Mon 05/04/21	Fri 16/04/21	
122	Development of MIWReaders	1.6.3.1.4.3.	10 days	Mon 05/04/21	Fri 16/04/21	
123	Development of Workers	1.6.3.1.4.4	3 days	Mon 05/04/21	Wed 07/04/21	
124	Development of Routes	1.6.3.1.4.5	11 days	Mon 05/04/21	Mon 19/04/21	
125	**Development of Factories**	**1.6.3.1.5**	**5 days**	**Mon 21/06/21**	**Fri 25/06/21**	**108;110;113;1**
126	Development of ErrorHandlerFactory	1.6.3.1.5.1	2 days	Mon 21/06/21	Tue 22/06/21	
127	Development of LoaderFactory	1.6.3.1.5.2	2 days	Mon 21/06/21	Tue 22/06/21	
128	Development of ReaderFactory	1.6.3.1.5.3	5 days	Mon 21/06/21	Fri 25/06/21	
129	Development of WorkerFactory	1.6.3.1.5.4	1 day	Mon 21/06/21	Mon 21/06/21	
130	Development of LoggerFactory	1.6.3.1.5.5	1 day	Mon 21/06/21	Mon 21/06/21	
131	Development of InteractorFactory	1.6.3.1.5.6	2 days	Mon 21/06/21	Tue 22/06/21	
132	Development of RepositoryFactory	1.6.3.1.5.7	1 day	Mon 21/06/21	Mon 21/06/21	

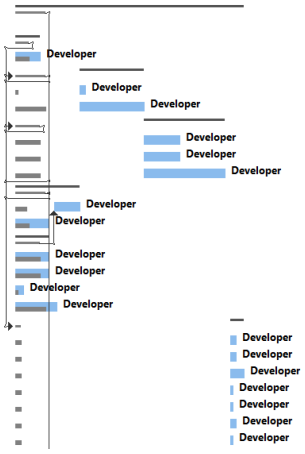


Figure 38: Final Planning. Generation of the code of the components and procedures of EIISERVER

4.3.1.4.3.2 GENERATION OF THE CODE OF THE COMPONENTS AND PROCEDURES OF EIIAPP

133	**Generation of the code of the components and procedures of EIIAPP**	**1.6.3.2**	**65 days**	**Mon 28/06/21**	**Fri 24/09/21**	**107**
134	**Development of Locators**	**1.6.3.2.1**	**2 days**	**Thu 23/09/21**	**Fri 24/09/21**	**136;140;142**
135	Development of Locator	1.6.3.2.1.1	2 days	Thu 23/09/21	Fri 24/09/21	
136	**Development of Core**	**1.6.3.2.2**	**28 days**	**Mon 26/07/21**	**Wed 01/09/21**	**142**
137	Development of Models	1.6.3.2.2.1	2 days	Mon 26/07/21	Tue 27/07/21	
138	Development of Services	1.6.3.2.2.2	16 days	Wed 11/08/21	Wed 01/09/21	139
139	Development of ViewModels	1.6.3.2.2.3	12 days	Mon 26/07/21	Tue 10/08/21	
140	**Development of Datasources**	**1.6.3.2.3**	**15 days**	**Thu 02/09/21**	**Wed 22/09/21**	**136**
141	Development of Datasource	1.6.3.2.3.1	15 days	Thu 02/09/21	Wed 22/09/21	
142	**Development of UI**	**1.6.3.2.4**	**20 days**	**Mon 28/06/21**	**Fri 23/07/21**	
143	Development of Views	1.6.3.2.4.1	20 days	Mon 28/06/21	Fri 23/07/21	
144	Development of Widgets	1.6.3.2.4.2	20 days	Mon 28/06/21	Fri 23/07/21	

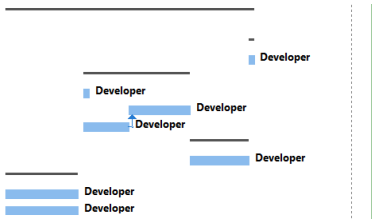


Figure 39: Final Planning. Generation of the code of the components and procedures of EIIAPP

4.3.1.4.4 SYSTEM INTRODUCTION AND ACCEPTANCE

153	**System Introduction and Acceptance**	**1.6.5**	**15 days**	**Wed 06/10/21**	**Tue 26/10/21**	
154	Establishment of the Introduction Plan	1.6.5.1	1 day	Wed 06/10/21	Wed 06/10/21	
155	Uploading Data to the Operating Environment	1.6.5.2	1 day	Thu 14/10/21	Thu 14/10/21	
156	Presentation and approval of the system and going into productior	1.6.5.3	1 day	Tue 26/10/21	Tue 26/10/21	

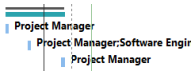


Figure 40: Final Planning. System Introduction and acceptance

4.3.1.4.5 HARDWARE INSTALLATION

158	Study of the needs of the client	1.7.1	2 days	Wed 06/10/21	Thu 07/10/21	
159	**Installation of the server**	**1.7.2**	**8 days**	**Fri 08/10/21**	**Tue 19/10/21**	**158**
160	Server Configuration	1.7.2.1	5 days	Fri 08/10/21	Thu 14/10/21	
161	Deployment of EIISERVER on the Server	1.7.2.2	3 days	Fri 15/10/21	Tue 19/10/21	160

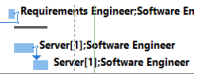


Figure 41: Final Planning. Hardware Installation

4.3.2 FINAL RISK REPORT

The main risks that threatened EIIProject are illustrated in Table 9: Final Risk Report. This report is complemented by Project incidents Log.

Date	Risk	Risk Information
20/04/2021	The clients did not accept the project. This situation **was not fully contemplated** on the identified risks and leaded to a delay in the construction of EIIProject systems	Unidentified Risk
27/07/2021	EIIAPP interface was revamped to apply all experience gained in this project	Development team inexperienced in the technologies used in EIIAPP and EIISERVER
20/10/2021	The clients needed to verify "uniovi.es" domain (DNS verification)	Bureaucratic processes that delay the project

Table 9: Final Risk Report

4.3.3 FINAL BUDGET

Final Cost Budget and Final Client Budget are shown in Cost Budget and Client Budget, respectively. To see the process followed, please see Final Budgeting.

4.3.3.1 COST BUDGET

Final Cost budget (**56.101,25 € = 56.071,25 € + 30,00€**) is shown in Table 10: Final Budget. Cost Budget.

Entry	*Item*	*Description*	*Cost*	*Client Cost*
01		**Hardware (Installation)**	**2.015,00 €**	**2.520,10 €**
02		**System Planning**	**2.233,75 €**	**2.793,68 €**
03		**System Study**	**51.822,50 €**	**64.812,80 €**
	01	System Analysis	18.767,50 €	23.471,93 €
	02	System Design	15.550,00 €	19.447,91 €
	03	System Construction	17.472,50 €	21.852,32 €
	04	System Introduction and Acceptance	32,50 €	40,65 €
Total				**70.126,58 €**

Cost Budget	**Average value**
56.071,25 €	14.047,81 €
Other Costs	**Total**
30,00 €	56.041,25 €
Benefits (25%)	**Percentage**
14.017,81 €	0,25066915

Table 10: Final Budget. Cost Budget

On the other hand, Final Summary Cost Budget is shown in Table 11: Final Budget. Summary Cost Budget.

Cod.	Entry	Total
01	Hardware (Installation)	2.015,00 €
02	System Planning	2.233,75 €
03	System Study	51.822,50 €
03	Other Costs (Travel and Expense Allowance)	30,00 €
Total		56.101,25 €

Table 11: Final Budget. Summary Cost Budget

4.3.3.2 CLIENT BUDGET

Final Detailed Client Budget (70.126,58 €) is shown in Table 12: Final Budget. Detailed Client Budget.

Entry	*Item*	*Description*	*Price*	*Total*
01		Hardware (Installation)	2.520,10 €	2.520,10 €
02		System Planning	2.793,68 €	2.793,68 €
03		System Study		64.812,80 €
	01	System Analysis	23.471,93 €	
	02	System Design	19.447,91 €	
	03	System Construction	21.852,32 €	
	04	System Introduction and Acceptance	40,65 €	
Total				70.126,58 €

Table 12: Final Budget. Detailed Client Budget

Final Summary Client Budget, on the other hand is shown in Table 13: Final Budget. Summary Client Budget.

Code	*Description*	*Price*	*Total*
01	Hardware (Installation)	2.520,10 €	2.520,10 €
02	System Planning	2.793,68 €	2.793,68 €
03	System Study	64.812,80 €	64.812,80 €
Total			70.126,58 €

Table 13: Final Budget. Summary Client Budget

4.3.4 LESSONS LEARNED REPORT

Some of the main lessons learned from this project are summarized below:

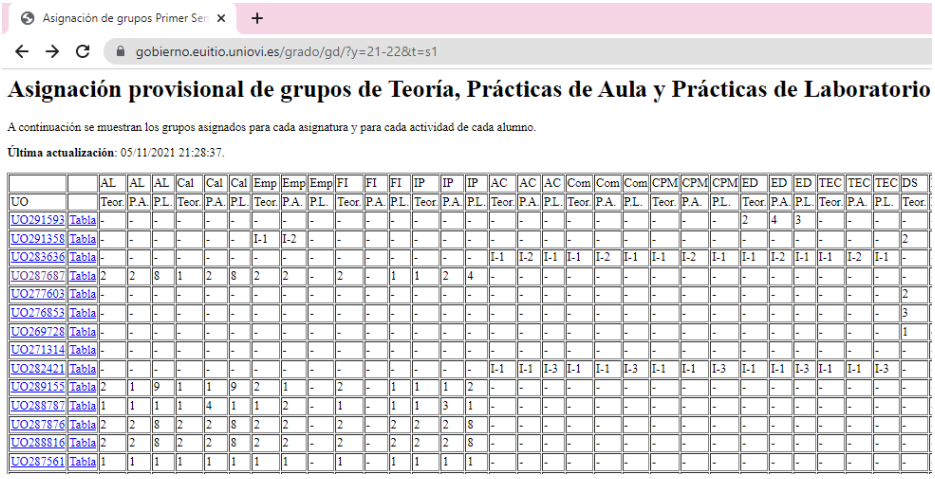
- A **Planning** is the decisive entry point of any Project, and it requires a good dose of knowledge and **experience** to fine-tune the estimations.
- The same goes for **Risk Management** since risks are inherent to all projects by nature and threat the success of these ones.
- **Quality is not free** and needs a balance between Scope, Cost and Time, i.e., resources are limited.
- Knowing **what we need to do** is the key part of every project that can lead us to the success as well as to the failure if no required attention is being paid.

5 ANALYSIS OF THE INFORMATION SYSTEM

5.1 SYSTEM DEFINITION

5.1.1 DETERMINATION OF THE SCOPE OF THE SYSTEM

In the present, the School of Computer Science **communicates with its students** of the Software Engineering Degree and Master in Web Engineering only via email, or the forum provided by the Virtual Campus (this could be unsuitable, for instance, if a lecturer is on leave, and students needs to be communicated). The School has **no official communication channel**.



Asignación de grupos Primer Se

gobierno.euitio.uniovi.es/grado/gd/?y=21-22&t=s1

Asignación provisional de grupos de Teoría, Prácticas de Aula y Prácticas de Laboratorio

A continuación se muestran los grupos asignados para cada asignatura y para cada actividad de cada alumno.

Última actualización: 05/11/2021 21:28:37.

		AL	AL	AL	Cal	Cal	Cal	Emp	Emp	Emp	FI	FI	FI	IP	IP	IP	AC	AC	AC	Com	Com	Com	CPM	CPM	CPM	ED	ED	ED	TEC	TEC	TEC	DS
UO		Teor.	P.A.	P.L.	Teor.	P.A.	P.L.	Teor.	P.A.	P.L.	Teor.	P.A.	P.L.	Teor.	P.A.	P.L.	Teor.	P.A.	P.L.	Teor.	P.A.	P.L.	Teor.	P.A.	P.L.	Teor.	P.A.	P.L.	Teor.	P.A.	P.L.	Teor.
UO291593	Tabla	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	4	3	-	-	-	-
UO291358	Tabla	-	-	-	-	-	I-1	I-2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2
UO283636	Tabla	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	I-1	I-2	I-1	I-1	I-2	I-1	I-1	I-2	I-1	I-1	I-2	I-1	I-1	I-2	I-1	-
UO287687	Tabla	2	2	8	1	2	8	2	2	-	2	-	1	1	2	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
UO277603	Tabla	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2
UO276853	Tabla	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	3
UO269728	Tabla	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
UO271314	Tabla	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
UO282421	Tabla	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	I-1	I-1	I-3	I-1	I-1	I-3	I-1	I-1	I-3	I-1	I-1	I-3	I-1	I-1	I-3	-
UO289155	Tabla	2	1	9	1	1	9	2	1	-	2	-	1	1	1	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
UO288787	Tabla	1	1	1	1	4	1	1	2	-	1	-	1	1	3	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
UO287876	Tabla	2	2	8	2	2	8	2	2	-	2	-	2	2	2	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
UO288816	Tabla	2	2	8	2	2	8	2	2	-	2	-	2	2	2	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
UO287561	Tabla	1	1	1	1	1	1	1	1	-	1	-	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 42: System Definition. Website

These School students are also able to **check their schedules** accessing the website of the School created for that purpose (see Figure 42: System Definition. Website). The schedule is displayed on a tabular format and can be exported to a CSV Excel file (see Figure 43: System Definition. Schedules). However, if the website is not available temporary or the student has no internet connection, and its schedule has not been previously exported or downloaded, he **will not know his timetable**, being **offline**.

Planificación de asignaturas

gobierno.euitio.uniovi.es/grado/plan/plan.php?vista=tabla&y=21-22&t=s1&FI_T_1=FI.T.1&FI_S_-=

También puedes verlo en otros formatos:

- Web
- CSV

	Viernes, 10/09/2021	Sábado, 11/09/2021	Domingo, 12/09/2021		Lunes, 13/09/2021	Martes, 14/09/2021	Miércoles, 15/09/2021	Jueves, 16/09/2021	Viernes, 17/09/2021	Sábado, 18/09/2021	Domingo, 19/09/2021
9.00-9.30	FI.L.1			9.00-9.30			DS.L.1		FI.L.1		
9.30-10.00	FI.L.1			9.30-10.00			DS.L.1		FI.L.1		
10.00-10.30	FI.L.1			10.00-10.30			DS.L.1		FI.L.1		
10.30-11.00	FI.L.1			10.30-11.00			DS.L.1		FI.L.1		
11.00-11.30				11.00-11.30		DS.T.2					
11.30-12.00				11.30-12.00		DS.T.2					
12.00-12.30	DS.S.4			12.00-12.30		DS.T.2					
12.30-13.00	DS.S.4			12.30-13.00		DS.T.2					
13.00-13.30				13.00-13.30		FI.T.1		FI.T.1			
13.30-14.00				13.30-14.00		FI.T.1		FI.T.1			

Figure 43: System Definition. Schedules

The **objective** of the developed system, hereinafter **EIISERVER** and **EIIAPP**, is to create a **centralized** official **application**, EIIAPP, for the School where students, identified by their **UO**, can **receive notifications** and check their **schedules**, **events** and **information of their interest**.

The server, **EIISERVER**, will provide **EIIAPP** with the information about **events** (sessions and exams), and **Links of Interest** (dynamic information managed in **EIISERVER**). **EIIAPP** shall let an identified student, update its information manually, as well as periodically by scheduling daily or weekly updates. To get it, **EIIAPP**, will communicate with the server, **EIISERVER**, to obtain such information customized by student's **UO**.

EIISERVER will communicate with the **LDAP** Authentication Service of the University of Oviedo to authenticate the users in the system. **EIISERVER** shall let an identified user with the role of **NOTIFIER**, send a broadcast notification, via **Firebase Messaging** API, to one or more **academic degrees** (received by students enrolled at the Software Engineering Degree or the Master in Web Engineering), **academic years** (received by students enrolled at the first, second, third or fourth year of the Software Engineering Degree; or first, second year of the Master in Web Engineering), **courses** (received by students enrolled at AC, CPM or SSI, among other courses), **groups** (received by students enrolled at AC T.1, AC S.2, CPM L.1 or SSI L.I-2, among other groups), **lecturers** (received by students that attend to sessions taught by a lecturer) or **students themselves**.

EIISERVER shall let an identified user with the role of **ADMINISTRATOR**, update the information stored related to the Software Engineering Degree and the Master in Web Engineering manually, as well as periodically by scheduling daily or weekly updates, as well as the information contained in the **Links of Interest**.

EIIAPP shall enable an identified student access all his information: events, notifications and Links of Interest, online as well as **offline**. Students identified on EIIAPP will also be able to **report a breakdown** of a computer system on the School premises and **export** their events **to Google Calendar**, **EIIAPP** will communicate with **Google OAuth** to authenticate the user and **Google Calendar API** to export the events.

5.2 REQUIREMENTS ESTABLISHMENT

The requirements of EIISERVER and EIIAPP systems are in EIISERVER Requirements and EIIAPP Requirements sections, respectively.

5.2.1 EIISERVER REQUIREMENTS

5.2.1.1 AUTHENTICATION

RSERVAuth1. EIISERVER shall let a non-authenticated user to authenticate.

RSERVAuth1.1. The next data will be requested for authentication:

RSERVAuth1.1.1. An identifier: username.

RSERVAuth1.1.1.1. Mandatory data.

RSERVAuth1.1.2. A password.

RSERVAuth1.1.2.1. Mandatory data.

RSERVAuth1.2. EIISERVER shall verify that the identifier (**RSERVAuth1.1.1**) is registered on the database.

RSERVAuth1.2.1. If **RSERVAuth1.2** is true, then the system shall communicate with the LDAP Authentication Service to authenticate the user.

RSERVAuth1.2.1.1. EIISERVER shall send the identifier (**RSERVAuth1.1.1**) and password (**RSERVAuth1.1.2**) to the LDAP Authentication Service.

RSERVAuth1.2.1.2. EIISERVER shall collect the response from the LDAP Authentication Service:

RSERVAuth1.2.1.2.1. If collected response is OK, the user will be authenticated.

RSERVAuth1.2.1.2.1.1. EIISERVER shall obtain the ROLES of the authenticated user from the database.

RSERVAuth1.2.1.2.1.1.1. If ROLES only contain ADMINISTRATOR role, the user will be authenticated as **ADMINISTRATOR** (see **RSERVAdmin1**, **RSERVRole1**).

RSERVAuth1.2.1.2.1.1.2. If ROLES only contain NOTIFIER role, the user will be authenticated as **NOTIFIER** (see **RSERVNotif1**, **RSERVRole1**).

RSERVAuth1.2.1.2.1.1.3. If ROLES contain NOTIFIER and ADMINISTRATOR role, the user will be authenticated as **ADMINISTRATOR** (see **RSERVAdmin1**, **RSERVRole1**).

RSERVAuth1.2.1.2.2. If collected response is ERROR, EIISERVER shall show an error message (the user will be kept non-authenticated).

RSERVAuth1.2.2. If **RSERVAuth1.2** is false, EIISERVER shall show an error message (the user will be kept non-authenticated).

5.2.1.2 AUTHENTICATED USER

RSERVRole1. EIISERVER shall let an authenticated user change its role.

RSERVRole1.1. The next data will be requested:

RSERVRole1.1.1. Role.

RSERVRole1.1.1.1. Mandatory data.

RSERVRole1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidRole1**.

RSERVRole1.1.2. If **RSERVRole1.1.1.1.** validation returns true, EIISERVER shall verify that a tuple (User Identifier (**RSERVAuth1.1.1**), Role (**RSERVRole1.1.1**)) exists on the database.

RSERVRole1.1.2.1. If **RSERVRole1.1.1** is true, user will be authenticated as Role (**RSERVRole1.1.1**).

RSERVRole1.1.2.2. If **RSERVRole1.1.1** is false, EIISERVER shall show an error message.

RSERVRole1.1.3. If **RSERVRole1.1.1.1.** validation returns false, EIISERVER shall show an error message.

5.2.1.3 NOTIFIER

RSERVNotif1. EIISERVER shall let a user authenticated as NOTIFIER send a notification.

RSERVNotif1.1. The next data will be requested for sending:

RSERVNotif1.1.1. A title.

RSERVNotif1.1.1.1. Mandatory data.

RSERVNotif1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidTitle1**.

RSERVNotif1.1.2. A content.

RSERVNotif1.1.2.1. Mandatory data.

RSERVNotif1.1.2.2. EIISERVER shall verify that is valid according to **RSERVValidContent1**.

RSERVNotif1.1.3. Recipients.

RSERVNotif1.1.3.1. Mandatory data.

RSERVNotif1.1.3.2. EIISERVER shall verify that is valid according to **RSERVValidRecipients1**.

RSERVNotif1.2. If **RSERVNotif1.1.1.2**, **RSERVNotif1.1.2.2** or **RSERVNotif1.1.3.2** validation returns false, EIISERVER shall show an error message.

RSERVNotif1.3. If **RSERVNotif1.1.1.2**, **RSERVNotif1.1.2.2** and **RSERVNotif1.1.3.2** validation returns true, EIISERVER shall communicate with Firebase Messaging API for sending.

RSERVNotif1.3.1. EIISERVER shall send the title (**RSERVNotif1.1.1**), content (**RSERVNotif1.1.2**) and recipients (**RSERVNotif1.1.3**) to Firebase Messaging API.

RSERVNotif1.3.2. EIISERVER shall collect the response from Firebase Messaging API:

RSERVNotif1.3.2.1. If collected response is OK, EIISERVER shall show a message indicating the number of notifications sent.

RSERVNotif1.3.2.2. If collected response is ERROR, EIISERVER shall show an error message.

5.2.1.4 ADMINISTRATOR

RSERVAdmin1. EIISERVER shall let a user authenticated as ADMINISTRATOR do the following actions:

RSERVAdmin1.1. Create an Academic Degree (see **RCreateAcDegree1**).

RSERVAdmin1.2. Delete an Academic Degree (see **RDeleteAcDegree1**).

RSERVAdmin1.3. Consult all Academic Degrees (see **RConsultAcDegree1**).

RSERVAdmin1.4. Create an Academic Year (see **RCreateAcYear1**).

RSERVAdmin1.5. Delete an Academic Year (see **RDeleteAcYear1**).

RSERVAdmin1.6. Consult all Academic Years (see **RConsultAcYear1**).

RSERVAdmin1.7. Create a Course (see **RCreateCourse1**).

RSERVAdmin1.8. Delete a Course (see **RDeleteCourse1**).

RSERVAdmin1.9. Consult all Courses (see **RConsultCourse1**).

RSERVAdmin1.10. Create a Group (see **RCreateGroup1**).

RSERVAdmin1.11. Delete a Group (see **RDeleteGroup1**).

RSERVAdmin1.12. Consult all Groups (see **RConsultGroup1**).

RSERVAdmin1.13. Create a Student (see **RCreateStudent1**).

RSERVAdmin1.14. Delete a Student (see **RDeleteStudent1**).

RSERVAdmin1.15. Consult all Students (see **RConsultStudent1**).

RSERVAdmin1.16. Create an Attendance (see **RCreateAttendance1**).

RSERVAdmin1.17. Delete an Attendance (see **RDeleteAttendance1**).

RSERVAdmin1.18. Consult all Attendances (see **RConsultAttendance1**).

RSERVAdmin1.19. Create a Lecturer (see **RCreateLecturer1**).

RSERVAdmin1.20. Delete a Lecturer (see **RDeleteLecturer1**).

RSERVAdmin1.21. Consult all Lecturers (see **RConsultLecturer1**).

RSERVAdmin1.22. Create a Teach (see **RCreateTeach1**).

RSERVAdmin1.23. Delete a Teach (see **RDeleteTeach1**).

RSERVAdmin1.24. Consult all Teaches (see **RConsultTeach1**).

RSERVAdmin1.25. Create a Session (see **RCreateSession1**).

RSERVAdmin1.26. Delete a Session (see **RDeleteSession1**).

RSERVAdmin1.27. Consult all Sessions (see **RConsultSession1**).

RSERVAdmin1.28. Create an Exam (see **RCreateExam1**).

RSERVAdmin1.29. Delete an Exam (see **RDeleteExam1**).

RSERVAdmin1.30. Consult all Exams (see **RConsultExam1**).

RSERVAdmin1.31. Create a Non-Working Day (see **RCreateNonWDay1**).

RSERVAdmin1.32. Delete a Non-Working Day (see **RDeleteNonWDay1**).

RSERVAdmin1.33. Consult all Non-Working Days (see **RConsultNonWDay1**).

RSERVAdmin1.34. Create an Authorized User (see **RCreateAuthUser1**).

RSERVAdmin1.35. Delete an Authorized User (see **RDeleteAuthUser1**).

RSERVAdmin1.36. Consult all Authorized Users (see **RConsultAuthUser1**).

RSERVAdmin1.37. Edit a Lecturer (see **REditLecturer1**).

RSERVAdmin1.38. Edit a Session (see **REditSession1**).

RSERVAdmin1.39. Edit an Exam (see **REditExam1**).

RSERVAdmin1.40. Edit the Links of Interest Page (see R**EditLinks1**).

RSERVAdmin1.41. Consult the Links of Interest Page (see R**ConsultLinks1**).

RSERVAdmin1.42. Update the information stored on the database (see **RUpdate1**).

RSERVAdmin1.43. Schedule an update (see **RSchedule1)**.

5.2.1.5 ACADEMIC DEGREE

RCreateAcDegree1. EIISERVER shall let a user authenticated as ADMINISTRATOR create an Academic Degree:

RCreateAcDegree1.1. The next data will be requested:

RCreateAcDegree1.1.1. Academic Degree.

RCreateAcDegree1.1.1.1. Mandatory data.

RCreateAcDegree1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidAcDegree1**.

RCreateAcDegree1.1.2. If **RCreateAcDegree1.1.1.2** validation returns true, EIISERVER shall store the Academic Degree.

RCreateAcDegree1.1.2.1. EIISERVER shall store Academic Degree (**RCreateAcDegree1.1.1**) on the database.

RCreateAcDegree1.1.2.2. EIISERVER shall show a message indicating the creation succeed.

RCreateAcDegree1.1.3. If **RCreateAcDegree1.1.1.2** validation returns false, EIISERVER shall show an error message.

RDeleteAcDegree1. EIISERVER shall let a user authenticated as ADMINISTRATOR delete an Academic Degree:

RDeleteAcDegree1.1. The next data will be requested:

RDeleteAcDegree1.1.1. Academic Degree.

RDeleteAcDegree1.1.1.1. Mandatory data.

RDeleteAcDegree1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidAcDegree1**.

RDeleteAcDegree1.1.2. If **RDeleteAcDegree1.1.1.2** validation returns true, EIISERVER shall verify that Academic Degree is registered on the database.

RSERVAdmin1.1.2.1. If **RDeleteAcDegree1.1.2** is true, EIISERVER shall remove Academic Degree from the database.

RSERVAdmin1.1.2.2. If **RDeleteAcDegree1.1.2** is false, EIISERVER shall show an error message.

RDeleteAcDegree1.1.3. If **RDeleteAcDegree1.1.1.2** validation returns false, EIISERVER shall show an error message.

RConsultAcDegree1. EIISERVER shall let a user authenticated as ADMINISTRATOR consult all Academic Degrees:

RConsultAcDegree1.1. EIISERVER shall obtain all Academic Degrees from Database.

RConsultAcDegree1.1.1. EIISERVER shall show the following fields for each Academic Degree obtained:

RConsultAcDegree1.1.1.1. Academic Degree Code.

5.2.1.6 ACADEMIC YEAR

RCreateAcYear1. EIISERVER shall let a user authenticated as ADMINISTRATOR create an Academic Year:

RCreateAcYear1.1. The next data will be requested:

RCreateAcYear1.1.1. Academic Year.

RCreateAcYear1.1.1.1. Mandatory data.

RCreateAcYear1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidAcYear1**.

RCreateAcYear1.1.2. If **RCreateAcYear1.1.1.2** validation returns true, EIISERVER shall store the Academic Year.

RCreateAcYear1.1.2.1. EIISERVER shall store Academic Year (**RCreateAcYear1.1.1**) on the database.

RCreateAcYear1.1.2.2. EIISERVER shall show a message indicating the creation succeed.

RCreateAcYear1.1.3. If **RCreateAcYear1.1.1.2** validation returns false, EIISERVER shall show an error message.

RDeleteAcYear1. EIISERVER shall let a user authenticated as ADMINISTRATOR delete an Academic Year:

RDeleteAcYear1.1. The next data will be requested:

RDeleteAcYear1.1.1. Academic Year.

RDeleteAcYear1.1.1.1. Mandatory data.

RDeleteAcYear1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidAcYear1**.

RDeleteAcYear1.1.2. If **RDeleteAcYear1.1.1.2** validation returns true, EIISERVER shall verify that Academic Year is registered on the database.

RDeleteAcYear1.1.2.1. If **RDeleteAcYear1.1.2** is true, EIISERVER shall remove Academic Year from the database.

RDeleteAcYear1.1.2.2. If **RDeleteAcYear1.1.2** is false, EIISERVER shall show an error message.

RDeleteAcYear1.1.3. If **RDeleteAcYear1.1.1.2** validation returns false, EIISERVER shall show an error message.

RConsultAcYear1. EIISERVER shall let a user authenticated as ADMINISTRATOR consult all Academic Year:

RConsultAcYear1.1. EIISERVER shall obtain all Academic Years from Database.

RConsultAcYear1.1.1. EIISERVER shall show the following fields for each Academic Year obtained:

RConsultAcYear1.1.1.1. Academic Degree Code.

RConsultAcYear1.1.1.2. Year.

5.2.1.7 COURSE

RCreateCourse1. EIISERVER shall let a user authenticated as ADMINISTRATOR create a Course:

RCreateCourse1.1. The next data will be requested:

RCreateCourse1.1.1. Course.

RCreateCourse1.1.1.1. Mandatory data.

RCreateCourse1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidCourse1**.

RCreateCourse1.1.2. If **RCreateCourse1.1.1.2** validation returns true, EIISERVER shall store the Course.

RCreateCourse1.1.2.1. EIISERVER shall store Course (**RCreateCourse1.1.1**) on the database.

RCreateCourse1.1.2.2. EIISERVER shall show a message indicating the creation succeed.

RCreateCourse1.1.3. If **RCreateCourse1.1.1.2** validation returns false, EIISERVER shall show an error message.

RDeleteCourse1. EIISERVER shall let a user authenticated as ADMINISTRATOR delete a Course:

RDeleteCourse1.1. The next data will be requested:

RDeleteCourse1.1.1. Course.

RDeleteCourse1.1.1.1. Mandatory data.

RDeleteCourse1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidCourse1**.

RDeleteCourse1.1.2. If **RDeleteCourse1.1.1.2** validation returns true, EIISERVER shall verify that Course is registered on the database.

RDeleteCourse1.1.2.1. If **RDeleteCourse1.1.2** is true, EIISERVER shall remove Course from the database.

RDeleteCourse1.1.2.2. If **RDeleteCourse1.1.2** is false, EIISERVER shall show an error message.

RDeleteCourse1.1.3. If **RDeleteCourse1.1.1.2** validation returns false, EIISERVER shall show an error message.

RConsultCourse1. EIISERVER shall let a user authenticated as ADMINISTRATOR consult all Courses:

RConsultCourse1.1. EIISERVER shall obtain all Courses from Database.

RConsultCourse1.1.1. EIISERVER shall show the following fields for each Course obtained:

RConsultCourse1.1.1.1. Academic Degree Code.

RConsultCourse1.1.1.2. Year.

RConsultCourse1.1.1.3. Course Code.

RConsultCourse1.1.1.4. Course Name.

RConsultCourse1.1.1.5. Course SIES Code.

5.2.1.8 GROUP

RCreateGroup1. EIISERVER shall let a user authenticated as ADMINISTRATOR create a Group:

RCreateGroup1.1. The next data will be requested:

RCreateGroup1.1.1. Group.

RCreateGroup1.1.1.1. Mandatory data.

RCreateGroup1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidGroup1**.

RCreateGroup1.1.2. If **RCreateGroup1.1.1.2** validation returns true, EIISERVER shall store the Group.

RCreateGroup1.1.2.1. EIISERVER shall store Group (**RCreateCourse1.1.1**) on the database.

RCreateGroup1.1.2.2. EIISERVER shall show a message indicating the creation succeed.

RCreateGroup1.1.3. If **RCreateGroup1.1.1.2** validation returns false, EIISERVER shall show an error message.

RDeleteGroup1. EIISERVER shall let a user authenticated as ADMINISTRATOR delete a Group:

RDeleteGroup1.1. The next data will be requested:

RDeleteGroup1.1.1. Group.

RDeleteGroup1.1.1.1. Mandatory data.

RDeleteGroup1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidGroup1**.

RDeleteGroup1.1.2. If **RDeleteGroup1.1.1.2** validation returns true, EIISERVER shall verify that Group is registered on the database.

RDeleteGroup1.1.2.1. If **RDeleteGroup1.1.2** is true, EIISERVER shall remove Group from the database.

RDeleteGroup1.1.2.2. If **RDeleteGroup1.1.2** is false, EIISERVER shall show an error message.

RDeleteGroup1.1.3. If **RDeleteGroup1.1.1.2** validation returns false, EIISERVER shall show an error message.

RConsultGroup1. EIISERVER shall let a user authenticated as ADMINISTRATOR consult all Groups:

RConsultGroup1.1. EIISERVER shall obtain all Groups from Database.

RConsultGroup1.1.1. EIISERVER shall show the following fields for each Group obtained:

RConsultGroup1.1.1.1. Academic Degree Code.

RConsultGroup1.1.1.2. Course Code.

RConsultGroup1.1.1.3. Group Code.

5.2.1.9 STUDENT

RCreateStudent1. EIISERVER shall let a user authenticated as ADMINISTRATOR create a Student:

RCreateStudent1.1. The next data will be requested:

RCreateStudent1.1.1. Student.

RCreateStudent1.1.1.1. Mandatory data.

RCreateStudent1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidStudent1**.

RCreateStudent1.1.2. If **RCreateStudent1.1.1.2** validation returns true, EIISERVER shall store the Student.

RCreateStudent1.1.2.1. EIISERVER shall store Student (**RCreateStudent1.1.1**) on the database.

RCreateStudent1.1.2.2. EIISERVER shall show a message indicating the creation succeed.

RCreateStudent1.1.3. If **RCreateStudent1.1.1.2** validation returns false, EIISERVER shall show an error message.

RDeleteStudent1. EIISERVER shall let a user authenticated as ADMINISTRATOR delete a Student:

RDeleteStudent1.1. The next data will be requested:

RDeleteStudent1.1.1. Student.

RDeleteStudent1.1.1.1. Mandatory data.

RDeleteStudent1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidStudent1**.

RDeleteStudent1.1.2. If **RDeleteStudent1.1.1.2** validation returns true, EIISERVER shall verify that Student is registered on the database.

RDeleteStudent1.1.2.1. If **RDeleteStudent1.1.2** is true, EIISERVER shall remove Student from the database.

RDeleteStudent1.1.2.2. If **RDeleteStudent1.1.2** is false, EIISERVER shall show an error message.

RDeleteStudent1.1.3. If **RDeleteStudent1.1.1.2** validation returns false, EIISERVER shall show an error message.

RConsultStudent1. EIISERVER shall let a user authenticated as ADMINISTRATOR consult all Students:

RConsultStudent1.1. EIISERVER shall obtain all Students from Database.

RConsultStudent1.1.1. EIISERVER shall show the following fields for each Student obtained:

RConsultStudent1.1.1.1. Student UO.

5.2.1.10 ATTENDANCE

RCreateAttendance1. EIISERVER shall let a user authenticated as ADMINISTRATOR create an Attendance:

RCreateAttendance1.1. The next data will be requested:

RCreateAttendance1.1.1. Attendance.

RCreateAttendance1.1.1.1. Mandatory data.

RCreateAttendance1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidAttendance1**.

RCreateAttendance1.1.2. If **RCreateAttendance1.1.1.2** validation returns true, EIISERVER shall store the Attendance.

RCreateAttendance1.1.2.1. EIISERVER shall store Attendance (**RCreateAttendance1.1.1**) on the database.

RCreateAttendance1.1.2.2. EIISERVER shall show a message indicating the creation succeed.

RCreateAttendance1.1.3. If **RCreateAttendance1.1.1.2** validation returns false, EIISERVER shall show an error message.

RDeleteAttendance1. EIISERVER shall let a user authenticated as ADMINISTRATOR delete an Attendance:

RDeleteAttendance1.1. The next data will be requested:

RDeleteAttendance1.1.1. Attendance.

RDeleteAttendance1.1.1.1. Mandatory data.

RDeleteAttendance1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidAttendance1**.

RDeleteAttendance1.1.2. If **RDeleteAttendance1.1.1.2** validation returns true, EIISERVER shall verify that Attendance is registered on the database.

RDeleteAttendance1.1.2.1. If **RDeleteAttendance1.1.2** is true, EIISERVER shall remove Attendance from the database.

RDeleteAttendance1.1.2.2. If **RDeleteAttendance1.1.2** is false, EIISERVER shall show an error message.

RDeleteAttendance1.1.3. If **RDeleteAttendance1.1.1.2** validation returns false, EIISERVER shall show an error message.

RConsultAttendance1. EIISERVER shall let a user authenticated as ADMINISTRATOR consult all Attendances:

RConsultAttendance1.1. EIISERVER shall obtain all Attendances from Database.

RConsultAttendance1.1.1. EIISERVER shall show the following fields for each Attendance obtained:

RConsultAttendance1.1.1.1. Course Code.

RConsultAttendance1.1.1.2. Academic Degree Code.

RConsultAttendance1.1.1.3. Group Code.

RConsultAttendance1.1.1.4. Student UO.

5.2.1.1 LECTURER

RCreateLecturer1. EIISERVER shall let a user authenticated as ADMINISTRATOR create a Lecturer:

RCreateLecturer1.1. The next data will be requested:

RCreateLecturer1.1.1. Lecturer.

RCreateLecturer1.1.1.1. Mandatory data.

RCreateLecturer1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidLecturer1**.

RCreateLecturer1.1.2. If **RCreateLecturer1.1.1.2** validation returns true, EIISERVER shall store the Lecturer.

RCreateLecturer1.1.2.1. EIISERVER shall store Lecturer (**RCreateLecturer1.1.1**) on the database.

RCreateLecturer1.1.2.2. EIISERVER shall show a message indicating the creation succeed.

RCreateLecturer1.1.3. If **RCreateLecturer1.1.1.2** validation returns false, EIISERVER shall show an error message.

RDeleteLecturer1. EIISERVER shall let a user authenticated as ADMINISTRATOR delete a Lecturer:

RDeleteLecturer1.1. The next data will be requested:

RDeleteLecturer1.1.1. Lecturer.

RDeleteLecturer1.1.1.1. Mandatory data.

RDeleteLecturer1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidLecturer1**.

RDeleteLecturer1.1.2. If **RDeleteLecturer1.1.1.2** validation returns true, EIISERVER shall verify that Lecturer is registered on the database.

RDeleteLecturer1.1.2.1. If **RDeleteLecturer1.1.2** is true, EIISERVER shall remove Lecturer from the database.

RDeleteLecturer1.1.2.2. If **RDeleteLecturer1.1.2** is false, EIISERVER shall show an error message.

RDeleteLecturer1.1.3. If **RDeleteLecturer1.1.1.2** validation returns false, EIISERVER shall show an error message.

RConsultLecturer1. EIISERVER shall let a user authenticated as ADMINISTRATOR consult all Lecturers:

RConsultLecturer1.1. EIISERVER shall obtain all Lecturers from Database.

RConsultLecturer1.1.1. EIISERVER shall show the following fields for each Lecturer obtained:

RConsultLecturer1.1.1.1. Lecturer Email.

RConsultLecturer1.1.1.2. Lecturer Name.

REditLecturer1. EIISERVER shall let a user authenticated as ADMINISTRATOR edit a Lecturer:

REditLecturer1.1. The next data will be requested:

REditLecturer1.1.1. Lecturer.

REditLecturer1.1.1.1. Mandatory data.

REditLecturer1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidLecturer1**.

REditLecturer1.2. If **REditLecturer1.1.1.2** validation returns true, EIISERVER shall verify that Lecturer is registered on the database.

REditLecturer1.2.1. If **REditLecturer1.2** is true, EIISERVER shall update the following fields of Lecturer on the database:

REditLecturer1.2.1.1. Name

REditLecturer1.2.2. If **REditLecturer1.2** is false, EIISERVER shall show an error message.

REditLecturer1.3. If **REditLecturer1.1.1.2** validation returns false, EIISERVER shall show an error message.

5.2.1.2 TEACH

RCreateTeach1. EIISERVER shall let a user authenticated as ADMINISTRATOR create a Teach:

RCreateTeach1.1. The next data will be requested:

RCreateTeach1.1.1. Teach.

RCreateTeach1.1.1.1. Mandatory data.

RCreateTeach1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidTeach1**.

RCreateTeach1.1.2. If **RCreateTeach1.1.1.2** validation returns true, EIISERVER shall store the Teach.

RCreateTeach1.1.2.1. EIISERVER shall store Teach (**RCreateTeach1.1.1**) on the database.

RCreateTeach1.1.2.2. EIISERVER shall show a message indicating the creation succeed.

RCreateTeach1.1.3. If **RCreateTeach1.1.1.2** validation returns false, EIISERVER shall show an error message.

RDeleteTeach1. EIISERVER shall let a user authenticated as ADMINISTRATOR delete a Teach:

RDeleteTeach1.1. The next data will be requested:

RDeleteTeach1.1.1. Teach.

RDeleteTeach1.1.1.1. Mandatory data.

RDeleteTeach1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidTeach1**.

RDeleteTeach1.1.2. If **RDeleteTeach1.1.1.2** validation returns true, EIISERVER shall verify that Teach is registered on the database.

RDeleteTeach1.1.2.1. If **RDeleteTeach1.1.2** is true, EIISERVER shall remove Teach from the database.

RDeleteTeach1.1.2.2. If **RDeleteTeach1.1.2** is false, EIISERVER shall show an error message.

RDeleteTeach1.1.3. If **RDeleteTeach1.1.1.2** validation returns false, EIISERVER shall show an error message.

RConsultTeach1. EIISERVER shall let a user authenticated as ADMINISTRATOR consult all Teaches:

RConsultTeach1.1. EIISERVER shall obtain all Teaches from Database.

RConsultTeach1.1.1. EIISERVER shall show the following fields for each Teach obtained:

RConsultTeach1.1.1.1. Course Code.

RConsultTeach1.1.1.2. Academic Degree Code.

RConsultTeach1.1.1.3. Group Code.

RConsultTeach1.1.1.4. Lecturer Email.

5.2.1.3 SESSION

RCreateSession1. EIISERVER shall let a user authenticated as ADMINISTRATOR create a Session:

RCreateSession1.1. The next data will be requested:

RCreateSession1.1.1. Session.

RCreateSession1.1.1.1. Mandatory data.

RCreateSession1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidSession1**.

RCreateSession1.1.2. If **RCreateSession1.1.1.2** validation returns true, EIISERVER shall store the Session.

RCreateSession1.1.2.1. EIISERVER shall store Session (**RCreateSession1.1.1**) on the database.

RCreateSession1.1.2.2. EIISERVER shall show a message indicating the creation succeed.

RCreateSession1.1.3. If **RCreateSession1.1.1.2** validation returns false, EIISERVER shall show an error message.

RDeleteSession1. EIISERVER shall let a user authenticated as ADMINISTRATOR delete a Session:

RDeleteSession1.1. The next data will be requested:

RDeleteSession1.1.1. Session.

RDeleteSession1.1.1.1. Mandatory data.

RDeleteSession1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidSession1**.

RDeleteSession1.1.2. If **RDeleteSession1.1.1.2** validation returns true, EIISERVER shall verify that Teach is registered on the database.

RDeleteSession1.1.2.1. If **RDeleteSession1.1.2** is true, EIISERVER shall remove Session from the database.

RDeleteSession1.1.2.2. If **RDeleteSession1.1.2** is false, EIISERVER shall show an error message.

RDeleteSession1.1.3. If **RDeleteSession1.1.1.2** validation returns false, EIISERVER shall show an error message.

RConsultSession1. EIISERVER shall let a user authenticated as ADMINISTRATOR consult all Sessions:

RConsultSession1.1. EIISERVER shall obtain all Sessions from Database.

RConsultSession1.1.1. EIISERVER shall show the following fields for each Session obtained:

RConsultSession1.1.1.1. Session Name.

RConsultSession1.1.1.2. Session Description.

RConsultSession1.1.1.3. Course Code.

RConsultSession1.1.1.4. Academic Degree Code.

RConsultSession1.1.1.5. Group Code.

RConsultSession1.1.1.6. Starting Date.

RConsultSession1.1.1.7. Ending Date.

RConsultSession1.1.1.8. Starting Time.

RConsultSession1.1.1.9. Ending Time.

RConsultSession1.1.1.10. Location.

REditSession1. EIISERVER shall let a user authenticated as ADMINISTRATOR edit a Session:

REditSession1.1. The next data will be requested:

REditSession1.1.1. Session.

REditSession1.1.1.1. Mandatory data.

REditSession1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidSession1**.

REditSession1.2. If **REditSession1.1.1.2** validation returns true, EIISERVER shall verify that Session is registered on the database.

REditSession1.2.1. If **REditSession1.2** is true, EIISERVER shall update the following fields of Session on the database:

REditSession1.2.1.1. Session Name.

REditSession1.2.1.2. Session Description.

REditSession1.2.1.3. Course Code.

REditSession1.2.1.4. Academic Degree Code.

REditSession1.2.1.5. Group Code.

REditSession1.2.1.6. Starting Date.

REditSession1.2.1.7. Ending Date.

REditSession1.2.1.8. Starting Time.

REditSession1.2.1.9. Ending Time.

REditSession1.2.1.10. Location.

REditSession1.2.2. If **REditSession1.2** is false, EIISERVER shall show an error message.

REditSession1.3. If **REditSession1.1.1.2** validation returns false, EIISERVER shall show an error message.

5.2.1.4 EXAM

RCreateExam1. EIISERVER shall let a user authenticated as ADMINISTRATOR create an Exam:

RCreateExam1.1. The next data will be requested:

RCreateExam1.1.1. Exam.

RCreateExam1.1.1.1. Mandatory data.

RCreateExam1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidExam1**.

RCreateExam1.1.2. If **RCreateExam1.1.1.2** validation returns true, EIISERVER shall store the Exam.

RCreateExam1.1.2.1. EIISERVER shall store Exam (**RCreateExam1.1.1**) on the database.

RCreateExam1.1.2.2. EIISERVER shall show a message indicating the creation succeed.

RCreateExam1.1.3. If **RCreateExam1.1.1.2** validation returns false, EIISERVER shall show an error message.

RDeleteExam1. EIISERVER shall let a user authenticated as ADMINISTRATOR delete an Exam:

RDeleteExam1.1. The next data will be requested:

RDeleteExam1.1.1. Exam.

RDeleteExam1.1.1.1. Mandatory data.

RDeleteExam1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidExam1**.

RDeleteExam1.1.2. If **RDeleteExam1.1.1.2** validation returns true, EIISERVER shall verify that Exam is registered on the database.

RDeleteExam1.1.2.1. If **RDeleteExam1.1.2** is true, EIISERVER shall remove Exam from the database.

RDeleteExam1.1.2.2. If **RDeleteExam1.1.2** is false, EIISERVER shall show an error message.

RDeleteExam1.1.3. If **RDeleteExam1.1.1.2** validation returns false, EIISERVER shall show an error message.

RConsultExam1. EIISERVER shall let a user authenticated as ADMINISTRATOR consult all Exam:

RConsultExam1.1. EIISERVER shall obtain all Exams from Database.

RConsultExam1.1.1. EIISERVER shall show the following fields for each Exam obtained:

RConsultExam1.1.1.1. Exam Name.

RConsultExam1.1.1.2. Exam Description.

RConsultExam1.1.1.3. Course Code.

RConsultExam1.1.1.4. Academic Degree Code.

RConsultExam1.1.1.5. Starting Date.

RConsultExam1.1.1.6. Ending Date.

RConsultExam1.1.1.7. Starting Time.

RConsultExam1.1.1.8. Ending Time.

RConsultExam1.1.1.9. Location.

REditExam1. EIISERVER shall let a user authenticated as ADMINISTRATOR edit an Exam:

REditExam1.1. The next data will be requested:

REditExam1.1.1. Exam.

REditExam1.1.1.1. Mandatory data.

REditExam1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidExam1**.

REditExam1.2. If **REditExam1.1.1.2** validation returns true, EIISERVER shall verify that Exam is registered on the database.

REditExam1.2.1. If **REditExam1.2** is true, EIISERVER shall update the following fields of Exam on the database:

REditExam1.2.1.1. Session Name.

REditExam1.2.1.2. Session Description.

REditExam1.2.1.3. Course Code.

REditExam1.2.1.4. Academic Degree Code.

REditExam1.2.1.5. Starting Date.

REditExam1.2.1.6. Ending Date.

REditExam1.2.1.7. Starting Time.

REditExam1.2.1.8. Ending Time.

REditExam1.2.1.9. Location.

REditExam1.2.2. If **REditExam1.2** is false, EIISERVER shall show an error message.

REditExam1.3. If **REditExam1.1.1.2** validation returns false, EIISERVER shall show an error message.

5.2.1.5 NON-WORKING DAY

RCreateNonWDay1. EIISERVER shall let a user authenticated as ADMINISTRATOR create a Non-Working Day:

RCreateNonWDay1.1. The next data will be requested:

RCreateNonWDay1.1.1. Non-Working Day.

RCreateNonWDay1.1.1.1. Mandatory data.

RCreateNonWDay1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidNonWDay1**.

RCreateNonWDay1.1.2. If **RCreateNonWDay1.1.1.2** validation returns true, EIISERVER shall store the Non-Working Day.

RCreateNonWDay1.1.2.1. EIISERVER shall store Non-Working Day (**RCreateNonWDay1.1.1**) on the database.

RCreateNonWDay1.1.2.2. EIISERVER shall show a message indicating the creation succeed.

RCreateNonWDay1.1.3. If **RCreateNonWDay1.1.1.2** validation returns false, EIISERVER shall show an error message.

RDeleteNonWDay1. EIISERVER shall let a user authenticated as ADMINISTRATOR delete a Non-Working Day:

RDeleteNonWDay1.1. The next data will be requested:

RDeleteNonWDay1.1.1. Non-Working Day.

RDeleteNonWDay1.1.1.1. Mandatory data.

RDeleteNonWDay1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidNonWDay1**.

RDeleteNonWDay1.1.2. If **RDeleteNonWDay1.1.1.2** validation returns true, EIISERVER shall verify that Non-Working Day is registered on the database.

RDeleteNonWDay1.1.2.1. If **RDeleteNonWDay1.1.2** is true, EIISERVER shall remove Non-Working Day from the database.

RDeleteNonWDay1.1.2.2. If **RDeleteNonWDay1.1.2** is false, EIISERVER shall show an error message.

RDeleteNonWDay1.1.3. If **RDeleteNonWDay1.1.1.2** validation returns false, EIISERVER shall show an error message.

RConsultNonWDay1. EIISERVER shall let a user authenticated as ADMINISTRATOR consult all Non-Working Days:

RConsultNonWDay1.1. EIISERVER shall obtain all Non-Working Days from Database.

RConsultNonWDay1.1.1. EIISERVER shall show the following fields for each Non-Working Day obtained:

RConsultNonWDay1.1.1.1. Day.

5.2.1.6 AUTHORIZED USER

RCreateAuthUser1. EIISERVER shall let a user authenticated as ADMINISTRATOR create an Authorized User:

RCreateAuthUser1.1. The next data will be requested:

RCreateAuthUser1.1.1. Authorized User.

RCreateAuthUser1.1.1.1. Mandatory data.

RCreateAuthUser1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidExam1**.

RCreateAuthUser1.1.2. If **RCreateAuthUser1.1.1.2** validation returns true, EIISERVER shall store the Authorized User.

RCreateAuthUser1.1.2.1. EIISERVER shall store Authorized User (**RCreateAuthUser1.1.1**) on the database.

RCreateAuthUser1.1.2.2. EIISERVER shall show a message indicating the creation succeed.

RCreateAuthUser1.1.3. If **RCreateAuthUser1.1.1.2** validation returns false, EIISERVER shall show an error message.

RDeleteAuthUser1. EIISERVER shall let a user authenticated as ADMINISTRATOR delete an Authorized User:

RDeleteAuthUser1.1. The next data will be requested:

RDeleteAuthUser1.1.1. Authorized User.

RDeleteAuthUser1.1.1.1. Mandatory data.

RDeleteAuthUser1.1.1.2. EIISERVER shall verify that is valid according to **RSERVValidAuthUser1**.

RDeleteAuthUser1.1.2. If **RDeleteAuthUser1.1.1.2** validation returns true, EIISERVER shall verify that Authorized User is registered on the database.

RDeleteAuthUser1.1.2.1. If **RDeleteAuthUser1.1.2** is true, EIISERVER shall remove Authorized User from the database.

RDeleteAuthUser1.1.2.2. If **RDeleteAuthUser1.1.2** is false, EIISERVER shall show an error message.

RDeleteAuthUser1.1.3. If **RDeleteAuthUser1.1.1.2** validation returns false, EIISERVER shall show an error message.

RConsultAuthUser1. EIISERVER shall let a user authenticated as ADMINISTRATOR consult all Authorized Users:

RConsultAuthUser1.1. EIISERVER shall obtain all Authorized Users from Database.

RConsultAuthUser1.1.1. EIISERVER shall show the following fields for each Authorized User obtained:

RConsultAuthUser1.1.1.1. Username.

RConsultAuthUser1.1.1.2. Role.

5.2.1.7 LINKS OF INTEREST

REditLinks1. EIISERVER shall let a user authenticated as ADMINISTRATOR edit the Links of Interest.

REditLinks1.1. The next data will be requested:

REditLinks1.1.1. Content

REditLinks1.1.1.1. Mandatory data

REditLinks1.2. EIISERVER shall store Content (**REditLinks1.1.1**) on the database.

RConsultLinks1. Consult the Links of Interest Page.

RConsultLinks1.1. EIISERVER shall obtain the Links of Interest Page from database.

RConsultLinks1.1.1. EIISERVER shall show the following fields of the obtained Links of Interest Page:

RConsultLinks1.1.1.1. Content.

5.2.1.8 UPDATES

RUpdate1. Update the information stored on the database.

RUpdate1.1. The next data will be requested:

RUpdate1.1.1. Academic Degrees.

RUpdate1.1.1.1. Mandatory data.

RUpdate1.1.1.2. EIISERVER shall verify that Academic Degrees' length is greater than zero.

RUpdate1.1.1.3. EIISERVER shall verify that each Academic Degree is valid according to **RSERVValidAcDegree1**.

RUpdate1.1.2. If **RUpdate1.1.1.2** and **RUpdate1.1.1.3** validation returns true, EIISERVER shall update the database:

RUpdate1.1.2: EIISERVER shall communicate with the Official School of Computer Science Endpoints of the selected Academic Degrees (**RUpdate1.1.1.**):

RUpdate1.1.2.1: EIISERVER shall collect the answer of the Official School of Computer Science Endpoints:

RUpdate1.1.2.1.1: If collected response is ERROR, EIISERVER shall show an error message.

RUpdate1.1.2.1.2: If collected response is OK, EIISERVER shall store the information of the response on the database.

RUpdate1.1.3. If **RUpdate1.1.1.2** or **RUpdate1.1.1.3** validation returns false, EIISERVER shall show an error message.

RSchedule1. Schedule an update.

RSchedule1.1. The next data will be requested:

RSchedule1.1.1. Periodicity.

RSchedule1.1.1.1. Mandatory data.

RSchedule1.1.1.2. EIISERVER shall verify that it is one of the following values:

RSchedule1.1.1.2.1. DAILY.
RSchedule1.1.1.2.1. WEEKLY.
RSchedule1.1.1.2.1. NEVER.

RSchedule1.1.2. Daily Time.

RSchedule1.1.2.1. Mandatory data.

RSchedule1.1.2.2. EIISERVER shall verify that it is valid according to **RSERVValidTime1**.

RSchedule1.1.3. Weekly Time.

RSchedule1.1.3.1. Mandatory data.

RSchedule1.1.3.2. EIISERVER shall verify that it is valid according to **RSERVValidTime1**.

RSchedule1.1.4. Weekly Day.

RSchedule1.1.4.1. Mandatory data.

RSchedule1.1.4.2. EIISERVER shall verify that it is one of the following values:

RSchedule1.1.4.2.1. MONDAY.
RSchedule1.1.4.2.2. TUESDAY.
RSchedule1.1.4.2.3. WEDNESDAY.
RSchedule1.1.4.2.4. THURSDAY.
RSchedule1.1.4.2.5. FRIDAY.
RSchedule1.1.4.2.6. SATURDAY.
RSchedule1.1.4.2.7. SUNDAY.

RSchedule1.1.2. If **RSchedule1.1.1.2, RSchedule1.1.2.2, RSchedule1.1.3.2** and **RSchedule1.1.4.2** validation returns true, EIISERVER shall schedule the database:

RSchedule1.1.2.1. EIISERVER shall update the database (see **RSchedule1.1.2**) when selected Periodicity (**RSchedule1.1.1.**) matches current time:

RSchedule1.1.2.1.1. If Periodicity is DAILY, when current time matches Daily Time (**RSchedule1.1.2**).

RSchedule1.1.2.1.2. If Periodicity is WEEKLY, when current time matches Weekly Time (**RSchedule1.1.3**) and current day matches Weekly Day (**RSchedule1.1.4.2**).

RSchedule1.1.2.1.3. If Periodicity is NEVER, never is updated.

RSchedule1.1.3. If **RSchedule1.1.1.2, RSchedule1.1.2.2, RSchedule1.1.3.2** or **RSchedule1.1.4.2** validation returns false, EIISERVER shall show an error message.

5.2.1.9 VALIDATION

5.2.1.9.1 NOTIFICATION

RSERVValidTitle1. EIISERVER shall verify that title (**RSERVNotif1.1.1.**) text string length is greater than zero.

RSERVValidTitle1.1. If **RSERVValidTitle1** is true, EIISERVER shall verify that title (**RSERVNotif1.1.1.**) text string is lesser than TITLE_MAX_CHARACTERS.

RSERVValidTitle1.1.1. The initial value of TITLE_MAX_CHARACTERS is 200.

RSERVValidTitle1.1.2. The value of TITLE_MAX_CHARACTERS shall be able to be modified by the administrator.

RSERVValidTitle1.1.3. If **RSERVValidTitle1.1** is true, then title (**RSERVNotif1.1.1.**) is valid.

RSERVValidTitle1.1.4. If **RSERVValidTitle1.1** is false, then title (**RSERVNotif1.1.1.**) is not valid.

RSERVValidTitle1.2. If RSERVValidTitle1 is false, then title (**RSERVNotif1.1.1.**) is not valid.

RSERVValidContent1. EIISERVER shall verify that content (**RSERVNotif1.1.2.**) text string length is greater than zero.

RSERVValidContent1.1. If **RSERVValidContent1** is true, EIISERVER shall verify that content (**RSERVNotif1.1.2.**) text string is lesser than CONTENT_MAX_CHARACTERS.

RSERVValidContent1.1.1. The initial value of CONTENT_MAX_CHARACTERS is 1024.

RSERVValidContent1.1.2. The value of CONTENT_MAX_CHARACTERS shall be able to be modified by the administrator.

RSERVValidContent1.1.3. If **RSERVValidContent1.1** is true, then content (**RSERVNotif1.1.2.**) is valid.

RSERVValidContent1.1.4. If **RSERVValidContent1.1** is false, then content (**RSERVNotif1.1.2.**) is not valid.

RSERVValidTitle1.2. If RSERVValidContent1 is false, then content (**RSERVNotif1.1.2.**) is not valid.

RSERVValidRecipients1. EIISERVER shall verify that recipients are valid.

RSERVValidRecipients1.1. EIISERVER shall verify that recipients' length is greater than zero.

RSERVValidRecipients1.1.1. If **RSERVValidRecipients1.1** is true, then EIISERVER shall verify that each recipient is valid.

RSERVValidRecipients1.1.1.1. If recipients are ACADEMIC DEGREES, then EIISERVER shall verify that each recipient is valid according to **RSERVValidAcDegree1**.

RSERVValidRecipients1.1.1.1.2. If **RSERVValidRecipients1.1.1.1** validation returns true, then EIISERVER shall verify that each recipient is registered on the database.

RSERVValidRecipients1.1.1.1.2.1. If **RSERVValidRecipients1.1.1.1.2** is true, then recipients are valid.

RSERVValidRecipients1.1.1.1.2.2. If **RSERVValidRecipients1.1.1.1.2** is false, then recipients are not valid.

RSERVValidRecipients1.1.1.1.3. If **RSERVValidRecipients1.1.1.1** validation returns false, then recipients are not valid.

RSERVValidRecipients1.1.1.2. If recipients are ACADEMIC YEARS, then EIISERVER shall verify that each recipient is valid according to **RSERVValidAcYear1**.

RSERVValidRecipients1.1.1.2.1. If **RSERVValidRecipients1.1.1.2** validation returns true, then EIISERVER shall verify that each recipient is registered on the database.

RSERVValidRecipients1.1.1.2.1.1. If **RSERVValidRecipients1.1.1.2.1** is true, then recipients are valid.

RSERVValidRecipients1.1.1.2.1.2. If **RSERVValidRecipients1.1.1.2.1** is false, then recipients are not valid.

RSERVValidRecipients1.1.1.2.2. If **RSERVValidRecipients1.1.1.2** validation returns false, then recipients are not valid.

RSERVValidRecipients1.1.1.3. If recipients are COURSES, then EIISERVER shall verify that each recipient is valid according to **RSERVValidCourse1**.

RSERVValidRecipients1.1.1.3.1. If **RSERVValidRecipients1.1.1.3** validation returns true, then EIISERVER shall verify that each recipient is registered on the database.

RSERVValidRecipients1.1.1.3.1.1. If **RSERVValidRecipients1.1.1.3.1** is true, then recipients are valid.

RSERVValidRecipients1.1.1.3.1.2. If **RSERVValidRecipients1.1.1.3.1** is false, then recipients are not valid.

RSERVValidRecipients1.1.1.3.2. If **RSERVValidRecipients1.1.1.3** validation returns false, then recipients are not valid.

RSERVValidRecipients1.1.1.4. If recipients are GROUPS, then EIISERVER shall verify that each recipient is valid according to **RSERVValidGroup1**.

RSERVValidRecipients1.1.1.4.1. If **RSERVValidRecipients1.1.1.4** validation returns true, then EIISERVER shall verify that each recipient is registered on the database.

RSERVValidRecipients1.1.1.4.1.1. If **RSERVValidRecipients1.1.1.4.1** is true, then recipients are valid.

RSERVValidRecipients1.1.1.4.1.2. If **RSERVValidRecipients1.1.1.4.1** is false, then recipients are not valid.

RSERVValidRecipients1.1.1.4.2. If **RSERVValidRecipients1.1.1.4** validation returns false, then recipients are not valid.

RSERVValidRecipients1.1.1.5. If recipients are STUDENTS, then EIISERVER shall verify that each recipient is valid according to **RSERVValidStudent1**.

RSERVValidRecipients1.1.1.5.1. If **RSERVValidRecipients1.1.1.5** validation returns true, then EIISERVER shall verify that each recipient is registered on the database.

RSERVValidRecipients1.1.1.5.1.1. If **RSERVValidRecipients1.1.1.5.1** is true, then recipients are valid.

RSERVValidRecipients1.1.1.5.1.2. If **RSERVValidRecipients1.1.1.5.1** is false, then recipients are not valid.

RSERVValidRecipients1.1.1.5.2. If **RSERVValidRecipients1.1.1.5** validation returns false, then recipients are not valid.

RSERVValidRecipients1.1.1.6. If recipients are LECTURERS, then EIISERVER shall verify that each recipient is valid according to **RSERVValidLecturer1**.

RSERVValidRecipients1.1.1.6.1. If **RSERVValidRecipients1.1.1.6** validation returns true, then EIISERVER shall verify that each recipient is registered on the database.

RSERVValidRecipients1.1.1.6.1.1. If **RSERVValidRecipients1.1.1.6.1** is true, then recipients are valid.

RSERVValidRecipients1.1.1.6.1.2. If **RSERVValidRecipients1.1.1.6.1** is false, then recipients are not valid.

RSERVValidRecipients1.1.1.6.2. If **RSERVValidRecipients1.1.1.6** validation returns false, then recipients are not valid.

RSERVValidRecipients1.1.2. If **RSERVValidRecipients1.1** is false, then recipients are not valid.

5.2.1.9.2 ACADEMIC DEGREE

RSERVValidAcDegree1. EIISERVER shall verify that Academic Degree is valid.

RSERVValidAcDegree1.1. EIISERVER shall verify that Academic Degree Code text string length is greater than zero.

RSERVValidAcDegree1.1.1. If **RSERVValidAcDegree1.1** is true, Academic Degree is valid.

RSERVValidAcDegree1.1.2. If **RSERVValidAcDegree1.1** is false, Academic Degree is not valid.

5.2.1.9.3 ACADEMIC YEAR

RSERVValidAcYear1. EIISERVER shall verify that Academic Year is valid.

RSERVValidAcYear1.1. EIISERVER shall verify that Academic Degree Code text string length is greater than zero.

RSERVValidAcYear1.2. EIISERVER shall verify that Year is numeric.

RSERVValidAcYear1.2.1. If **RSERVValidAcYear1.1** and **RSERVValidAcYear1.2** is true, Academic Year is valid.

RSERVValidAcYear1.2.2. If **RSERVValidAcYear1.1** or **RSERVValidAcYear1.2** is false, Academic Year is not valid.

5.2.1.9.4 COURSE

RSERVValidCourse1. EIISERVER shall verify that Course is valid.

RSERVValidCourse1.1. EIISERVER shall verify that Course Code text string length is greater than zero.

RSERVValidCourse1.2. EIISERVER shall verify that Academic Degree is valid according to **RSERVValidAcDegree1**.

RSERVValidCourse1.3. EIISERVER shall verify that SIES Code text string length is greater than zero.

RSERVValidCourse1.4. EIISERVER shall verify that Name text string length is greater than zero.

RSERVValidCourse1.5. EIISERVER shall verify that Year is numeric.

RSERVValidCourse1.5.1. If **RSERVValidCourse1.1**, **RSERVValidCourse1.2**, **RSERVValidCourse1.3**, **RSERVValidCourse1.4** and **RSERVValidCourse1.5** is true, Course is valid.

RSERVValidCourse1.5.2. If **RSERVValidCourse1.1**, **RSERVValidCourse1.2**, **RSERVValidCourse1.3**, **RSERVValidCourse1.4** or **RSERVValidCourse1.5** is false, Course is not valid.

5.2.1.9.5 GROUP

RSERVValidGroup1. EIISERVER shall verify that Group is valid.

RSERVValidGroup1.1. EIISERVER shall verify that Course is valid according to **RSERVValidCourse1**.

RSERVValidGroup1.2. EIISERVER shall verify that Group Code text string length is greater than zero.

RSERVValidGroup1.2.1. If **RSERVValidGroup1.1** and **RSERVValidGroup1.2** is true, Group is valid.

RSERVValidGroup1.2.2. If **RSERVValidGroup1.1** or **RSERVValidGroup1.2** is false, Group is not valid.

5.2.1.9.6 STUDENT

RSERVValidStudent1. EIISERVER shall verify that Student is valid.

RSERVValidStudent1.1. EIISERVER shall verify that Student UO text string length is greater than zero.

RSERVValidStudent1.1.1. If **RSERVValidStudent1.1** is true, EIISERVER shall verify that Student UO starts with the one of the following characters followed by one or more numbers:

RSERVValidStudent1.1.1.1. "UO".

RSERVValidStudent1.1.1.2. "uo".

RSERVValidStudent1.1.1.3. If **RSERVValidStudent1.1.1** is false, Student is not valid.

RSERVValidStudent1.1.1.4. If **RSERVValidStudent1.1.1** is true, Student is valid.

RSERVValidStudent1.1.2. If **RSERVValidStudent1.1** is false, Student is not valid.

5.2.1.9.7 ATTENDANCE

RSERVValidAttendance1. EIISERVER shall verify that Attendance is valid.

RSERVValidAttendance1.1. EIISERVER shall verify that Course is valid according to **RSERVValidCourse1**.

RSERVValidAttendance1.2. EIISERVER shall verify that Group is valid according to **RSERVValidGroup1**.

RSERVValidAttendance1.3. EIISERVER shall verify that Student is valid according to **RSERVValidStudent1**.

RSERVValidAttendance1.3.1. If **RSERVValidAttendance1.1**, **RSERVValidAttendance1.2** and **RSERVValidAttendance1.3** is true, Attendance is valid.

RSERVValidGroup1.2.2. If **RSERVValidGroup1.1** or **RSERVValidGroup1.2** is false, Attendance is not valid.

5.2.1.9.8 LECTURER

RSERVValidLecturer1. EIISERVER shall verify that Lecturer is valid.

RSERVValidLecturer1.1. EIISERVER shall verify that Email text string length is greater than zero.

RSERVValidLecturer1.2. EIISERVER shall verify that Name text string length is greater than zero.

RSERVValidLecturer1.2.1. If **RSERVValidLecturer1.1**, and **RSERVValidLecturer1.2** is true, Lecturer is valid.

RSERVValidGroup1.2.2. If **RSERVValidLecturer1.1**, or **RSERVValidLecturer1.2** is false, Lecturer is not valid.

5.2.1.9.9 TEACH

RSERVValidTeach1. EIISERVER shall verify that Teach is valid.

RSERVValidTeach1.1. EIISERVER shall verify that Course is valid according to **RSERVValidCourse1**.

RSERVValidTeach1.2. EIISERVER shall verify that Group is valid according to **RSERVValidGroup1**.

RSERVValidTeach1.3. EIISERVER shall verify that Lecturer is valid according to **RSERVValidLecturer1**.

RSERVValidTeach1.3.1. If **RSERVValidTeach1.1**, **RSERVValidTeach1.2** and **RSERVValidTeach1.3** is true, Teach is valid.

RSERVValidTeach1.2.2. If **RSERVValidTeach1.1** or **RSERVValidTeach1.2** is false, Teach is not valid.

5.2.1.9.10 SESSION

RSERVValidSession1. EIISERVER shall verify that Session is valid.

RSERVValidSession1.1. EIISERVER shall verify that Name text string length is greater than zero.

RSERVValidSession1.2. EIISERVER shall verify that Description text string length is greater than zero.

RSERVValidSession1.3. EIISERVER shall verify that Course is valid according to **RSERVValidCourse1**.

RSERVValidSession1.4. EIISERVER shall verify that Group is valid according to **RSERVValidGroup1**.

RSERVValidSession1.5. EIISERVER shall verify that Starting Date is valid according to **RSERVValidDate1**.

RSERVValidSession1.6. EIISERVER shall verify that Ending Date is valid according to **RSERVValidDate1**.

RSERVValidSession1.7. EIISERVER shall verify that Starting Time is valid according to **RSERVValidTime1**.

RSERVValidSession1.8. EIISERVER shall verify that Ending Time is valid according to **RSERVValidTime1**.

RSERVValidSession1.9. EIISERVER shall verify that Location text string length is greater than zero.

RSERVValidSession1.9.1. If **RSERVValidSession1.1**, **RSERVValidSession1.2**, **RSERVValidSession1.3**, **RSERVValidSession1.4**, **RSERVValidSession1.5**, **RSERVValidSession1.6**, **RSERVValidSession1.7**, **RSERVValidSession1.8** and **RSERVValidSession1.9** is true, Session is valid.

RSERVValidSession1.9.2. If **RSERVValidSession1.1**, **RSERVValidSession1.2**, **RSERVValidSession1.3**, **RSERVValidSession1.4**, **RSERVValidSession1.5**, **RSERVValidSession1.6**, **RSERVValidSession1.7**, **RSERVValidSession1.8** or **RSERVValidSession1.9** is false, Session is not valid.

5.2.1.9.11 EXAM

RSERVValidExam1. EIISERVER shall verify that Exam is valid.

RSERVValidExam1.1. EIISERVER shall verify that Name text string length is greater than zero.

RSERVValidExam1.2. EIISERVER shall verify that Description text string length is greater than zero.

RSERVValidSession1.3. EIISERVER shall verify that Course is valid according to **RSERVValidCourse1**.

RSERVValidSession1.4. EIISERVER shall verify that Starting Date is valid according to **RSERVValidDate1**.

RSERVValidSession1.5. EIISERVER shall verify that Ending Date is valid according to **RSERVValidDate1**.

RSERVValidSession1.6. EIISERVER shall verify that Starting Time is valid according to **RSERVValidTime1**.

RSERVValidSession1.7. EIISERVER shall verify that Ending Time is valid according to **RSERVValidTime1**.

RSERVValidSession1.8. EIISERVER shall verify that Location text string length is greater than zero.

RSERVValidExam1.8.1. If **RSERVValidExam1.1**, **RSERVValidExam1.2**, **RSERVValidExam1.3**, **RSERVValidExam1.4**, **RSERVValidExam1.5**, **RSERVValidExam1.6**, **RSERVValidExam1.7**, and **RSERVValidExam1.8** is true, Exam is valid.

RSERVValidExam1.8.2. If **RSERVValidExam1.1**, **RSERVValidExam1.2**, **RSERVValidExam1.3**, **RSERVValidExam1.4**, **RSERVValidExam1.5**, **RSERVValidExam1.6**, **RSERVValidExam1.7**, or **RSERVValidExam1.8** is false, Exam is not valid.

5.2.1.9.12 NON-WORKING DAY

RSERVValidNonWDay1. EIISERVER shall verify that Non-Working Day is valid.

RSERVValidWDay1.1. EIISERVER shall verify that Day is valid according to **RSERVValidDate1**.

RSERVValidWDay1.1.1. If **RSERVValidWDay1.1** is true, then Non-Working Day is valid.

RSERVValidWDay1.1.2. If **RSERVValidWDay1.1** is false, then Non-Working Day is not valid.

5.2.1.9.13 AUTHORIZED USER

RSERVValidAuthUser1. EIISERVER shall verify that Authorized User is valid.

RSERVValidAuthUser1.1. EIISERVER shall verify that Username string text length is greater than zero.

RSERVValidAuthUser1.2. EIISERVER shall verify that ROLE is valid according to **RSERVValidRole1**.

RSERVValidAuthUser1.2.1. If **RSERVValidAuthUser1.1** and **RSERVValidAuthUser1.2** is true, then Authorized User is valid.

RSERVValidAuthUser1.2.2. If **RSERVValidAuthUser1.1** or **RSERVValidAuthUser1.2** is false, then Authorized User is not valid.

5.2.1.9.14 DATE

RSERVValidDate1. EIISERVER shall verify that Date is valid.

RSERVValidDate1.1. EIISERVER shall verify that Date follows “dd/mm/yyyy” pattern.

RSERVValidDate1.1.1. If **RSERVValidDate1.1** validation returns true, then Date is valid.

RSERVValidDate1.1.2. If **RSERVValidDate1.1** validation returns false, then Date is not valid.

5.2.1.9.15 TIME

RSERVValidTime1. EIISERVER shall verify that Time is valid.

RSERVValidTime1.1. EIISERVER shall verify that Time follows “hh:mm” pattern.

RSERVValidTime1.1.1. If **RSERVValidTime1.1** validation returns true, then Time is valid.

RSERVValidTime1.1.2. If **RSERVValidTime1.1** validation returns false, then Time is not valid.

5.2.1.9.16 ROLE

RSERVValidRole1. EIISERVER shall verify that Role is valid.

RSERVValidRole1.1. EIISERVER shall verify that Role is one of the following:

RSERVValidRole1.1.1. ADMINISTRATOR.

RSERVValidRole1.1.1. NOTIFIER.

RSERVValidRole1.1.3. If **RSERVValidRole1.1** validation returns true, then Role is valid.

RSERVValidRole1.1.4. If **RSERVValidRole1.1** validation returns false, then Role is not valid.

5.2.2 EIIAPP REQUIREMENTS

5.2.2.1 AUTHENTICATION

RAPPAuth1. EIIAPP shall let a non-authenticated user to authenticate.

RAPPAuth1.1. The next data will be requested for authentication:

RAPPAuth1.1.1. An identifier: UO.

RAPPAuth1.1.1.1. Mandatory data.

RAPPAuth1.1.1.2. EIIAPP shall verify that is valid according to **RAPPValidUO1**.

RAPPAuth1.2. If **RAPPAuth1.1.1.2** validation returns true, then EIIAPP shall communicate with the EIISERVER to authenticate the user:

RAPPAuth1.2.1. EIIAPP shall send the identifier (**RAPPAuth1.1.1**) to EIISERVER.

RAPPAuth1.2.1.2. EIIAPP shall collect the response EIISERVER:

RAPPAuth1.2.1.2.1. If collected response is OK, EIIAPP shall authenticate the user:

RAPPAuth1.2.1.2.1.1. The user will be authenticated (see **RAPP1**).

RAPPAuth1.2.1.2.1.2. EIIAPP shall update the information stored on the database (see **RAPPUpdate1**).

RSERVAuth1.2.1.2.2. If collected response is ERROR, EIIAPP shall show an error message (the user will be kept non-authenticated).

RAPPAuth1.3. If **RAPPAuth1.1.1.2** validation returns false, EIIAPP shall show an error message (the user will be kept non-authenticated).

5.2.2.2 AUTHENTICATED USER

RAPP1. An authenticated user on EIIAPP shall be able to do the following actions:

RAPP1.1. Consult Links of Interest (see **RAPPLink1**).

RAPP1.2. Consult Events (see **RAPPEvent1**).

RAPP1.3. Export Events (see **RAPPExport1**).

RAPP1.4. Report a breakdown (see **RAPPReport1**).

RAPP1.5. Consult Notifications (see **RAPPNotif1**).

RAPP1.6. Update information stored on the database (see **RAPPUpdate1**).

RAPP1.7. Schedule Updates (see **RAPPSchedule1**).

5.2.2.2.1 CONSULT LINKS OF INTEREST

RAPPLink1. EIIAPP shall let an authenticated user consult Links of Interest.

RAPPLink1.1. EIIAPP shall obtain Links of Interest from database.

RAPPLink1.1.1. EIIAPP shall show the following data for the Links of Interest obtained from database:

RAPPLink1.1.1.1. Content.

5.2.2.2.2 CONSULT EVENTS

RAPPEvent1. EIIAPP shall let an authenticated user consult Events.

RAPPEvent1.1. EIIAPP shall obtain Events from database.

RAPPEvent1.1.1. EIIAPP shall show the following data for each Event obtained from database:

RAPPEvent1.1.1.1. Name.

RAPPEvent1.1.1.2. Description.

RAPPEvent1.1.1.3. Starting Date.

RAPPEvent1.1.1.4. Ending Date.

RAPPEvent1.1.1.5. Starting Time.

RAPPEvent1.1.1.6. Ending Time.

RAPPEvent1.1.1.7. Location.

5.2.2.2.3 EXPORT EVENTS

RAPPExport1. EIIAPP shall let an authenticated user export Events to Google Calendar.

RAPPExport1.1. If user is authenticated with Google, EIIAPP shall communicate with Google Calendar API to obtain the calendars of the authenticated user:

RAPPExport1.1.1. EIIAPP shall send the identifier (**RAPPExport1.2.1.1**) and password (**RAPPExport1.2.1.2**) to Google Calendar API.

RAPPExport1.1.2. EIIAPP shall collect the response from Google Calendar API:

RAPPExport1.1.2.1. If collected response is OK, EIIAPP shall show received calendars on the response to authenticated user:

RAPPExport1.1.2.1.1. EIIAPP shall show the following data for each calendar received:

RAPPExport1.1.2.1.1.1. Calendar Name.

RAPPExport1.1.2.1.2. EIIAPP shall let a user export Events to a Calendar:

RAPPExport1.1.2.1.2.1. The next data will be requested:

RAPPExport1.1.2.1.2.1.1. A Calendar Name.

RAPPExport1.1.2.1.2.1.1. Mandatory data.

RAPPExport1.1.2.1.2.2. EIIAPP shall obtain Events stored on database.

RAPPExport1.1.2.1.2.2.1. EIIAPP shall communicate with Google Calendar API to export the events:

RAPPExport1.1.2.1.2.2.1.1. EIIAPP shall pass Calendar Name (**RAPPExport1.1.2.1.2.1.1**) and Events (**RAPPExport1.1.2.1.2.2**) to Google Calendar API.

RAPPExport1.1.2.1.2.2.1.2. EIIAPP shall collect the response from Google Calendar API:

RAPPExport1.1.2.1.2.2.1.2.1. If collected response is OK, EIIAPP shall show a message indicating the exportation succeeded.

RAPPExport1.1.2.1.2.2.1.2.2. If collected response is ERROR, EIIAPP shall show an error message.

RAPPExport1.1.2.2. If collected response is ERROR, EIIAPP shall show an error message.

RAPPExport1.2. If user is not authenticated with Google, EIIAPP shall let the user authenticate with Google:

RAPPExport1.2.1. The following data will be requested:

RAPPExport1.2.1.1. An identifier: email address.

RAPPExport1.2.1.1.1. Mandatory data.

RAPPExport1.2.1.2. A password.

RAPPExport1.2.1.2.1. Mandatory data.

RAPPExport1.2.2. EIIAPP shall communicate with Google OAuth to authenticate the user:

RAPPExport1.2.2.1. EIIAPP shall send the identifier (**RAPPExport1.2.1.1**) and password (**RAPPExport1.2.1.2**) to Google OAuth.

RAPPExport1.2.2.2. EIIAPP shall collect the response from Google OAuth:

RAPPExport1.2.2.2.1. If collected response is OK, the user will be authenticated with Google (see **RAPPExport1.1**).

RAPPExport1.2.2.2.2. If collected response is ERROR, EIIAPP shall show an error message (user will be kept non-authenticated with Google).

5.2.2.2.4 REPORT A BREAKDOWN

RAPPReport1. EIIAPP shall let an authenticated user report a Breakdown.

RAPPReport1.1. EIIAPP shall redirect the user to the School of Computer Science website for reporting breakdowns.

5.2.2.2.5 CONSULT NOTIFICATIONS

RAPPNotif1. EIIAPP shall let an authenticated user consult Notifications.

RAPPNotif1.1. EIIAPP shall obtain Notifications from database.

RAPPNotif1.1.1. EIIAPP shall show the following data for each Notification obtained from database:

RAPPNotif1.1.1.1. Title.

RAPPNotif1.1.1.2. Content.

RAPPNotif1.1.1.3. Date.

RAPPNotif1.1.1.4. Time.

5.2.2.2.6 UPDATE INFORMATION STORED ON THE DATABASE

RAPPUpdate1. EIIAPP shall let an authenticated user update the information stored on the database.

RAPPUpdate1.1. EIIAPP shall communicate with EIISERVER to update the Links of Interest.

RAPPUpdate1.1.1. EIIAPP shall collect the response from EIISERVER:

RAPPUpdate1.1.1.1. If collected response is OK, EIIAPP shall store on the database the Link of Interest contained on the response.

RAPPUpdate1.1.1.2. If collected response is ERROR, EIIAPP shall show an error message.

RAPPUpdate1.2. EIIAPP shall communicate with EIISERVER to update the Events.

RAPPUpdate1.2.1. EIIAPP shall send the identifier of the user (**RAPPAuth1.1.1**) to EIISERVER.

RAPPUpdate1.2.2. EIIAPP shall collect the response from EIISERVER:

RAPPUpdate1.2.2.1. If collected response is OK, EIIAPP shall store on the database the Events contained on the response.

RAPPUpdate1.2.2.2. If collected response is ERROR, EIIAPP shall show an error message.

5.2.2.2.7 SCHEDULE UPDATES

RAPPSchedule1. EIIAPP shall let an authenticated user schedule an update.

RSAPPSchedule1.1. The next data will be requested:

RSAPPSchedule1.1.1. Periodicity.

RSAPPSchedule1.1.1.1. Mandatory data.

RSAPPSchedule1.1.1.2. EIIAPP shall verify that it is one of the following values:

RSAPPSchedule1.1.1.2.1. DAILY.

RSAPPSchedule1.1.1.2.1. WEEKLY.

RSAPPSchedule1.1.1.2.1. NEVER.

RSAPPSchedule1.1.2. Daily Time.

RSAPPSchedule1.1.2.1. Mandatory data.

RSAPPSchedule1.1.2.2. EIIAPP shall verify that it is valid according to **RSERVValidTime1**.

RSAPPSchedule1.1.3. Weekly Time.

RSAPPSchedule1.1.3.1. Mandatory data.

RSAPPSchedule1.1.3.2. EIIAPP shall verify that it is valid according to **RSERVValidTime1**.

RSAPPSchedule1.1.4. Weekly Day.

RSAPPSchedule1.1.4.1. Mandatory data.

RSAPPSchedule1.1.4.2. EIIAPP shall verify that it is one of the following values:
RSAPPSchedule1.1.4.2.1. MONDAY.
RSAPPSchedule1.1.4.2.2. TUESDAY.
RSAPPSchedule1.1.4.2.3. WEDNESDAY.
RSAPPSchedule1.1.4.2.4. THURSDAY.
RSAPPSchedule1.1.4.2.5. FRIDAY.
RSAPPSchedule1.1.4.2.6. SATURDAY.
RSAPPSchedule1.1.4.2.7. SUNDAY.

RSAPPSchedule1.1.2. If **RSAPPSchedule1.1.1.2**, **RSAPPSchedule1.1.2.2**, **RSAPPSchedule1.1.3.2** and **RSAPPSchedule1.1.4.2** validation returns true, EIIAPP shall schedule the database:

RSAPPSchedule1.1.2.1. EIIAPP shall update the database (see **RAPPUpdate1**) when selected Periodicity (**RSAPPSchedule1.1.1.**) matches current time:

RSAPPSchedule1.1.2.1.1. If Periodicity is DAILY, when current time matches Daily Time (**RSAPPSchedule1.1.2**).

RSAPPSchedule1.1.2.1.2. If Periodicity is WEEKLY, when current time matches Weekly Time (**RSAPPSchedule1.1.3**) and current day matches Weekly Day (**RSAPPSchedule1.1.4.2**).

RSAPPSchedule1.1.2.1.3. If Periodicity is NEVER, never is updated.

RSAPPSchedule1.1.3. If **RSAPPSchedule1.1.1.2**, **RSAPPSchedule1.1.2.2**, **RSAPPSchedule1.1.3.2** or **RSAPPSchedule1.1.4.2** validation returns false, EIIAPP shall show an error message.

5.2.2.3 VALIDATION

RAPPValidUO1. EIIAPP shall verify that UO is valid:

RAPPValidUO1.1. UO starts with one of the following characters:

RAPPValidUO1.1.1. "UO".

RAPPValidUO1.1.1. "uo".

RAPPValidUO1.2. **RAPPValidUO1.1** is followed by one or more numbers.

RAPPValidUO1.3. If **RAPPValidUO1.1** and **RAPPValidUO1.2** is true, UO is valid.

RAPPValidUO1.4. If **RAPPValidUO1.1** or **RAPPValidUO1.2** is false, UO is not **valid**.

5.2.3 NON-FUNCTIONAL REQUIREMENTS

REIISERVERCommunication1. EIISERVER shall communicate with EIIAPP via HTTPS protocol.

REIISERVERCommunication2. EIISERVER shall communicate with the official information sources of School of Computer Science via HTTP protocol.

REIISERVERCommunication3. EIISERVER shall communicate with LDAP Authentication Service via the following protocols:

REIISERVERCommunication3.1. LDAPS.

REIISERVERCommunication3.2. SSL.

REIISERVERCommunication3.3. TLS.

REIISERVERCommunication4. EIISERVER shall communicate with Firebase Messaging API via HTTP protocol.

REIIAPPCommunication1. EIIAPP shall communicate with Google OAuth via OAuth 2.0 protocol.

REIIAPPCommunication2. EIIAPP shall communicate with Google Calendar API via HTTPS protocol.

5.2.4 IDENTIFICATION OF SYSTEM ACTORS

The actors of the EIISERVER and EIIAPP systems are listed in Actors of EIISERVER and Actors of EIIAPP.

5.2.4.1 ACTORS OF EIISERVER

The actors of the EIISERVER System are the following:

- **NOTIFIER**: An authenticated user in the EIISERVER system that can send broadcast notifications to students authenticated in the EIIAPP.
- **ADMINISTRATOR**: An authenticated user in the EIISERVER system that can update the information stored related to the Software Engineering Degree and the Master in Web Engineering as well as the information contained in the Links of Interest Page.
- **LDAP Authentication Service**: Service of the University of Oviedo used to authenticate a user of the EIISERVER.

- **Firebase Messaging API**: Service used to send broadcast notifications to students authenticated in the EIIAPP.
- **Non-Authenticated User**: A user that is no authenticated in EIISERVER can authenticate on EIISERVER system.
- **MIW and GIISOF information endpoints** [3]–[5]**:** Used to obtain the information related to Software Engineering Degree and Master in Web Engineering.

5.2.4.2 ACTORS OF EIIAPP

The actors of the EIIAPP System are the following:

- **Authenticated User**: An authenticated user in the EIIAPP that can: consult Links of Interest, Events, Notifications, Export Events, Report a Breakdown, Update Information and Schedule Updates.
- **Non-Authenticated User**: A user that is no authenticated in EIIAPP can authenticate on the system.
- **Google Calendar API**: Service used to export the events of an authenticated user.
- **Google OAuth Service**: Service used to authenticate a user of the EIIAPP.
- **EIISERVER**: EIISERVER itself is also an actor, from EIIAPP point of view, it is external to this system and exposes an API for communication.

5.2.5 USE CASE SPECIFICATION

The Use Case Diagrams of **EIIAPP** and **EIISERVER** are illustrated in Figure 44: Use Case Diagram. EIIAPP and Figure 45: Use Case Diagram. EIISERVER, and documented in EIIAPP Use Cases Documentation and EIISERVER Use Cases Documentation, respectively.

EIISERVER Use Case Diagram is **refined** in: Figure 46: Use Case Diagram. EIISERVER. Send Notifications Refinement and Figure 47: Use Case Diagram. EIISERVER. Manage Entities Refinement.

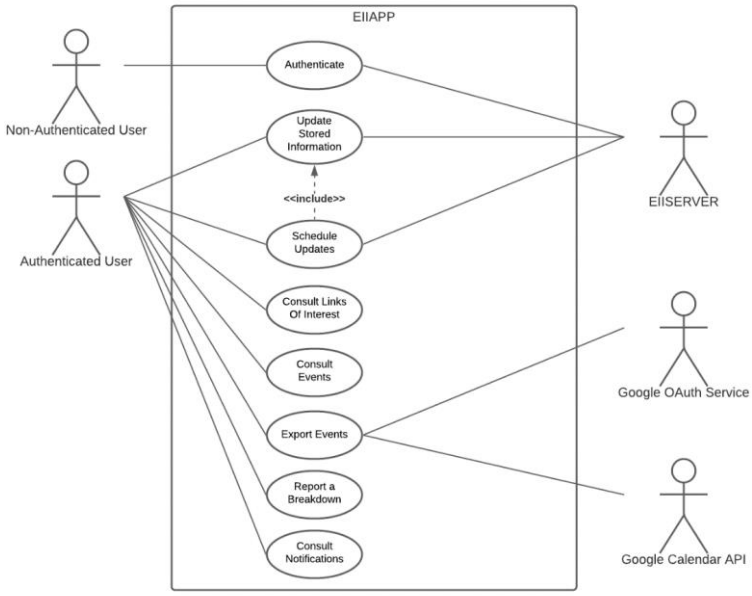


Figure 44: Use Case Diagram. EIIAPP

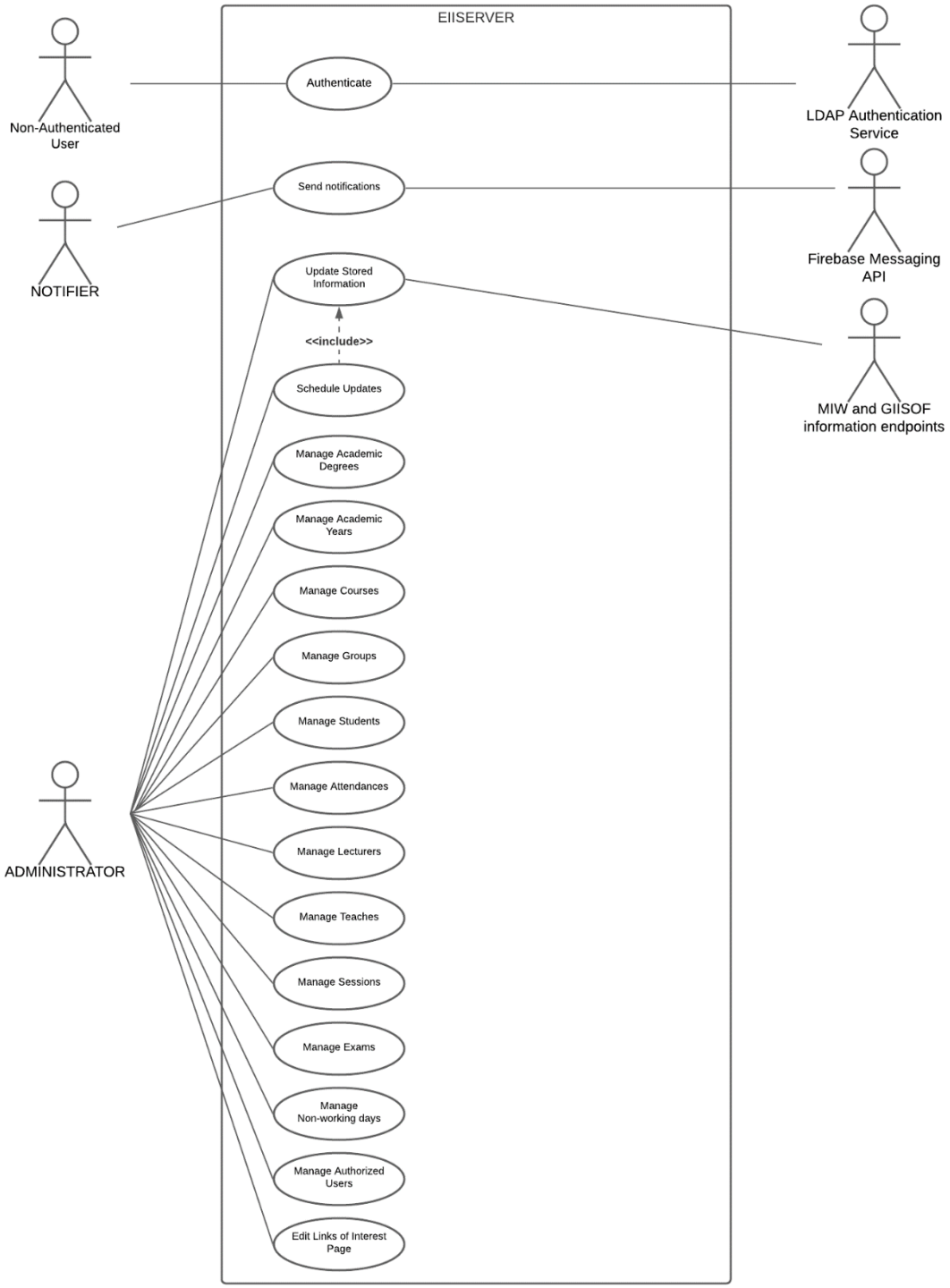


Figure 45: Use Case Diagram. EIISERVER

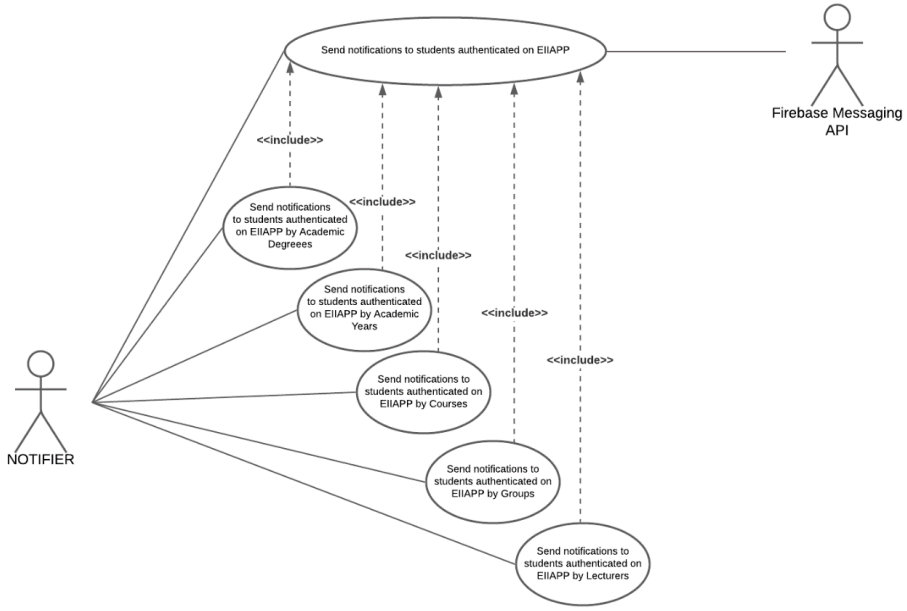


Figure 46: Use Case Diagram. EIISERVER. Send Notifications Refinement

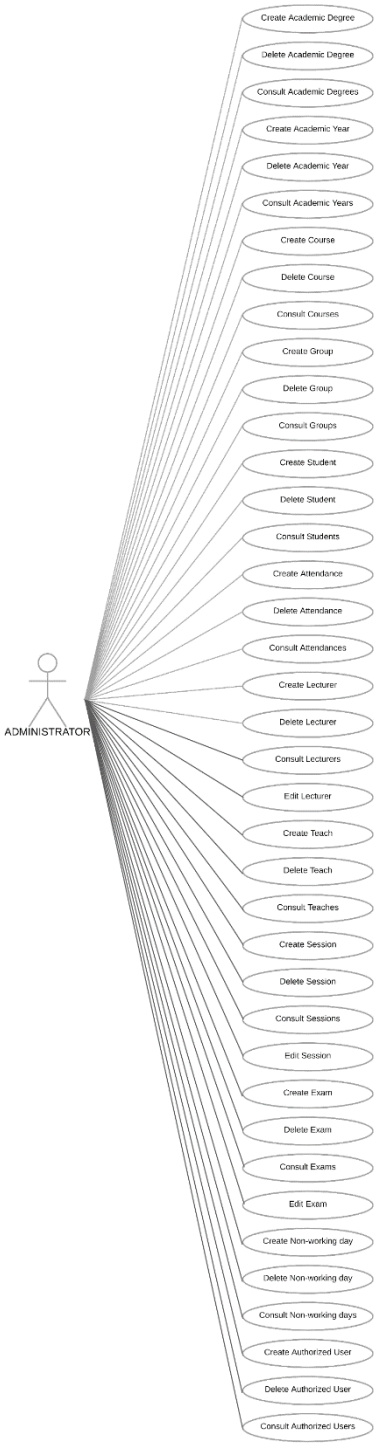


Figure 47: Use Case Diagram. EIISERVER. Manage Entities Refinement

5.2.5.1 EIIAPP USE CASES DOCUMENTATION

EIIAPP use cases are documented in tables from Table 14: Use Case Documentation. EIIAPP. Authenticate to Table 21: Use Case Documentation. EIIAPP. Consult Notifications.

Use Case	Authenticate	EIIAPP-UC-1
Actors	Non-Authenticated User, EIISERVER	
Started By	Non-Authenticated User	
BASIC PATH		
ACTOR	**SYSTEM**	
1. User introduces his UO.	2. Validates data: Checks UO format is valid, communicates with EIISERVER to authenticate the user.	
	3. Checks EIISERVER response is OK.	
ALTERNATIVE PATHS		
2a. System returns a validation error. Indicates it to the user and flow returns to 1. 3a. EIISERVER response is not OK. Indicates it to the user and flow returns to 1.		

Table 14: Use Case Documentation. EIIAPP. Authenticate

Use Case	Update Stored Information	EIIAPP-UC-2
Actors	Authenticated User, EIISERVER	
Started By	Authenticated User	
BASIC PATH		
ACTOR	**SYSTEM**	
1. User updates information.	2. Communicates with EIISERVER, passes authenticated user's UO.	
	3. Checks EIISERVER response is OK.	
ALTERNATIVE PATHS		
3a. EIISERVER response is not OK. Indicates it to the user and flow returns to 1.		

Table 15: Use Case Documentation. EIIAPP. Update Stored Information

Use Case	Schedule Updates	EIIAPP-UC-3
Actors	Authenticated User, EIISERVER	
Started By	Authenticated User	
Related Use Cases	EIIAPP-UC-2	
BASIC PATH		
ACTOR	**SYSTEM**	
1. User configures updates schedules: selects periodicity.	2. Validates data: checks configuration is valid.	
	3. Stores update configuration on database.	
	4. When current time matches selected periodicity, flow changes to step 1 of EIIAPP-UC-2.	
	4. Stores EIISERVER (EIIAPP-UC-2) response on database.	
ALTERNATIVE PATHS		
2a. System returns a validation error. Indicates it to the user and flow returns to 1. 3a. Database returns an error. Indicates it to the user and flow returns to 1. 4a. Database returns an error. Indicates it to the user and flow returns to 1.		

Table 16: Use Case Documentation. EIIAPP. Schedule Updates

Use Case	Consult Links of Interest	EIIAPP-UC-4
Actors	Authenticated User	
Started By	Authenticated User	
BASIC PATH		
ACTOR	SYSTEM	
1. User consults Links of Interest.	2. Gets Links of Interest from database.	
ALTERNATIVE PATHS		
2a. Database returns an error. Indicates it to the user and flow returns to 1.		

Table 17: Use Case Documentation. EIIAPP. Consult Links of Interest

Use Case	Consult Events	EIIAPP-UC-5
Actors	Authenticated User	
Started By	Authenticated User	
BASIC PATH		
ACTOR	SYSTEM	
1. User consults Events.	2. Gets Events from database.	
ALTERNATIVE PATHS		
2a. Database returns an error. Indicates it to the user and flow returns to 1.		

Table 18: Use Case Documentation. EIIAPP. Consult Events

Use Case	Export Events	EIIAPP-UC-6
Actors	Authenticated User, Google OAuth Service, Google Calendar API	
Started By	Authenticated User	
BASIC PATH		
ACTOR	SYSTEM	
1. User introduces its Google email and password.	2. Communicates with Google OAuth, passes user's email and password.	
	3. Checks Google OAuth response is OK	
	4. Communicates with Google Calendar API to obtain the Calendars of the Authenticated User	
	5. Checks Google Calendar API response is OK.	
6. Selects a calendar.		
	7. Gets Events from database.	
	8. Communicates with Google Calendar API, passes: selected calendar, events obtained from database.	
	9. Checks Google Calendar API response is OK.	
ALTERNATIVE PATHS		
1a. User is already Authenticated (on Google). Flow starts at 2. 3a. Google OAuth response is not OK. Indicates it to the user and flow returns to 1. 5a. Google Calendar API response is not OK. Indicates it to the user and flow returns to 1. 7a. Database returns an error. Indicates it to the user and flow returns to 6. 9a. Google Calendar API response is not OK. Indicates it to the user and flow returns to 6.		

Table 19: Use Case Documentation. EIIAPP. Export Events

Use Case	Report a Breakdown	EIIAPP-UC-7
Actors	Authenticated User	
Started By	Authenticated User	
BASIC PATH		
ACTOR	SYSTEM	
1. User reports a breakdown.	2. User is redirected to the School of Computer Science website to report a breakdown	

Table 20: Use Case Documentation. EIIAPP. Report a breakdown

Use Case	Consult Notifications	EIIAPP-UC-8
Actors	Authenticated User	
Started By	Authenticated User	
BASIC PATH		
ACTOR	SYSTEM	
1. User consults Notifications.	2. Gets Notifications from database.	
ALTERNATIVE PATHS		
2a. Database returns an error. Indicates it to the user and flow returns to 1.		

Table 21: Use Case Documentation. EIIAPP. Consult Notifications

5.2.5.2 EIISERVER USE CASES DOCUMENTATION

EIISERVER use cases are documented in tables from Table 22: Use Case Documentation. EIISERVER. Authenticate to Table 30: Use Case Documentation. EIISERVER. Edit Links of Interest Page.

Note that all Use Cases refined from "Send Notifications" Use Case (see Figure 45: Use Case Diagram. EIISERVER) follow the same basic path and alternative paths shown in Table 23: Use Case Documentation. EIISERVER. Send Notification to Students authenticated on EIIAPP. No other tables are documented to avoid repetition.

Note also that all Use Cases refined from "Manage Academic Degrees", "Manage Academic Years", "Manage Courses", "Manage Groups", "Manage Students", "Manage Attendances", "Manage Lecturers", "Manage Teaches", "Manage Sessions", "Manage Exams", "Manage non-working days" and "Manage Authorized Users" Use Cases follow the same basic path and alternatives paths shown in: Table 26: Use Case Documentation. EIISERVER. Create Academic Degree, Table 27: Use Case Documentation. EIISERVER. Delete Academic Degree, Table 28: Use Case Documentation. EIISERVER. Consult Academic Degrees and Table 29: Use Case Documentation. EIISERVER. Edit Lecturer. No other tables are documented to avoid repetition.

Use Case	Authenticate	EIISERVER-UC-1
Actors	Non-Authenticated User, LDAP Authentication Service	
Started By	Non-Authenticated User	
BASIC PATH		
ACTOR	SYSTEM	
1. User introduces his username and password	2. Checks user is authorized.	
	3. Communicates with LDAP Authentication Service, passes username and password.	
	4. Checks LDAP Authentication Service response is OK.	
ALTERNATIVE PATHS		
2a. User is not authorized. Indicates it to the user and flow returns to 1. 4a. LDAP Authentication Service response is not OK. Indicates it to the user and flow returns to 1.		

Table 22: Use Case Documentation. EIISERVER. Authenticate

Use Case	Send Notifications to Students authenticated on EIIAPP	EIISERVER-UC-2
Actors	NOTIFIER, Firebase Messaging API	
Started By	NOTIFIER	
BASIC PATH		
ACTOR	SYSTEM	
1. User introduces: notification title, notification content and recipients (students).	2. Validates data: Checks notification title and notification content are valid.	
	3. Checks students are stored on database.	
	4. Communicates with Firebase Messaging API, passes notification title, notification content and recipients.	
	5. Checks Firebase Messaging API response is OK.	
ALTERNATIVE PATHS		
2a. System returns a validation error. Indicates it to the user and flow returns to 1. 3a. Students are not stored on database. Indicates it to the user and flow returns to 1. 5a. Firebase Messaging API response is not OK. Indicates it to the user and flow returns to 1.		

Table 23: Use Case Documentation. EIISERVER. Send Notification to Students authenticated on EIIAPP

Use Case	Update Stored Information	EIISERVER -UC-3
Actors	ADMINISTRATOR, MIW and GIISOF endpoints	
Started By	ADMINISTRATOR	
BASIC PATH		
ACTOR	SYSTEM	
1. User updates information.	2. Communicates with MIW and GIISOF endpoints.	
	3. Checks MIW and GIISOF endpoints response is OK.	
ALTERNATIVE PATHS		
3a. MIW and GIISOF endpoints response is not OK. Indicates it to the user and flow returns to 1.		

Table 24: Use Case Documentation. EIISERVER. Update Stored Information

Use Case	Schedule Updates	EIISERVER -UC-4
Actors	ADMINISTRATOR, EIISERVER	
Started By	ADMINISTRATOR	
Related Use Cases	EIISERVER -UC-3	
BASIC PATH		
ACTOR	SYSTEM	
1. User configures updates schedules: selects periodicity.	2. Validates data: checks configuration is valid.	
	3. Stores update configuration on database.	
	4. When current time matches selected periodicity, flow changes to step 1 of EIISERVER -UC-3.	
	4. Stores MIW and GIISOF endpoints (EIISERVER-UC-3) response on database.	
ALTERNATIVE PATHS		
2a. System returns a validation error. Indicates it to the user and flow returns to 1. 3a. Database returns an error. Indicates it to the user and flow returns to 1. 4a. Database returns an error. Indicates it to the user and flow returns to 1.		

Table 25: Use Case Documentation. EIISERVER. Schedule Updates

Use Case	Create Academic Degree	EIISERVER-UC-5
Actors	ADMINISTRATOR	
Started By	ADMINISTRATOR	
BASIC PATH		
ACTOR	SYSTEM	
1. User introduces Academic Degree data.	2. Validates data: Checks academic degree data is valid.	
	3. Checks Academic Degree is not already stored on database.	
	4. Stores Academic Degree on database.	
ALTERNATIVE PATHS		
2a. System returns a validation error. Indicates it to the user and flow returns to 1. 3a. Database returns an error. Indicates it to the user and flow returns to 1.		

Table 26: Use Case Documentation. EIISERVER. Create Academic Degree

Use Case	Delete Academic Degree	EIISERVER-UC-6
Actors	ADMINISTRATOR	
Started By	ADMINISTRATOR	
BASIC PATH		
ACTOR	SYSTEM	
1. User introduces Academic Degree identifier.	2. Checks Academic Degree is stored on database.	
	3. Deletes Academic Degree from database.	
ALTERNATIVE PATHS		
2a. Database returns an error. Indicates it to the user and flow returns to 1.		

Table 27: Use Case Documentation. EIISERVER. Delete Academic Degree

Use Case	Consult Academic Degrees	EIISERVER-UC-7
Actors	ADMINISTRATOR	
Started By	ADMINISTRATOR	
BASIC PATH		
ACTOR	SYSTEM	
1. User consults Academic Degrees.	2. Gets Academic Degrees from database	
ALTERNATIVE PATHS		
2a. Database returns an error. Indicates it to the user and flow returns to 1.		

Table 28: Use Case Documentation. EIISERVER. Consult Academic Degrees

Use Case	Edit Lecturer	EIISERVER-UC-8
Actors	ADMINISTRATOR	
Started By	ADMINISTRATOR	
BASIC PATH		
ACTOR	SYSTEM	
1. User introduces Lecturer data.	2. Validates data: Checks lecturer data is valid.	
	3. Checks Lecturer is stored on database.	
	4. Updates Lecturer on database.	
ALTERNATIVE PATHS		
2a. System returns a validation error. Indicates it to the user and flow returns to 1. 3a. Database returns an error. Indicates it to the user and flow returns to 1.		

Table 29: Use Case Documentation. EIISERVER. Edit Lecturer

Use Case	Edit Links of Interest Page	EIISERVER-UC-9
Actors	ADMINISTRATOR	
Started By	ADMINISTRATOR	
BASIC PATH		
ACTOR	SYSTEM	
1. User introduces Links of Interest data.	2. Validates data: Checks data is valid.	
	4. Stores Links of Interest on database.	
ALTERNATIVE PATHS		
2a. System returns a validation error. Indicates it to the user and flow returns to 1.		

Table 30: Use Case Documentation. EIISERVER. Edit Links of Interest Page

5.3 IDENTIFICATION OF ANALYSIS SUBSYSTEMS

In this section, **EIISERVER** and **EIIAPP** systems will be analysed and deconstructed in **subsystems**. The **interfaces** between those subsystems will also be described.

On the one hand, **EIISERVER subsystems** are described in Reader Subsystem, Datasource Subsystem, LDAP Authentication Datasource Subsystem and PUSH notification Datasource Subsystem. On the other hand, **EIIAPP subsystems** are described in Export Events To Google Calendar Subsystem and **the interface between EIISERVER and EIIAPP** is described in EIISERVER Subsystem.

5.3.1 READER SUBSYSTEM

Reader Subsystem stands as EIISERVER system **data entry point**. It is located at **Transport Layer** of EIISERVER (see Definition of the technological architecture) and connects, via **network**, with the **official sources (endpoints) of School of Computer Science Software Engineering Degree and Master in Computer Science** [3]–[5] via its exposed APIs, using **HTTP** protocol. It collects the information from these sources, parses it and stores it on the system.

5.3.2 DATASOURCE SUBSYSTEM

Datasource Subsystem, located at **Datasource Layer**, connects with **PostgreSQL locally** (local **TCP/IP** connections [11]), via 5433 port to manage EIISERVER persistence.

5.3.3 LDAP AUTHENTICATION DATASOURCE SUBSYSTEM

LDAP Authentication Datasource Subsystem, located at **Datasource Layer**, connects with LDAP Authentication System of the University of Oviedo, via **Secure Lightweight Directory Access Protocol (LDAPS)**, **Secure Sockets Layer (SSL)** and **Transport Layer Security (TLS)** protocols.

5.3.4 PUSH NOTIFICATION DATASOURCE SUBSYSTEM

PUSH Notification Datasource Subsystem, located at **Datasource Layer**, connects with Firebase Messaging, via **network**, using **HTTP** protocol.

5.3.5 EIISERVER SUBSYSTEM

EIISERVER itself also stands as **subsystem**. EIIAPP connects with EIISERVER, via **network**, using its exposed **REST API** and **HTTPS** protocol.

5.3.6 EXPORT EVENTS TO GOOGLE CALENDAR SUBSYSTEM

Export Events to Google Calendar Subsystem, located at **Datasource** Layer, connects with **Google Calendar API**, via **network**, using **HTTPS** protocol, and **OAuth 2.0.** protocol to authenticate and authorize Google Users.

5.4 DEFINING USER INTERFACES

5.4.1 INTERFACE EVOLUTION

Within this section, we will display the **evolution of the more remarkable aspects and features of the interfaces** of EIISERVER and EIIAPP. Specially of EIIAPP since it has suffered from a major evolution to thrive into a more usable application.

The evolution of Home View can be seen on Figure 48: Interface Evolution. EIIAPP. Home View. The first picture represents part of the **prototype** handed in to the client, the School of Software Engineering. The final picture, the third one, evolves from the second, adding a **staggered view with all the routes** of the application and the **events preview** in a row with a state indicating the status of each event (how much time is left until the start, if the event has already started or if it has finished). It uses blue and green colours to match the **School's Look and Feel**, as well as the School's Logo itself.

The **buttons** that appeared on the second picture were **removed** according to **Material Design Guidelines** [12] since button texts should occupy just one line.

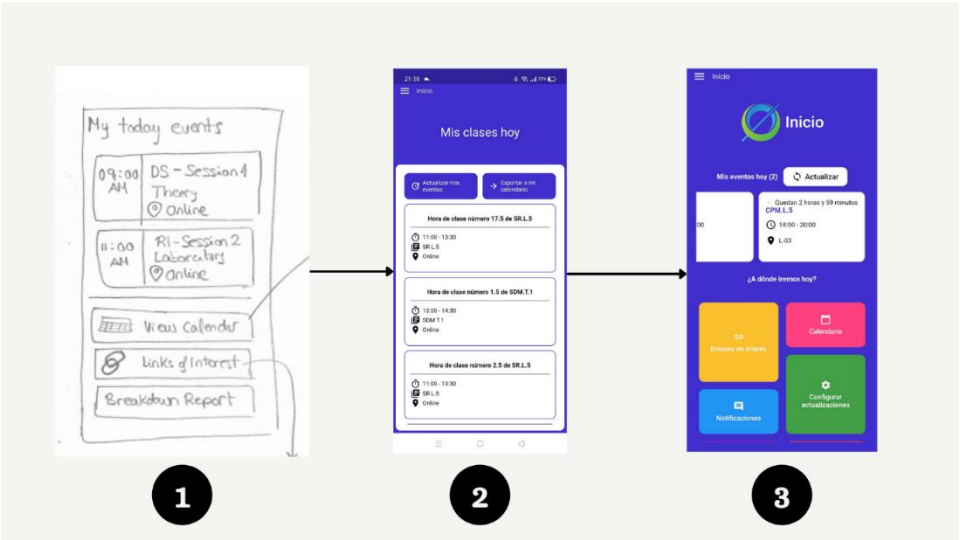


Figure 48: Interface Evolution. EIIAPP. Home View

The evolution of the Login Page is shown in Figure 49: Interface Evolution. EIIAPP. Login View. The prototype evolves to a minimalistic interface with the School Logo on the top. It also uses blue and green colours to match the **School's Look and Feel** to the detriment of lilac that appears on the second picture.

The **bottom navigation bar** that appears on the second picture is **removed** according to **Material Design Guidelines** [12] and replaced by a Navigation Drawer. The reason behind this change is that EIIAPP had more than five main destinations and a **Navigation Drawer** was more suitable.

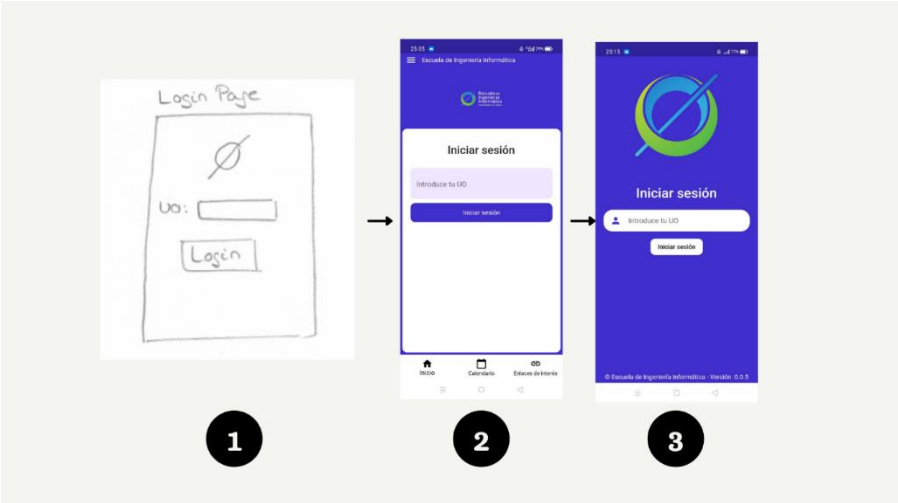


Figure 49: Interface Evolution. EIIAPP. Login View

The **Links Of Interest View** evolution is detailed on Figure 50: Interface Evolution. EIIAPP. Links Of Interest View.

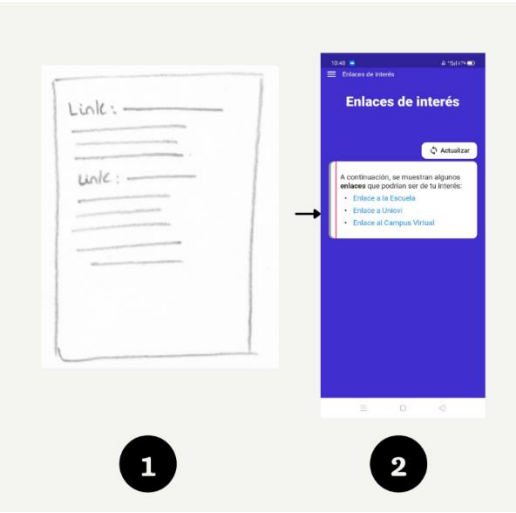


Figure 50: Interface Evolution. EIIAPP. Links Of Interest View

The **Export my events View** evolution is detailed on Figure 51: Interface Evolution. EIIAPP. Export my events View. For more information about the elements of this view, see Export my events.

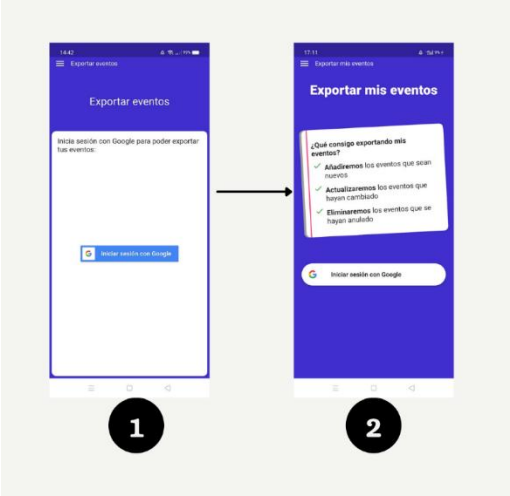


Figure 51: Interface Evolution. EIIAPP. Export my events View

The evolution of the Navigation Drawer is illustrated on Figure 52: Interface Evolution. EIIAPP. Navigation Drawer. The School's brand that appeared on the first picture was removed according to **Material Design Guidelines** [12] and replaced by the header that is displayed on the second one, showing user's UO and email account. The reason behind this changed is based on the fact that **the heading of a drawer should not be used for branding or commercial purposes**.

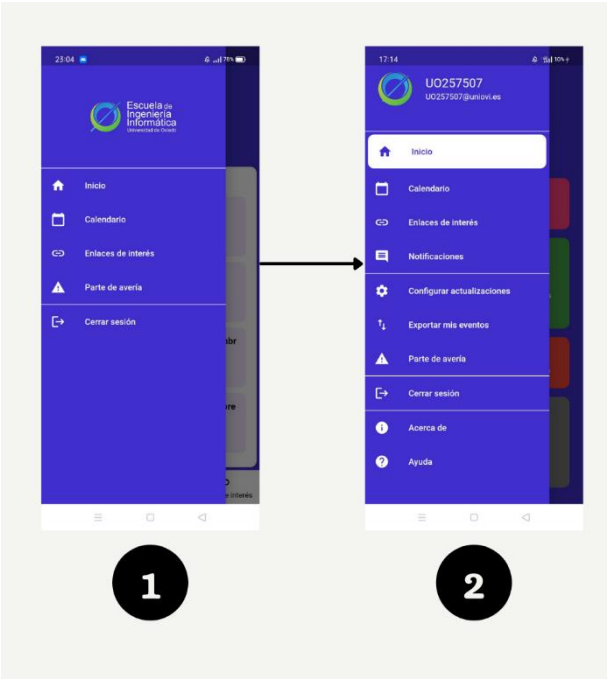


Figure 52: Interface Evolution. EIIAPP. Navigation Drawer

The evolution of the Calendar View is deconstructed on Figure 53: Interface Evolution. EIIAPP. Calendar View. The third picture added **symbols under days that contained events** (see Calendar) and indicated the **weekday** of every day displayed: 'Lun' for Mondays, 'Mar' for Tuesdays (like the 5th of October), 'Mié' for Wednesdays (like the 6th of October) and so on. **Saturdays and Sundays were highlighted** with a **bold** font weight and a **green** colour to facilitate the visualization of each end of week. The fourth view added **arrows at both sides of the days scroll view** to indicate that days can be scrolled.

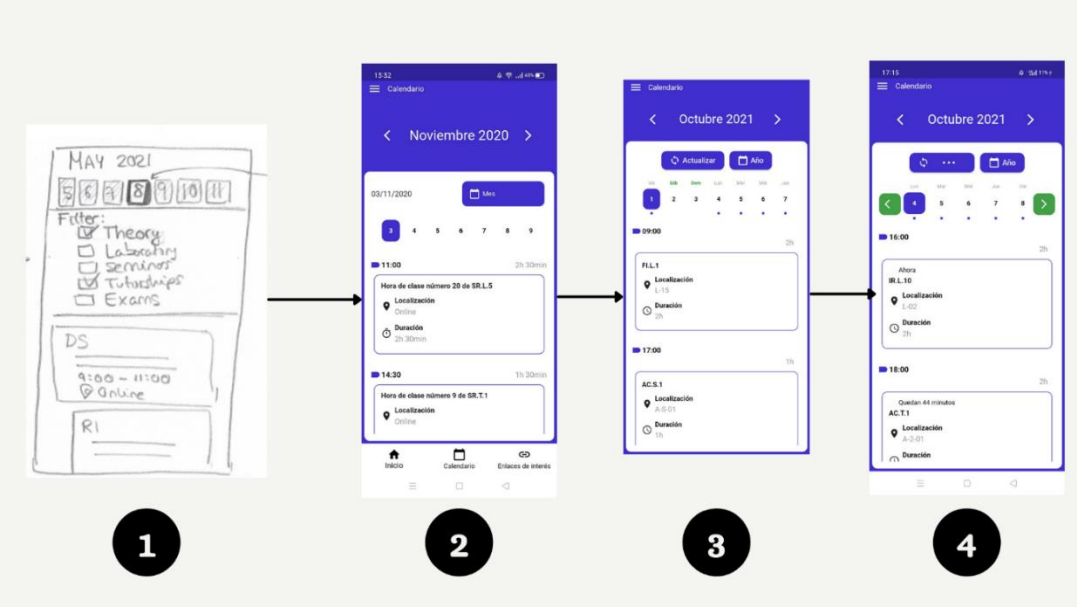


Figure 53: Interface Evolution. EIIAPP. Calendar View

Figure 54: Interface Evolution. EIIAPP. Notification View shows the evolution of the Notification View, using blue and green colours to match the **School's Look and Feel** to the detriment of lilac that appears on the first picture, as well as **adding a background** (third picture) to evoke the well-known default backgrounds from messaging applications such as **Telegram** or **WhatsApp**.

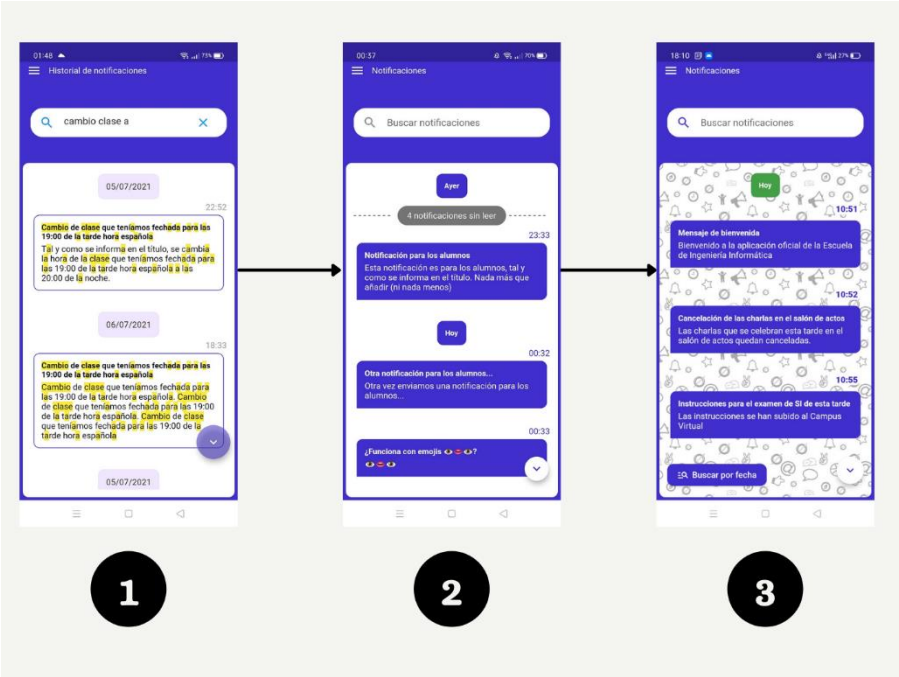


Figure 54: Interface Evolution. EIIAPP. Notification View

The evolution of the **EIISERVER** system can be seen from the prototype (see Figure 55: Interface Evolution. EIISERVER) to the figures shown in Interface aspect description of EIISERVER, since there are no major or remarkable changes such as the ones discussed before.

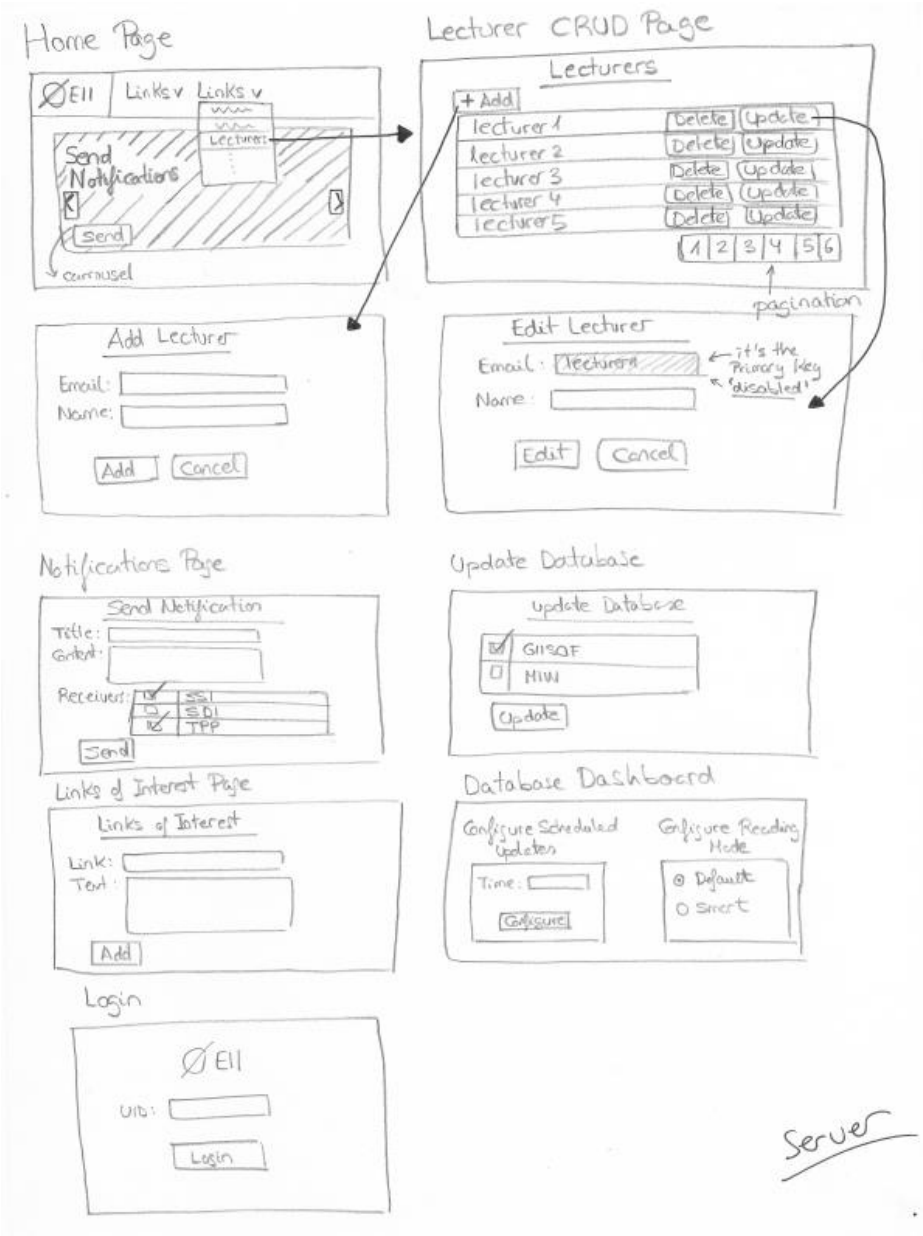


Figure 55: Interface Evolution. EIISERVER

5.4.2 INTERFACE ASPECT DESCRIPTION

Within this section, we will describe, deconstruct and analyse the taxonomy of all interfaces contained in both systems: **EIISERVER** and **EIIAPP**.

5.4.2.1 INTERFACE ASPECT DESCRIPTION OF EIIAPP

The descriptions of the interfaces of **EIIAPP** are specified in the sections below:

5.4.2.1.1 HOME

When an authenticated user opens EIIAPP, it is redirected to Home View which is illustrated in Figure 56: Interface Aspect Description of EIIAPP. Home View. Within this view, we present a **preview with the events**, of the authenticated user, **that occur on the current day** to lessen the

number of taps a user has to made if he only wants to know what events has that day (he **need not enter to the Calendar**).

Figure 56: Interface Aspect Description of EIIAPP. Home View

We also present a **staggered view** with all the locations of the application wrapped within cards: Links of Interest, Calendar, Notifications, Updates Configuration, Export my events, Breakdown Report, Help and About Us. By using a staggered view, we are applying the general **principle** of usability **of familiarity** since this element appears in multiple applications and systems, such as Windows 8 layouts and GAME Application for Android Devices (see Figure 57: Interface Aspect Description of EIIAPP. Home View. Staggered Views).

A layout for **landscape** orientation was also designed, as seen in Figure 58: Interface Aspect Description of EIIAPP. Home View. Portrait and Landscape. The strategy followed takes advantage of the horizontal space of the screen to display the today events preview on half of such space and the other half for the staggered view.

On the other hand, the **portrait** layout takes advantage of the vertical space, displaying the today events preview on a row placed on the top and the staggered view on the bottom (see Figure 56: Interface Aspect Description of EIIAPP. Home View).

Figure 57: Interface Aspect Description of EIIAPP. Home View. Staggered Views

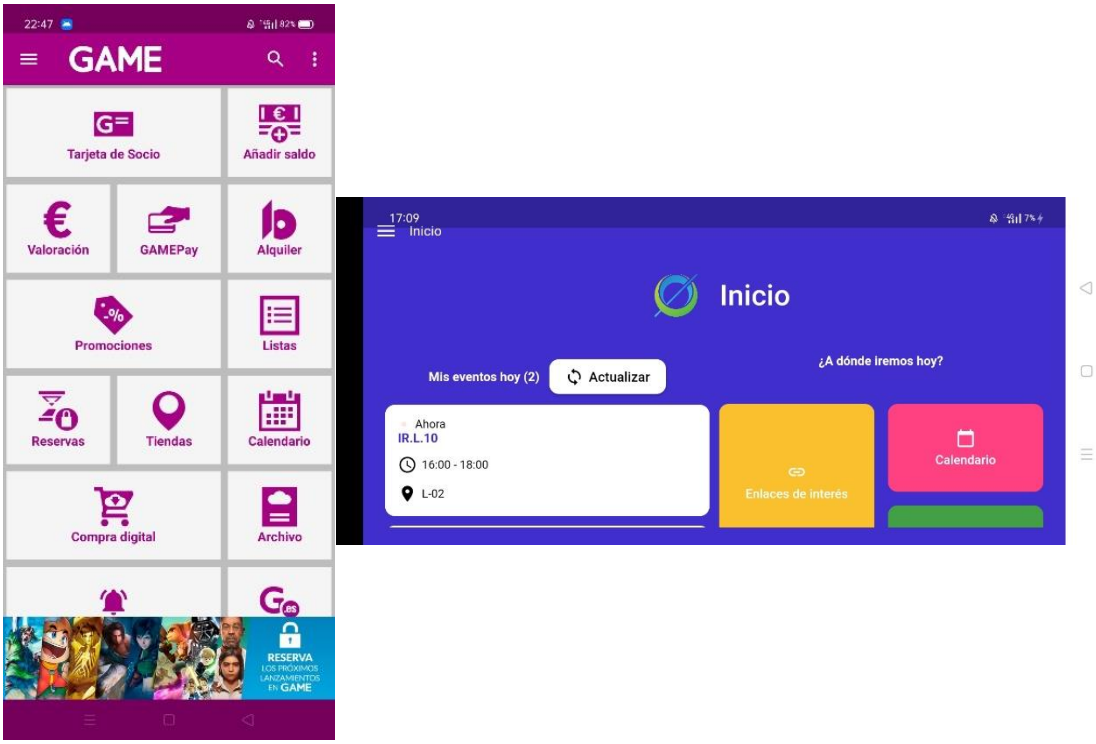


Figure 58: Interface Aspect Description of EIIAPP. Home View. Portrait and Landscape

The today events preview consists of a series of event **cards**. Within each card there is a **state bar** which indicates the state of the event at the current day and time. Figure 59: Interface Aspect Description of EIIAPP. Home View. Today events states illustrates all possible states of the event "SI T.1".The first state reminds the user that there are **4 minutes until the start of the event**, the second one, that the event has already **started** and the last one, that the event has **finished**.

Figure 59: Interface Aspect Description of EIIAPP. Home View. Today events states

Figure 60: Interface Aspect Description of EIIAPP. Home View. Events iconography

To **distinguish** between events of different types, a different symbol for each one is used, as illustrated in Figure 60: Interface Aspect Description of EIIAPP. Home View. Events iconography. A **graduation cap** for events of type **exam** and a **bullet** for events of type session.

On other hand, the symbol of a **clock** is shown to indicate the event duration and a **marker** for the location.

Note that the **meaning of all these symbols** is represented also textually along the screens of EIIAPP, including **Help View**, as Figure 61: Interface Aspect Description of EIIAPP. Home View. Symbols Meaning illustrates.

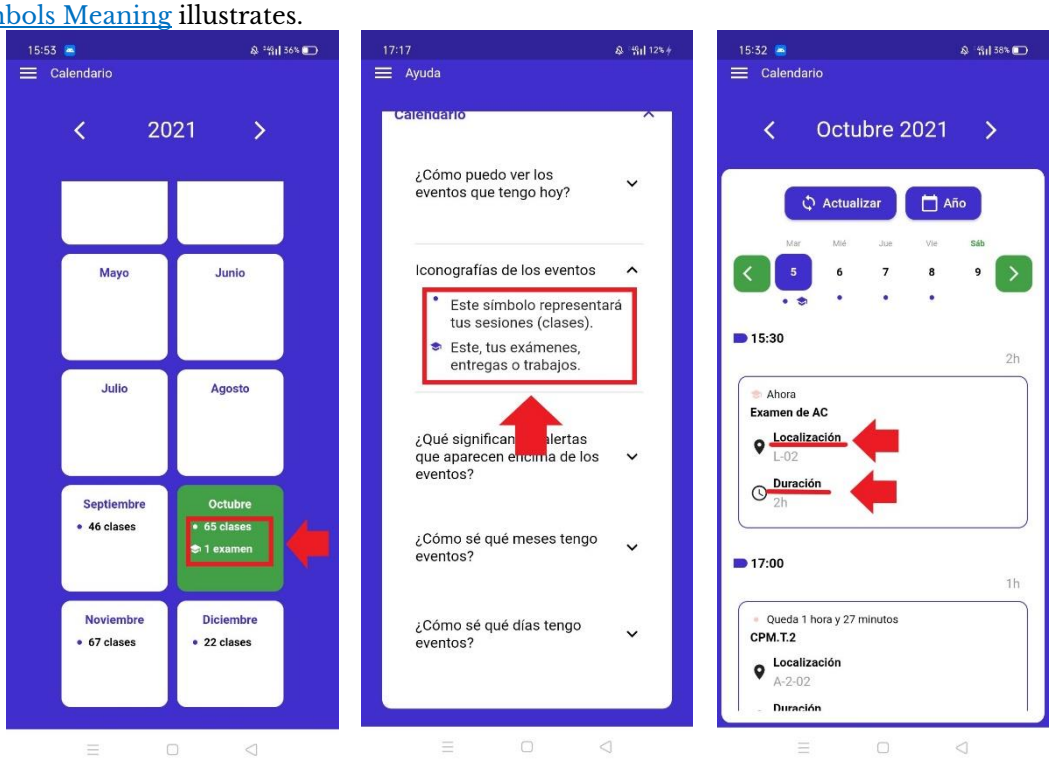
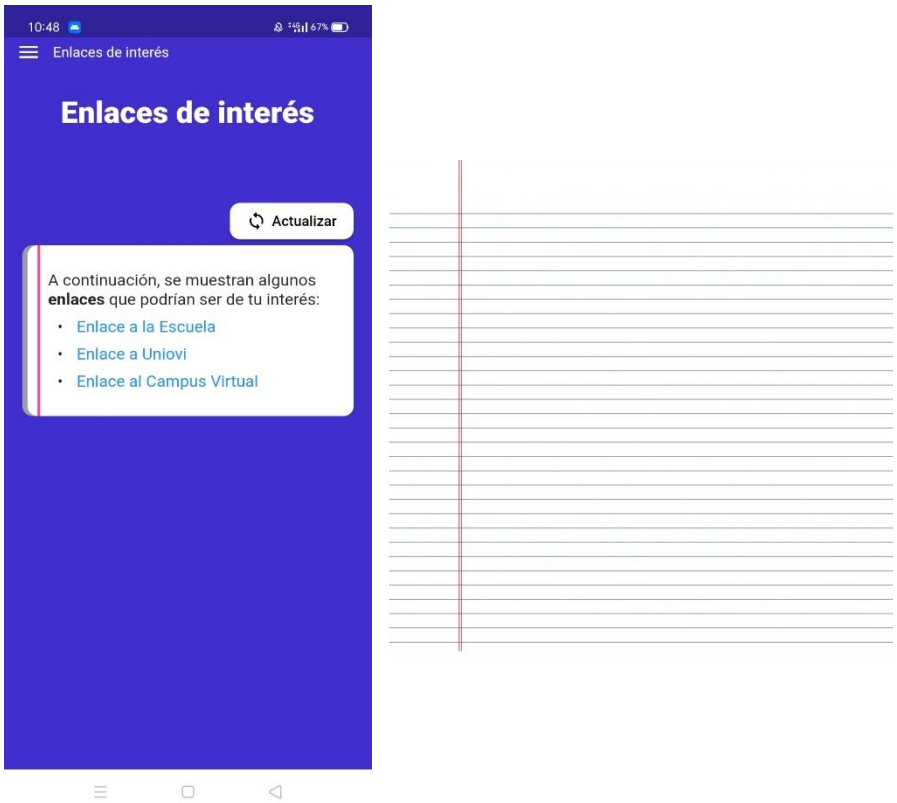


Figure 61: Interface Aspect Description of EIIAPP. Home View. Symbols Meaning

Another usability feature to point out is that when a user **taps on an event on** Home View **it is redirected to the Calendar View**, with current day as selected day.

5.4.2.1.2 LINKS OF INTEREST



For the Links of Interest View, a card wrapper that **reminds of a notebook page** with a red band is used to evoke a piece of paper with some interesting information within.

We also use the **well-known blue colour for hyperlinks**, as shown in Figure 62: Interface Aspect Description of EIIAPP. Links of Interest View.

Figure 62: Interface Aspect Description of EIIAPP. Links of Interest View

5.4.2.1.3 CALENDAR

The Calendar View is illustrated in Figure 63: Interface Aspect Description of EIIAPP. Calendar View. On the top of the view, the selected month and year is displayed between two arrows: the one on the left decrements selected month, and the one on the right increments it, as shown on Figure 64: Interface Aspect Description of EIIAPP. Calendar View. Month Switchers. The **familiarity** principle also applies here, since this element has appeared throughout systems such as **Google Calendar**, among many others.

We also use the symbols illustrated on Figure 61: Interface Aspect Description of EIIAPP. Home View. Symbols Meaning, a **bullet** for sessions and a **graduation cap for exams** here on the days scroll view of Calendar View (see Figure 63: Interface Aspect Description of EIIAPP. Calendar View).

When a day like the 5th of October has **sessions and exams**, we paint a **bullet along with a graduation cap**. When a day like the 6th of October has **only sessions**, a **bullet**; and so on. When a day has **no events** like the 9th of October, **no symbol is painted**. For this reason, we are applying the **predictability** principle, since a user can predict whether it has events on a day just by looking at the symbols bellow that day.

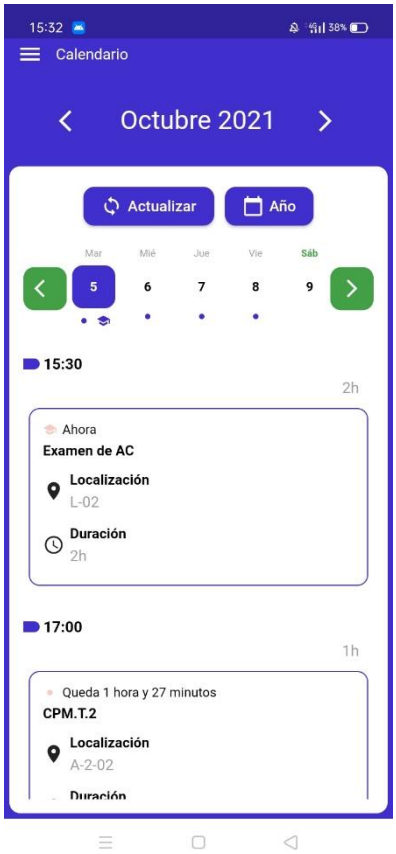


Figure 63: Interface Aspect Description of EIIAPP. Calendar View

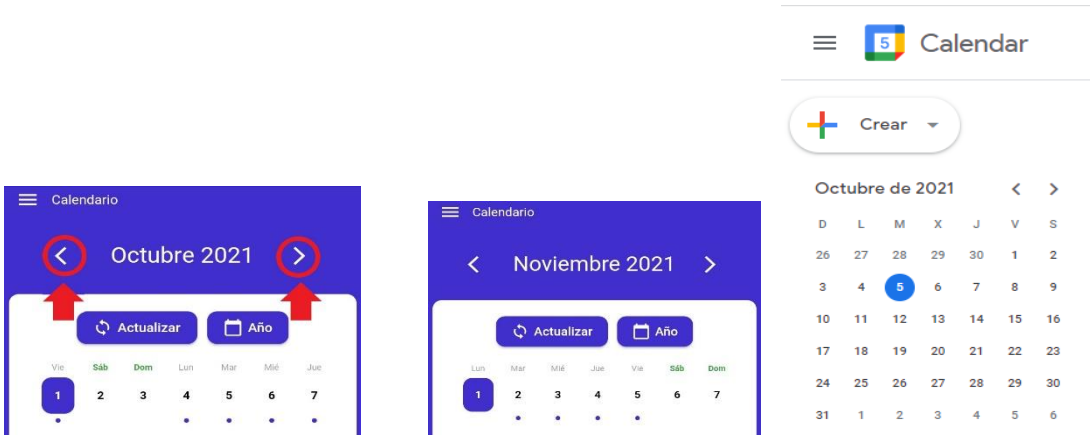


Figure 64: Interface Aspect Description of EIIAPP. Calendar View. Month Switchers

The same principle is used on the Calendar Month View, as seen on [Figure 65: Interface Aspect Description of EIIAPP. Calendar Year View]. The months where the user has **sessions and exams**, like October, a **bullet along with a graduation cap** is painted. The months where the user has only **sessions**, like September, a **bullet along** is painted. The months where the user has **no events**, like May, **no symbols** are painted.

Figure 65: Interface Aspect Description of EIIAPP. Calendar Year View

Another aspect to point out is that on **Calendar View** we indicate the user that **selected day** is the 5th of October by **wrapping it within a blue rectangle** (see Figure 63: Interface Aspect Description of EIIAPP. Calendar View) and on **Calendar Year View**, October is wrapped within a green rectangle for the same reason (see Figure 65: Interface Aspect Description of EIIAPP. Calendar Year View).

We also indicate the **weekday** of every day displayed on Figure 63: Interface Aspect Description of EIIAPP. Calendar View: 'Lun' for Mondays, 'Mar' for Tuesdays (like the 5th of October), 'Mié' for Wednesdays (like the 6th of October) and so on. **Saturdays and Sundays are highlighted** with a **bold** font weight and a **green** colour to facilitate the visualization of each end of week.

Another remarkable feature is that when a user accesses to **Calendar View**, the **selected day is the current one**, and on **Calendar Year View**, the **selected month** is also the **current** one. The current day on the case illustrated on Figure 63: Interface Aspect Description of EIIAPP. Calendar View was the 5th of October, and the current month on Figure 65: Interface Aspect Description of EIIAPP. Calendar Year View was October.

In Calendar View as well as in Home View, **the first event that is presented to the user is the nearest to the current hour**. In Figure 63: Interface Aspect Description of EIIAPP. Calendar View, the nearest event is "Examen AC" (note that it has already started and has not finished) and that is why

is the first one on the list. On Figure 58: Interface Aspect Description of EIIAPP. Home View. Portrait and Landscape, it was “IR L.10”.

Note that the scroll view is initially placed on the nearest unfinished event, but the **user can scroll to past events** whenever he wants to.

Calendar Year View also offers **a layout that takes advantage of the available horizontal space**, that is why on Portrait (see Figure 66: Interface Aspect Description of EIIAPP. Calendar Year View Landscape) there are only two months per row but on Landscape, five months.

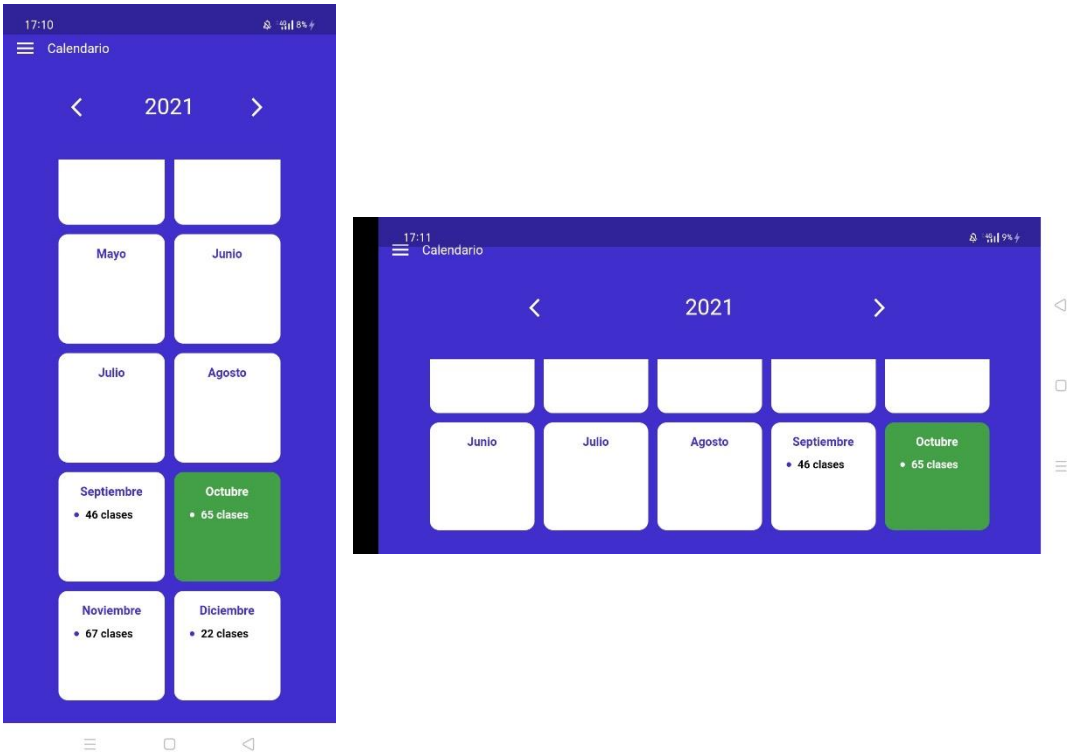


Figure 66: Interface Aspect Description of EIIAPP. Calendar Year View Landscape

A layout for **landscape** orientation was also designed for **Calendar View**, as seen in Figure 67: Interface Aspect Description of EIIAPP. Calendar View Landscape. The strategy followed takes advantage of the horizontal space of the screen to display the events on half of such space and the other half for the scroll view of days.

On the other hand, the **portrait** layout takes advantage of the vertical space, displaying the scroll view of days on a row placed on the top and the events on the bottom.

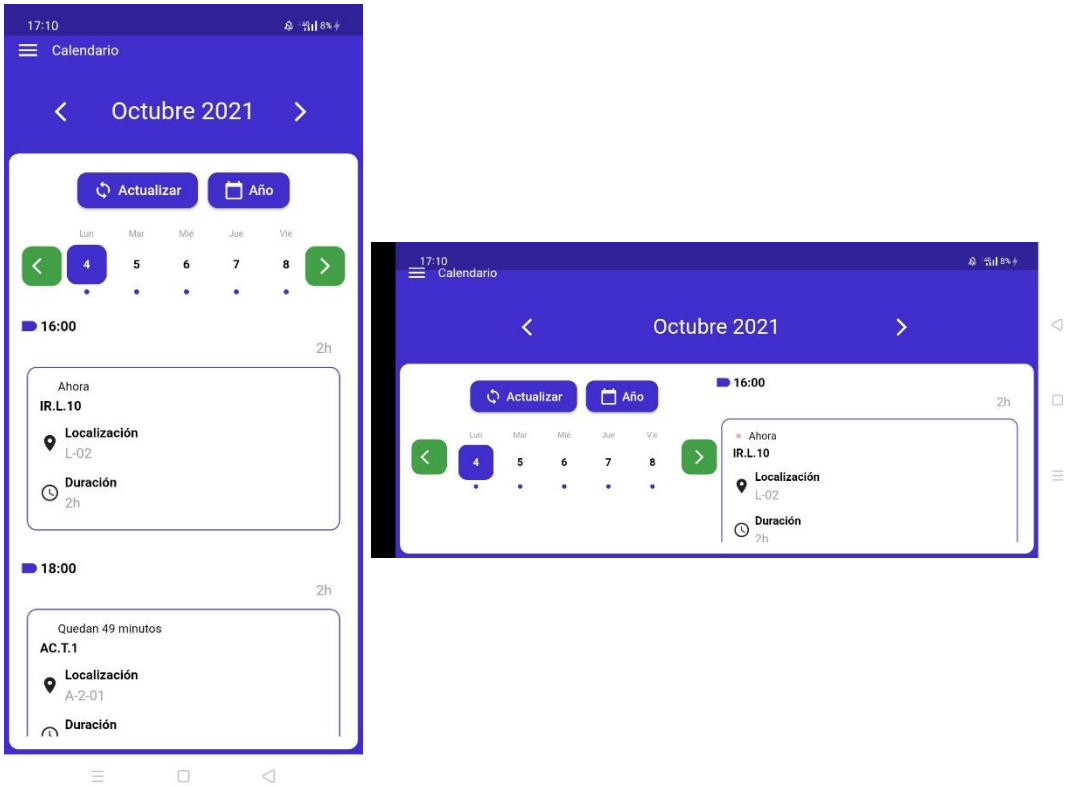


Figure 67: Interface Aspect Description of EIIAPP. Calendar View Landscape

On the other hand, when a user taps on the right arrow placed inside a green rectangle in the days scroll view, and the view is **on the last segment of the selected month** (see Figure 68: Interface Aspect Description of EIIAPP. Calendar View. Jump to next month), it jumps to the 1st of the next month. On the figure, it jumps from the last segment of October to the 1st of November.

If the user is on the first segment of the selected month and taps on the left arrow, it jumps to the last day of the previous month. On Figure 68: Interface Aspect Description of EIIAPP. Calendar View. Jump to next month it would jump from November to the 31th of October.

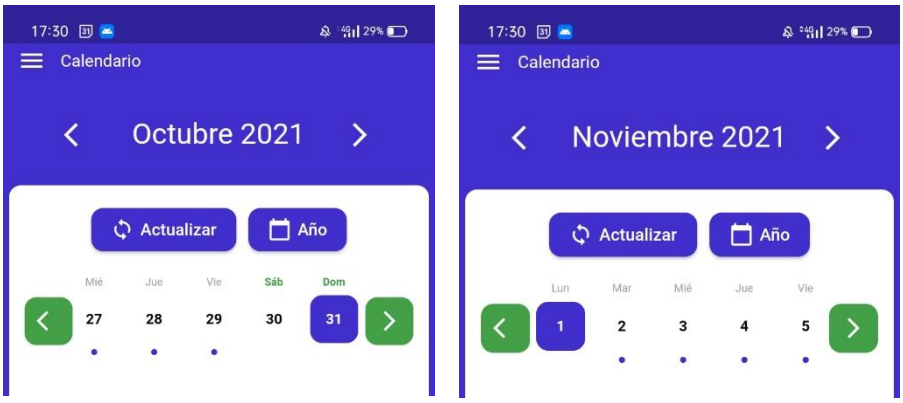


Figure 68: Interface Aspect Description of EIIAPP. Calendar View. Jump to next month

5.4.2.1.4 NOTIFICATIONS

The Notification View is illustrated in Figure 69: Interface Aspect Description of EIIAPP. Notification View. On the top of the view a **search bar** is placed where notifications can be searched by words, on the bottom, a button that enables **date searching**. A **text chip is displayed above unread notifications** (see the Figure previously mentioned) such as other well-known messaging applications (WhatsApp, Telegram, etc.).

Figure 69: Interface Aspect Description of EIIAPP. Notification View

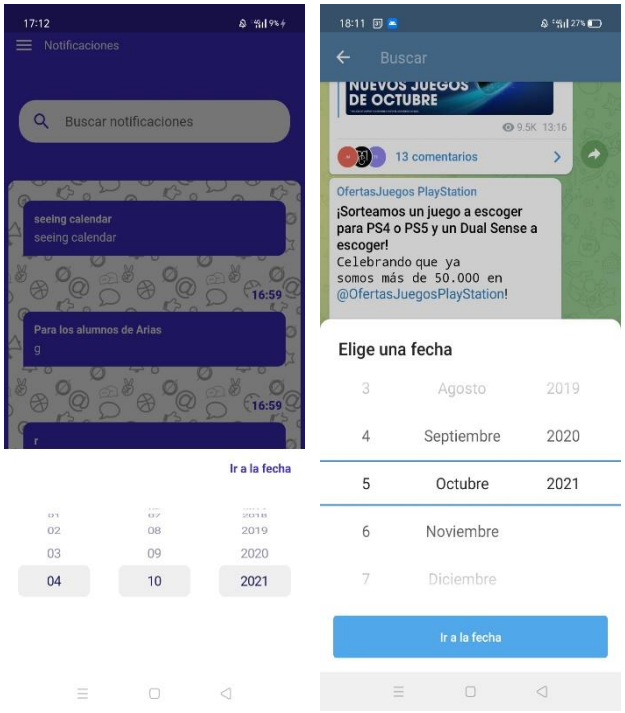


Figure 70: Interface Aspect Description of EIIAPP. Notification View vs. Telegram

These aspects reflect the principle of **familiarity** applied here.

On Figure 70: Interface Aspect Description of EIIAPP. Notification View vs. Telegram, we can see how **Telegram**, among other messaging application, has also placed a search bar on top and a button to search by dates on the bottom.

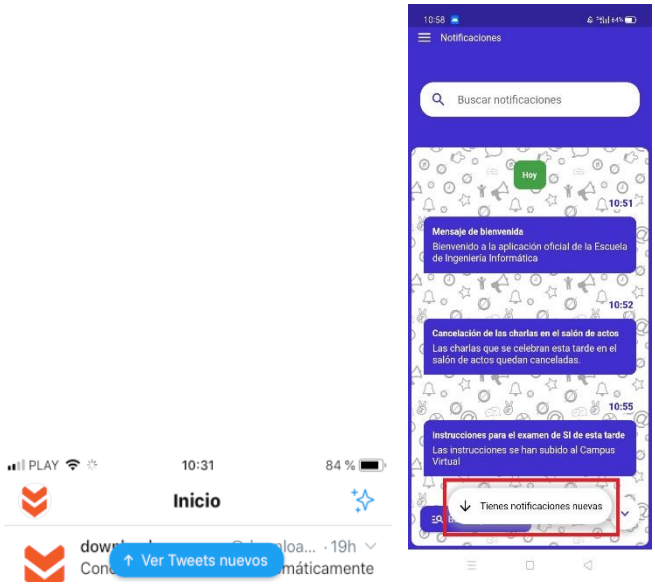
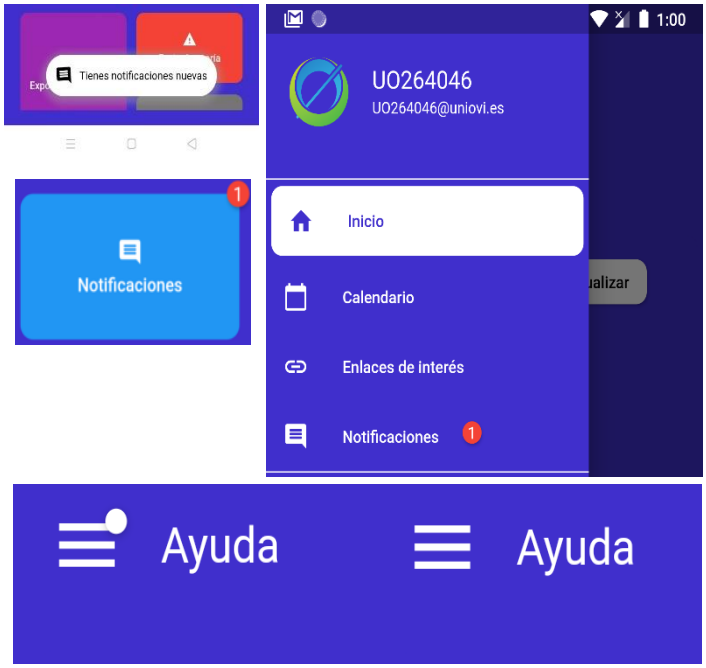


Figure 71: Interface Aspect Description of EIIAPP. Notification View vs. Twitter

When the application is opened (on Foreground, following Flutter conventions) and the user is on the Notifications View, an alert is displayed on the bottom of the screen indicating that new messages have been received below. This alert also complies with the principle of **familiarity** since other social networks such as **Twitter** has a similar interface element (see Figure 71: Interface Aspect Description of EIIAPP. Notification View vs. Twitter).



When the application is also opened but the user is not on the Notifications View, a similar **alert is displayed** but this time indicating that new notifications are received on the Notifications View (see the first picture on the left on Figure 72: Interface Aspect Description of EIIAPP. Notification View. New notification indicator).

We also use the **well-known badges** in the Notification Card displayed on the Home View, on the Notification Tile in the Navigation Drawer and even on the icon that opens the Navigation Drawer, just like **Dropbox** does on Figure 73: Interface Aspect Description of EIIAPP. Dropbox Badge.

Figure 72: Interface Aspect Description of EIIAPP. Notification View. New notification indicator

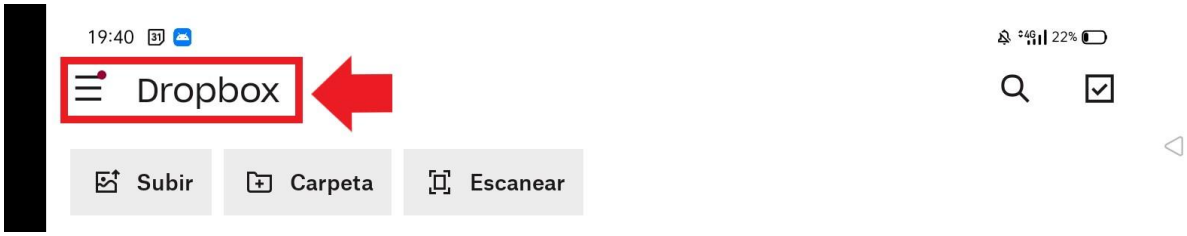


Figure 73: Interface Aspect Description of EIIAPP. Dropbox Badge

As we can see on Figure 74: Interface Aspect Description of EIIAPP. Floating Action Button, a **Floating Action Button** is placed on the right bottom corner of the view that scrolls down to the last notification of the view. This element is a fundamental part of a messaging application such as **WhatsApp** or **Telegram**.

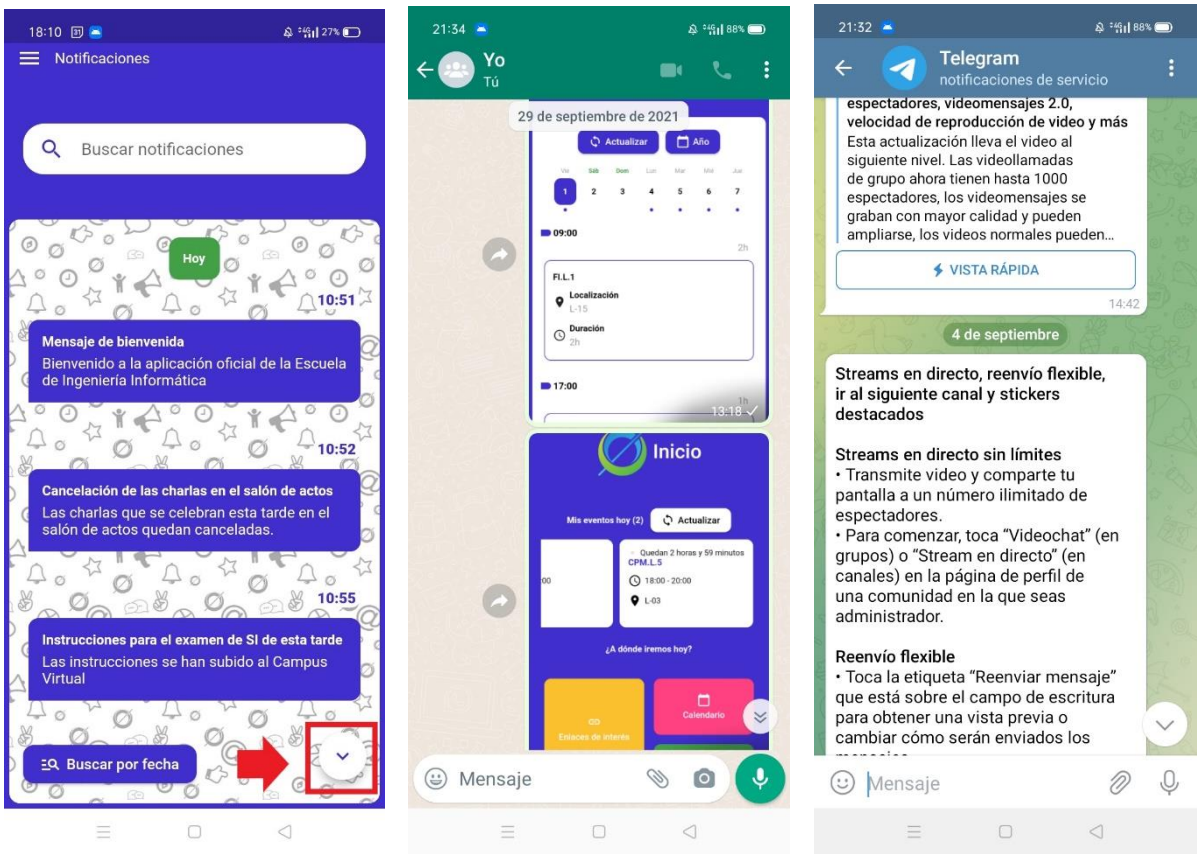


Figure 74: Interface Aspect Description of EIIAPP. Floating Action Button

As shown on Figure 74: Interface Aspect Description of EIIAPP. Floating Action Button, we **separate** notifications by date **displaying a bubble** or chip that sticks on the top of the view when user scrolls and disappears when user stops scrolling (on EIIAPP is the bubble that says "Hoy", "29 de septiembre de 2021" on WhatsApp and "4 de septiembre" on Telegram).

5.4.2.1.5 UPDATES CONFIGURATION

The Updates Configuration View is illustrated on Figure 75: Interface Aspect Description of EIIAPP. Updates Configuration View. On the first picture on the left, we can see that **radio buttons** are used to allow users to select one option from a set of three: Daily updates, weekly updates or never. To select the time, we make use of **Material Design** [12] **Time Picker widget**, as shown on the second and third picture of the previously mentioned figure. And to the select a day, a **dialog with radio buttons** also provided by **Material Design** [12].

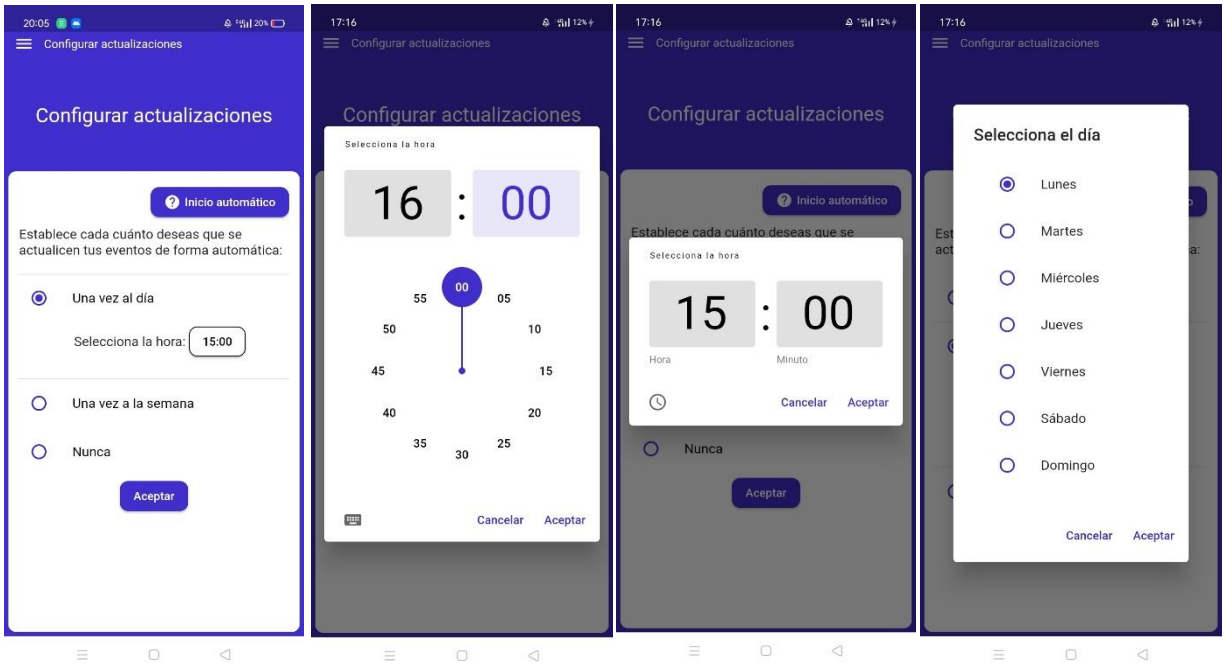


Figure 75: Interface Aspect Description of EIIAPP. Updates Configuration View

5.4.2.1.6 EXPORT MY EVENTS

When a non-authenticated-with-Google user accesses the Export my events view, it is shown the first picture on the left on Figure 76: Interface Aspect Description of EIIAPP. Export my events View. We have applied the principle of **familiarity** by using the **Login-with-Google** as well as **Logout-from-Google** the button, both **recommended** to be used **by Google** guidelines on **Material Design** [12].

When a user has selected a calendar on the second picture of Figure 76: Interface Aspect Description of EIIAPP. Export my events View, the third picture is displayed. On that picture we apply the **solidity principle of usability** about **response times** by notifying the user with a progress bar because response times can be greater than a few seconds. With that **progress bar**, we are also applying the **solidity principle of usability** about **observability**.

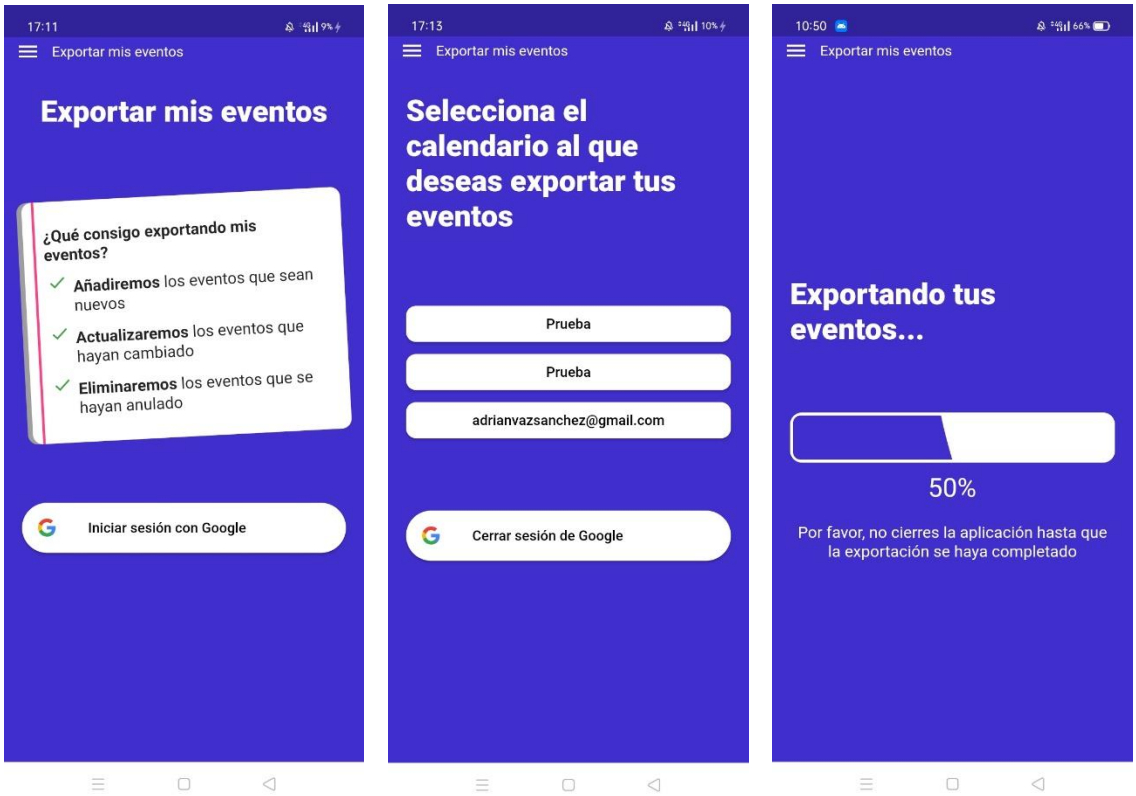


Figure 76: Interface Aspect Description of EIIAPP. Export my events View

5.4.2.1.7 BREAKDOWN REPORT

Figure 77: Interface Aspect Description of EIIAPP. Breakdown Report View

The Breakdown Report View is displayed on Figure 77: Interface Aspect Description of EIIAPP. Breakdown Report View. It includes a button that redirects to an external form of the School of Computer Science.

5.4.2.1.8 HELP

The Help View is illustrated on Figure 78: Interface Aspect Description of EIIAPP. Help View. We have used **expansible panels** or accordions that can be collapsed/expanded depending on the needs of the user.

Note that the **Help Tile** on the Navigation Drawer takes **last place**, as well as the last card of the staggered view placed on Home View, a **well-known** place for the Help route on software systems.

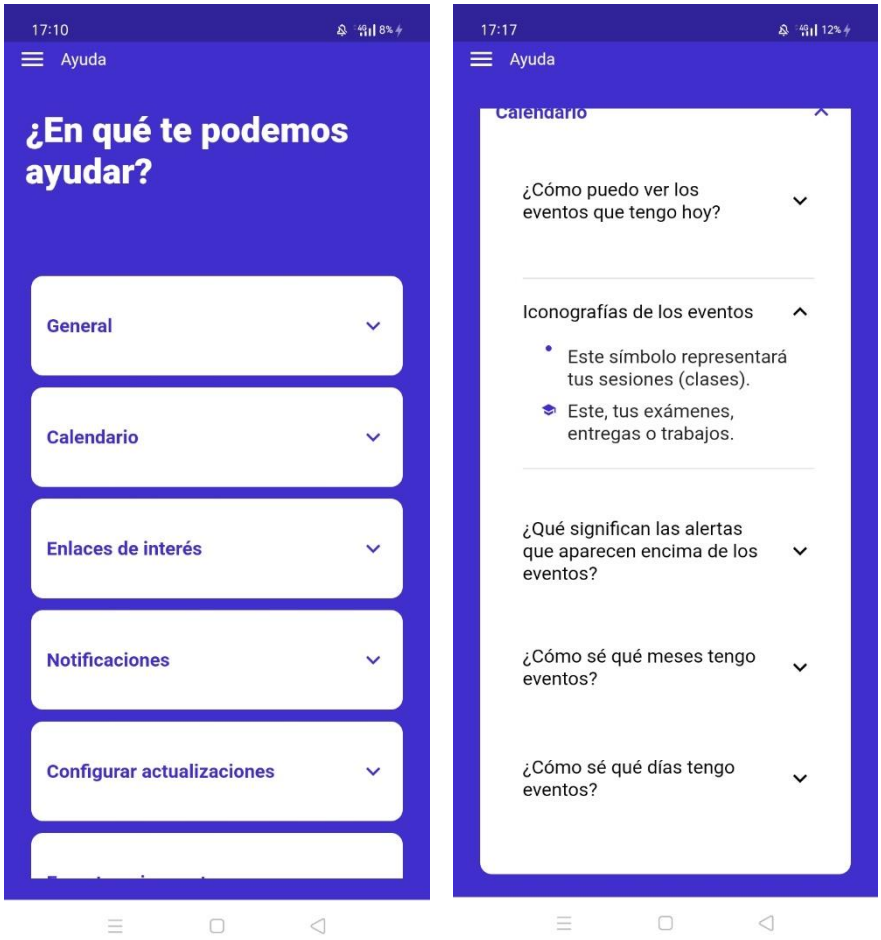


Figure 78: Interface Aspect Description of EIIAPP. Help View

5.4.2.1.9 ABOUT US

Figure 79: Interface Aspect Description of EIIAPP. About Us View

The About Us View is illustrated on Figure 79: Interface Aspect Description of EIIAPP. About Us View. Note that the **About Us Tile** on the Navigation Drawer takes **the penultimate place**, as well as the penultimate card of the staggered view placed on Home View, above Help Routes.

5.4.2.1.10 ANOTHER ASPECTS

Since we are building an application for a **real client**, the School of Compute Science, we have elaborated a **Look and Feel** specially crafted for this client and based on the colours and aspect of its logo (see Figure 80: Interface Aspect Description of EIIAPP. School Logo).

Figure 80: Interface Aspect Description of EIIAPP. School Logo

The **logo** is present throughout many routes of EIIAPP: Splash Screen, Navigation Drawer, Home View and even on the Application Icon (see Figure 81: Interface Aspect Description of EIIAPP. School Logo in EIIAPP). The School Logo plays the role of a **spinner** in loading screens for Export my events View and Login View, among others.

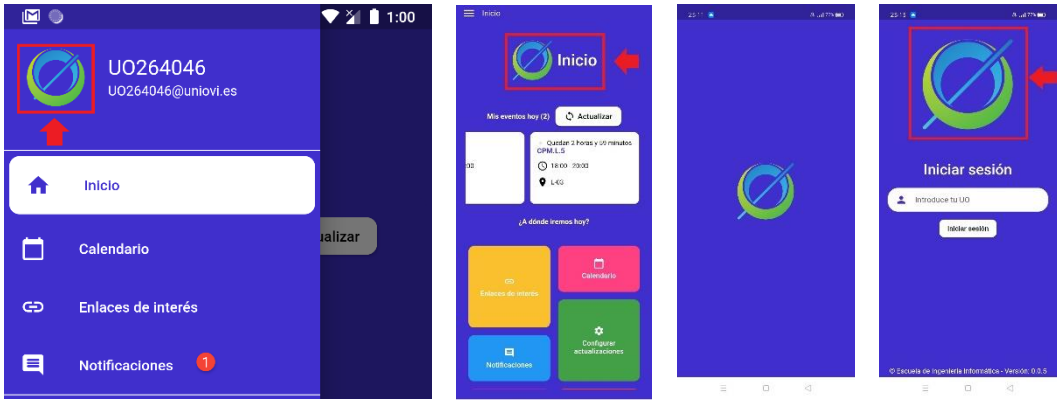


Figure 81: Interface Aspect Description of EIIAPP. School Logo in EIIAPP

And the **colours** of the logo: blue and green, are also used in places and UI elements. Some minimal examples can be seen on Figure 82: Interface Aspect Description of EIIAPP. School Colours on EIIAPP.

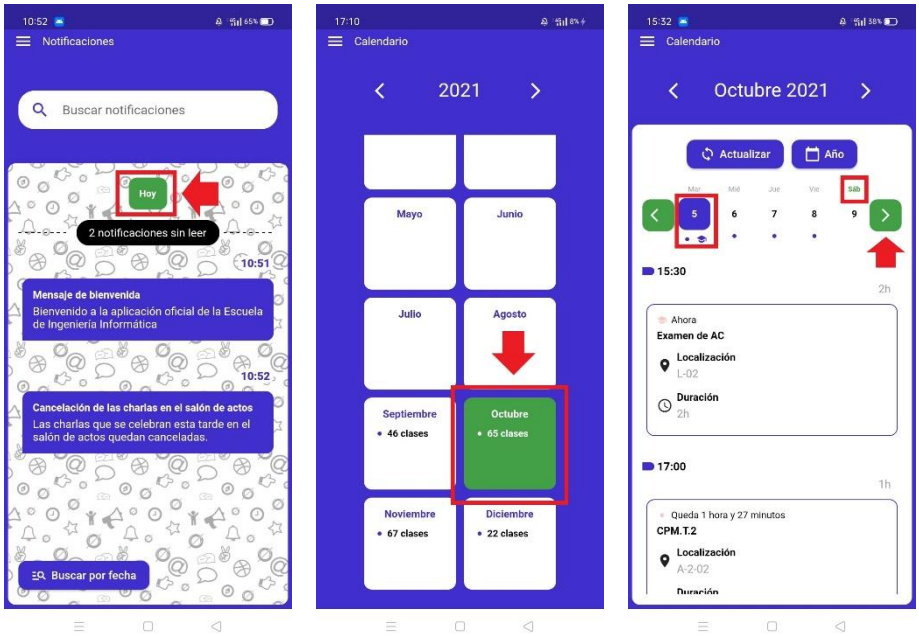


Figure 82: Interface Aspect Description of EIIAPP. School Colours on EIIAPP

5.4.2.2 INTERFACE ASPECT DESCRIPTION OF EIISERVER

The descriptions of the interfaces of **EIISERVER** are specified in the sections below:

5.4.2.2.1 HOME

The Home View of EIISERVER is illustrated on Figure 83: Interface Aspect Description of EIISERVER. Home View. A **toolbar** is placed on the top of the view, with all the **options** that the authenticated user can do **depending on its role**. On the previously mentioned figure, the authenticated user had the role of an ADMINISTRATOR. At the end of the toolbar the “About Us”, “Help” and “Profile” options are displayed. By clicking on “Profile” a drop-down menu is displayed with the following list of options: “Change Role” and “Log out”.

The School Logo is placed on the left corner of the same toolbar and all options with a little triangle on the right side open a drop-down menu with a list of options.

We also make use of a carousel, as shown in Figure 83: Interface Aspect Description of EIISERVER. Home View, with the main routes of EIISERVER, which are also present on the options displayed on the toolbar.

Figure 83: Interface Aspect Description of EIISERVER. Home View

The list of options displayed on each drop-down menu are **grouped by categories**, as illustrated on Figure 84: Interface Aspect Description of EIISERVER. Home View. Drop-down menus and **separated by thin grey lines** to facilitate the differentiation between different actions and groups of actions.

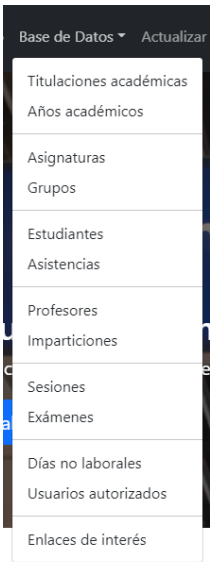


Figure 84: Interface Aspect Description of EIISERVER. Home View. Drop-down menus

5.4.2.2.2 ENTITIES VIEW

The **dashboard views** for all entities: Academic degrees, academic years, courses, groups, students, attendances, lecturers, teaches, sessions, exams and for non-working days and authorized users follow the same interface design presented on Figure 85: Interface Aspect Description of EIISERVER. Entities View. On the particular case represented by the mentioned figure, we are accessing the Academic Degrees View.

All academic degrees are listed on a table. On each row, an entity is represented along with a button, with the icon of a rubbish bin, on the right that **deletes the entity**. To **add a new entity**, a button with the icon of a plus sign is displayed on the top of the table.

The table is also **pageable** and the user can **configure** the number of entities that are listed on each page.

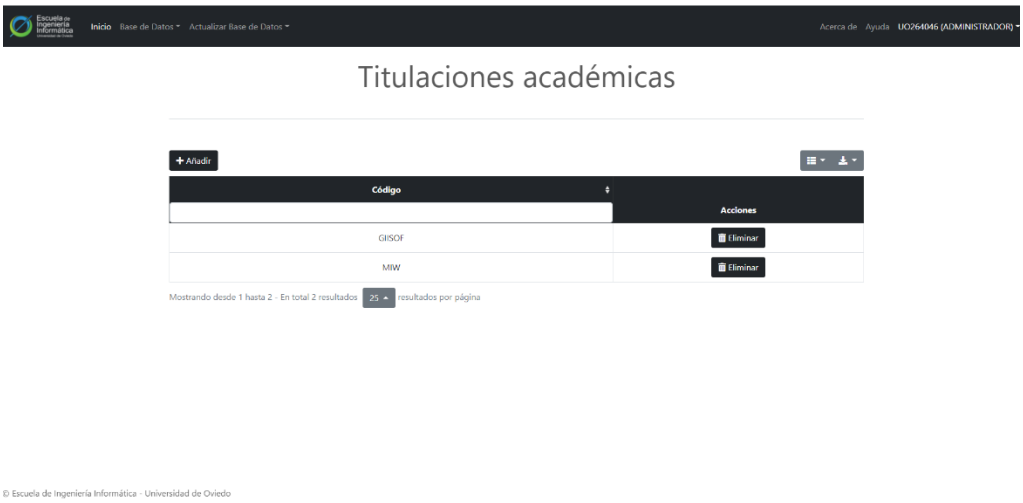


Figure 85: Interface Aspect Description of EIISERVER. Entities View

5.4.2.2.3 CREATE ENTITY VIEW

The views used for **creating new entities** follow the same interface design presented on Figure 86: Interface Aspect Description of EIISERVER. Create Entity View, where a form is displayed along with a button to cancel the operation and another to accept the operation and add the entity with the fields filled.

Note that fields as Course Code, Academic Degree Code, Group Code, and Students are **drop-down fields** (see Figure 87: Interface Aspect Description of EIISERVER. Create Entity View. Drop-downs) that contains a set with all the valid/accepted values **to prevent the user from inserting an incorrect value** and improve its experience.

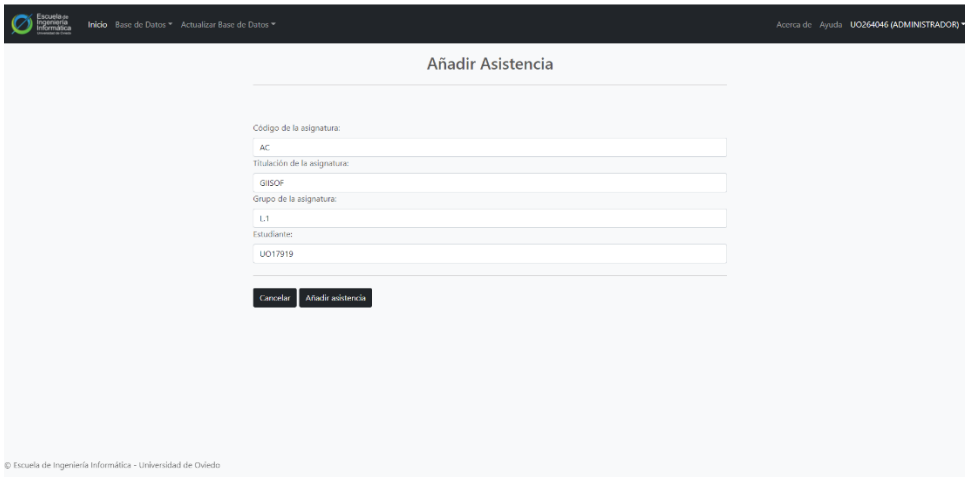


Figure 86: Interface Aspect Description of EIISERVER. Create Entity View

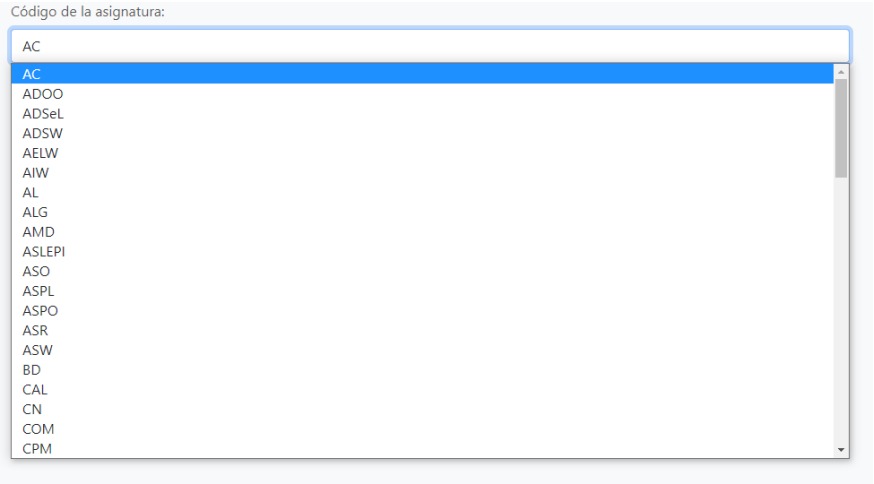


Figure 87: Interface Aspect Description of EIISERVER. Create Entity View. Drop-downs

5.4.2.2.4 EDIT ENTITY VIEW

The **Edit View** for all entities follows the same interface design described in the previous section: drop-down fields when necessary and cancel and accept button, among others. Figure 88: Interface Aspect Description of EIISERVER. Edit Entity View illustrates the view for editing a lecturer. The field "email" is disabled because it acts as the primary key.

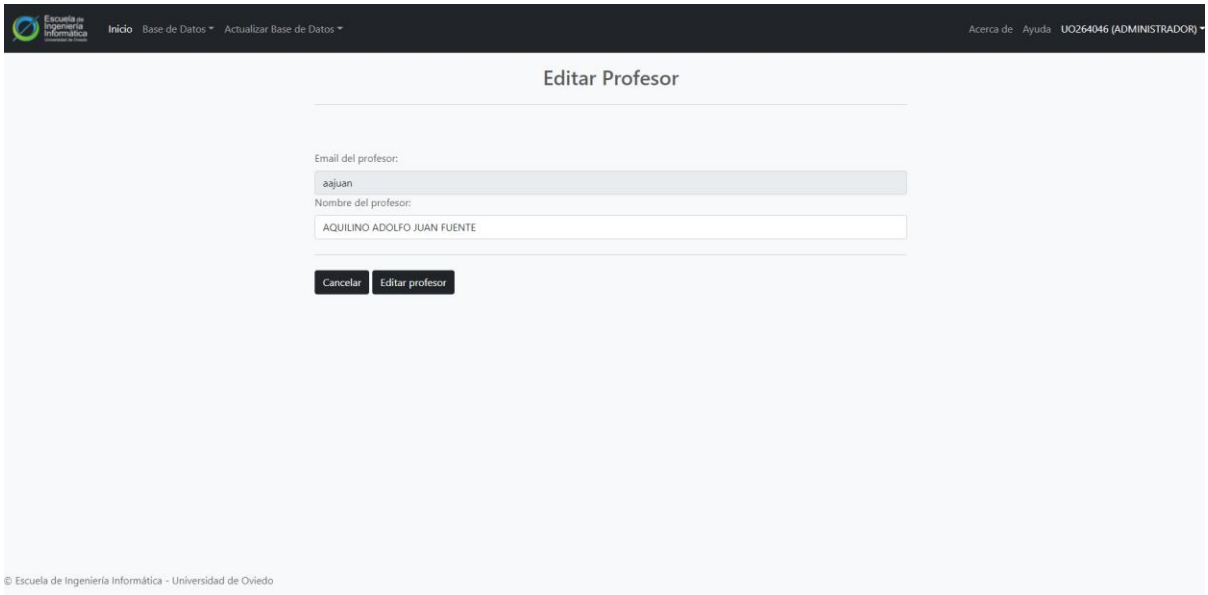


Figure 88: Interface Aspect Description of EIISERVER. Edit Entity View

5.4.2.2.5 UPDATE VIEW

The Update View is illustrated on Figure 89: Interface Aspect Description of EIISERVER. Update View. We present a table with **checkboxes** to allow users to select one or more rows from a set of academic degrees: **GIISOF** (Software Engineering Degree) and **MIW** (Master of Web Engineering).

When the update process is occurring, we notify the user by showing a loading spinner on the update button, placed below the table on Figure 89: Interface Aspect Description of EIISERVER.

Update View. In other words, we are applying the **solidity principle of usability** about **response times** by giving feedback to the user.

We also give feedback with the **logger** that is displayed on the same view (at the bottom), indicating how the process is going on and whether it succeeds or fails.

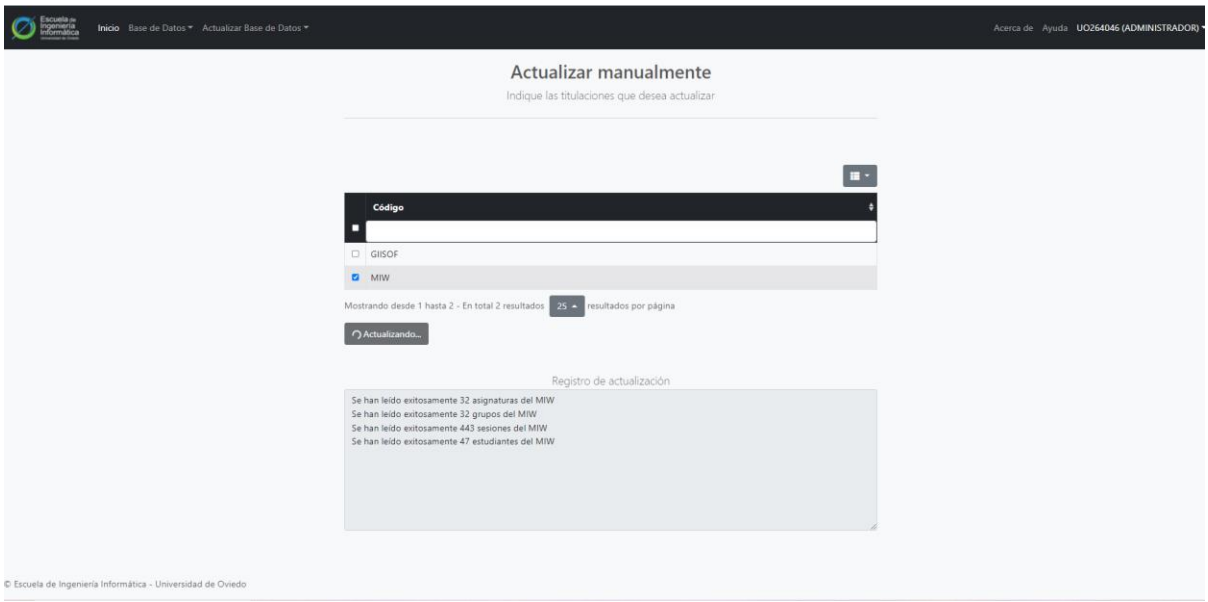


Figure 89: Interface Aspect Description of EIISERVER. Update View

5.4.2.2.6 UPDATE CONFIGURATION VIEW

Figure 90: Interface Aspect Description of EIISERVER. Update Configuration View illustrates the **Update Configuration View**. We make use of **cards** to display the three selectable update configurations: Daily Updates, Weekly Updates or Never Update.

Above the cards, we place an **status message** that shows if the last authomatic update triggered succeded or failed. We also wrap the current update configuration within a blue card, while unselected configuration options remain with a light grey color.

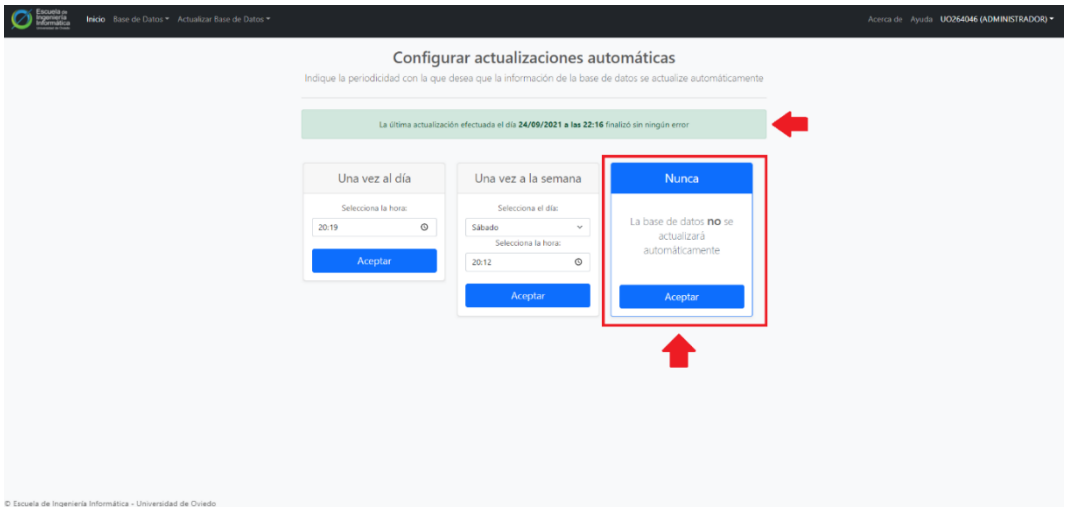


Figure 90: Interface Aspect Description of EIISERVER. Update Configuration View

5.4.2.2.7 SEND NOTIFICATION VIEW

On Figure 91: Interface Aspect Description of EIISERVER. Send Notification View, the **Send Notification to Academic Degrees View** is represented. All Send Notification Views follow the same interface design: when the sending process is occurring, we notify the user by showing a loading spinner on the send button, placed below the table on Figure 91: Interface Aspect Description of EIISERVER. Send Notification View, which contains a series of **checkboxes** to allow users to select one or more rows from a set of academic degrees: **GIISOF** (Software Engineering Degree) and **MIW** (Master of Web Engineering).

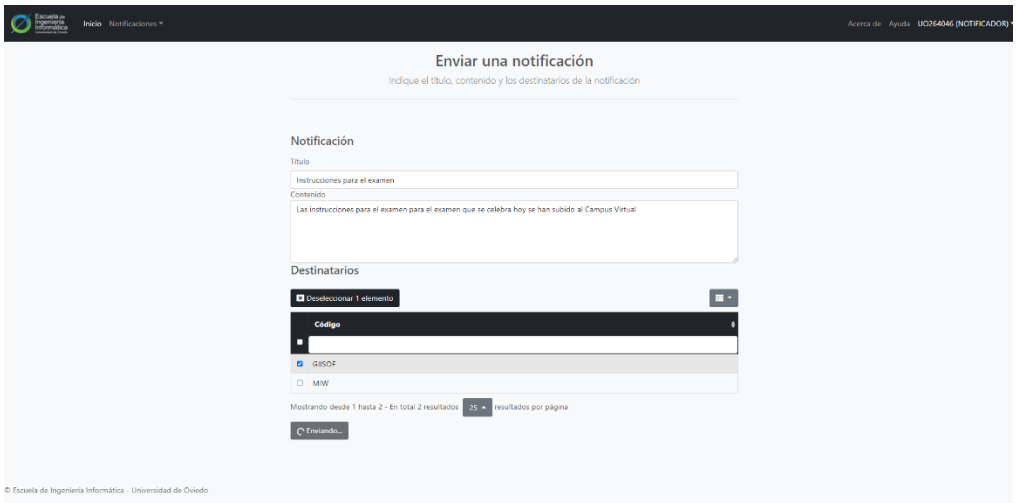


Figure 91: Interface Aspect Description of EIISERVER. Send Notification View

5.4.2.2.8 LINKS OF INTEREST VIEW

For the **Links of Interest View** (see Figure 92: Interface Aspect Description of EIISERVER. Links of Interest View) we use a **Markdown Editor**, with all the well-known features such as: putting a text in **Bold**, **Italic**, etcetera.

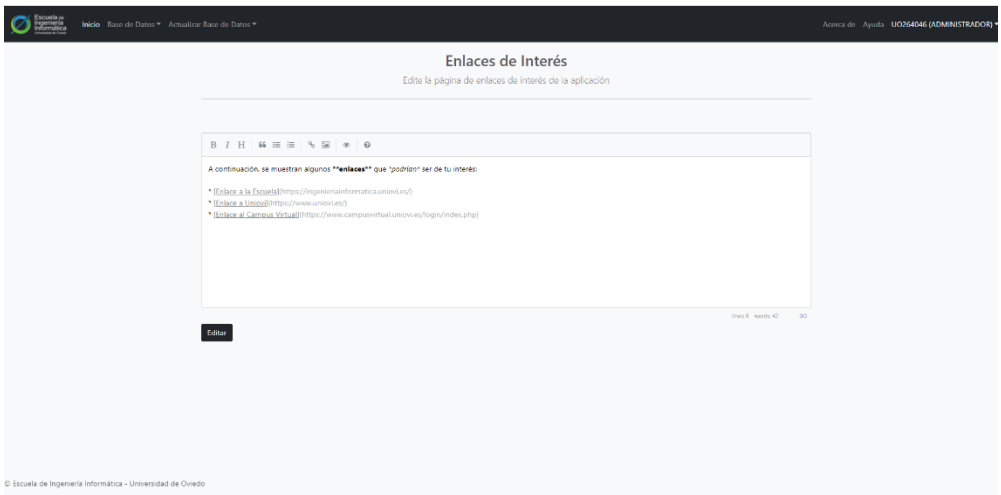


Figure 92: Interface Aspect Description of EIISERVER. Links of Interest View

5.4.2.2.9 HELP VIEW

The **Help View** is illustrated on Figure 93: Interface Aspect Description of EIISERVER. Help View. We have used **expansible panels** or accordions that can be collapsed/expanded depending on the needs of the user. The contents displayed on this view depend on the role of the authenticated user, showing help about the actions that can do with such role.

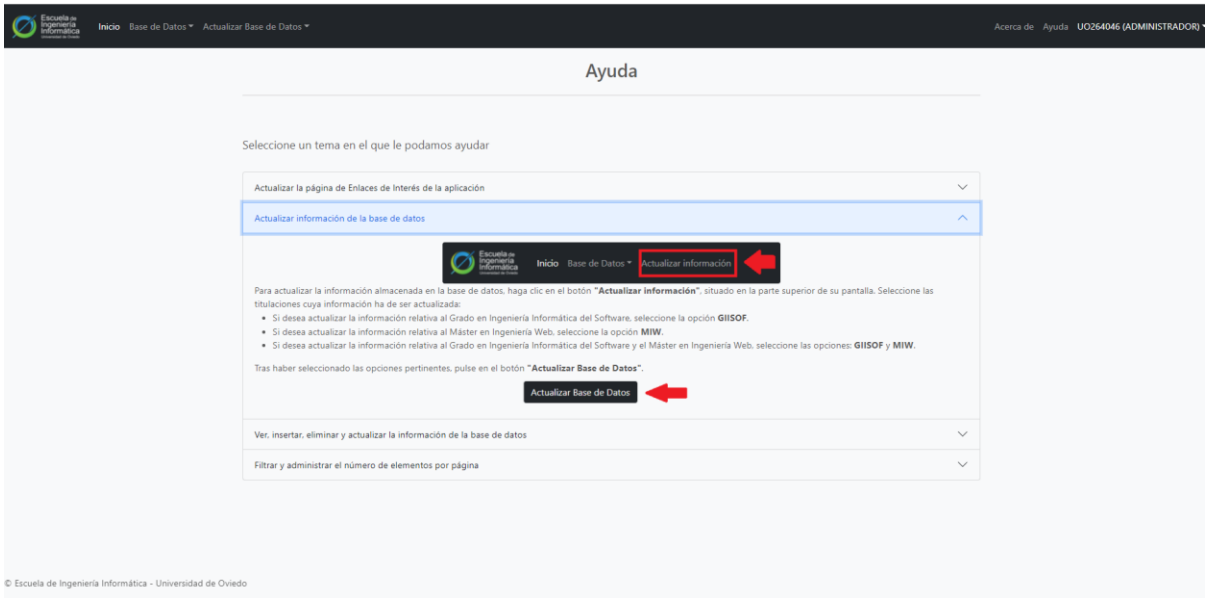


Figure 93: Interface Aspect Description of EIISERVER. Help View

5.4.2.2.10 ABOUT US VIEW

The **About Us View** is shown on Figure 94: Interface Aspect Description of EIISERVER. About Us View.

Figure 94: Interface Aspect Description of EIISERVER. About Us View

5.4.2.2.11 CHANGE ROLE VIEW

To change the role of the authenticated user, we make use of **radio buttons** to allow users to select one option from the set of available roles: ADMINISTRATOR or NOTIFIER in the case represented by Figure 95: Interface Aspect Description of EIISERVER. Change Role View.

Figure 95: Interface Aspect Description of EIISERVER. Change Role View

5.4.3 NAVIGATION DIAGRAM

The navigation diagrams presented on this section are built using **nodes** to represent **UI elements** and **arrows** to represent the **navigability** between the previously mentioned UI elements.

Next to arrows we can observe the **UI element that enables the navigability**. For instance, in Figure 96: Navigation Diagram. EIIAPP, we can navigate from "Home" to "Links of Interest" by clicking on the "Links Of Interest Card" displayed on "Home".

The navigation diagram of EIIAPP is illustrated in Figure 96: Navigation Diagram. EIIAPP. Note that "Navigation Drawer" is accessible from all UI elements represented in the diagram except Login.

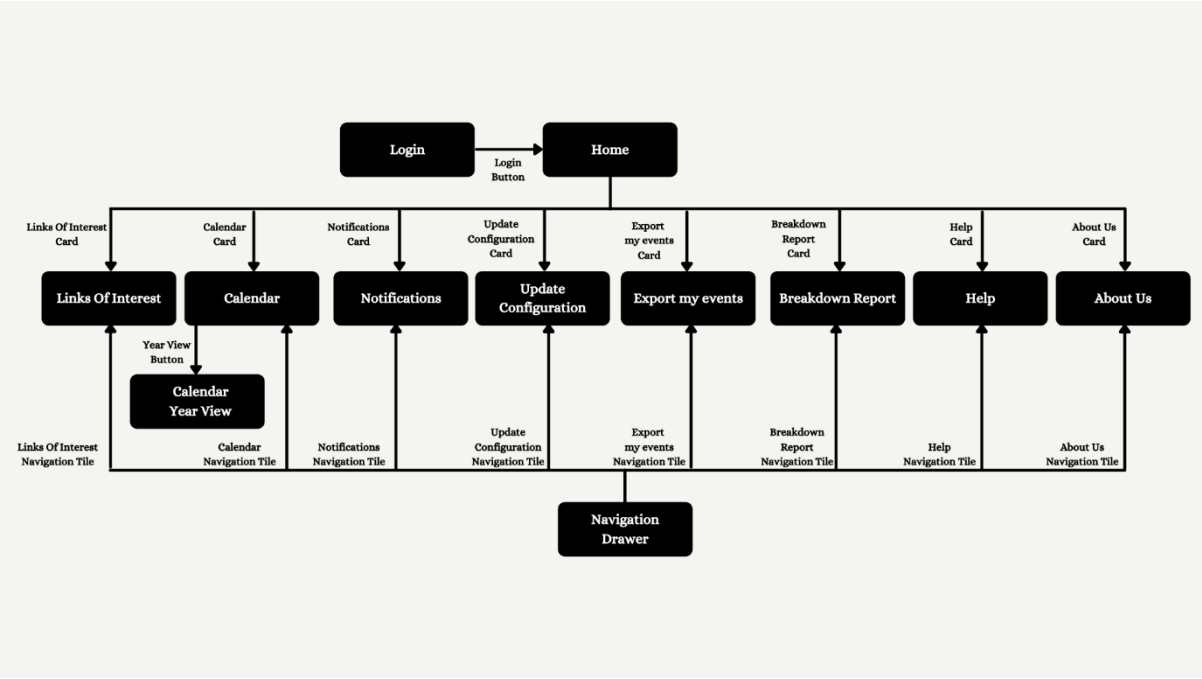


Figure 96: Navigation Diagram. EIIAPP

On the other hand, the navigation diagram of EIIServer is illustrated in Figure 97: Navigation Diagram. EIISERVER. Note that depending on the role of the authenticated user, it will see the options for ADMINISTRATORs or NOTIFIERs, as represented in the diagram.

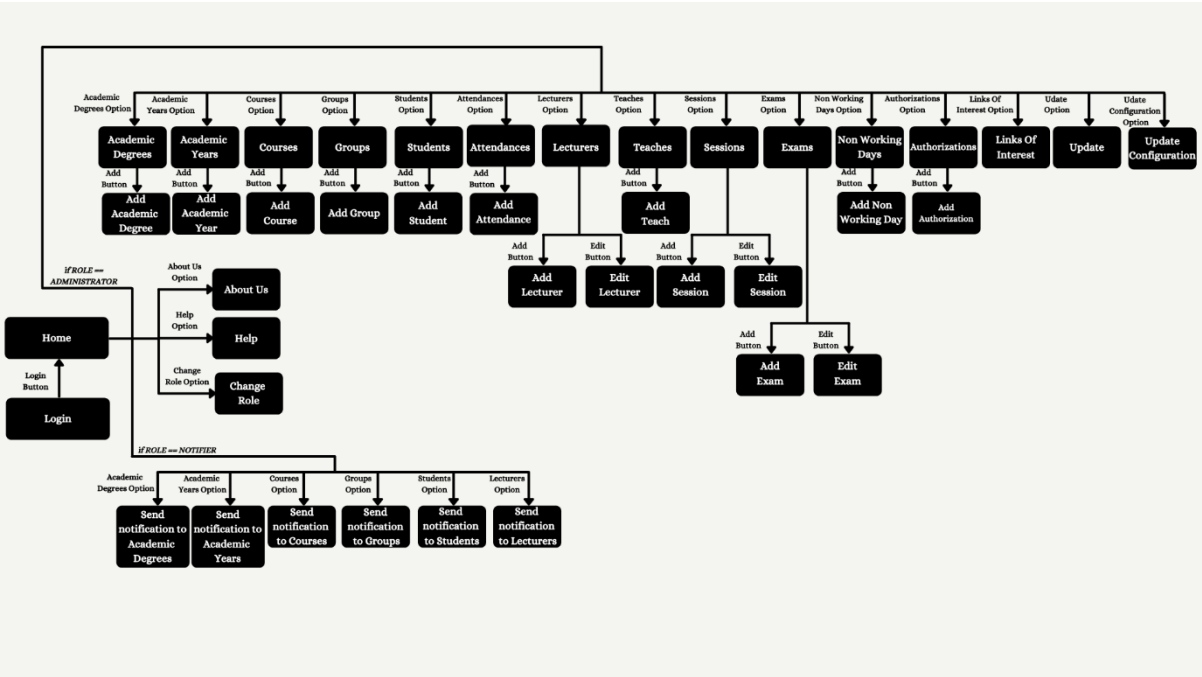


Figure 97: Navigation Diagram. EIISERVER

5.5 TESTING PLAN SPECIFICATION

To test EIISERVER and EIIAPP systems, the following tests will be carried out:

- **EIISERVER**
  - **Unit Tests**: self-documented in the code, automatized with **Jest** and parametrized with **Jest-Each**.
  - **Acceptance Tests**: self-documented using **Gherkin** along with **Cucumber**.
- **EIIAPP**
  - **Unit Tests**: self-documented in the code, automatized with **flutter_test**.
  - **Widget Tests**: self-documented in the code, automatized with **flutter_test**.
  - **Integration Tests**: self-documented in the code, automatized with **integration_test**.

6 DESIGN OF THE INFORMATION SYSTEM

In this section we will take a closer look at the design of both systems, EIISERVER and EIIAPP. All used **Design Patterns** will be described and compared with the Design Patterns included in the **well-known book of the Gang of Four** [13] following this structure:

- **Motivation**: What leaded to the use of the Design Pattern.
- **Design**: Design Diagrams.
- **Participants**: Comparison of the participants of the Design Pattern with the participants described by the **Gang of Four** [13].

6.1 EIISERVER

6.1.1 CROSS-CUTTING CONCEPTS

6.1.1.1 LOGGER

6.1.1.1.1 MOTIVATION

In **EIISERVER**, logging is used throughout the system (cross-cutting concept/concern). A logger that covered information about the whole system and dumped it to a file, was needed. As well as another one that displayed on screen how the update process was going on (see Figure 98: Design. Event Logger).

The main issue was discerning what information was desired to be logged for which logger. The file logger needed to register all information, while the other one just needed to log information related to the update process.

If EIIAPP communicates with EIISERVER via its API to get the events of a student, File Logger **shall log** something like "*GET /api/sessions/UO123456*", but it **should not be logged** by the Logger that displays how the update process is going on.

On the other hand, File Logger **shall dump all information to a file**, while the logger that shows how the update process is going on **shall send an event each time a message is logged** (an event is sent from **EIISERVER** to the web client that triggers the update process).

We would also like some flexibility.

- What would happen if we wanted to add another type of Logger?
- If we wanted to stop logging with File but not without Event Logger?

The design of the Logger Module is shown in Figure 99: Design. Logger. Observer Pattern. It uses an **Observer Pattern** that notifies all subscribed Logger each time a message is logged. For each Logger the method "getSubscribedTopics()" determined what type of information is Logger interested in logging.

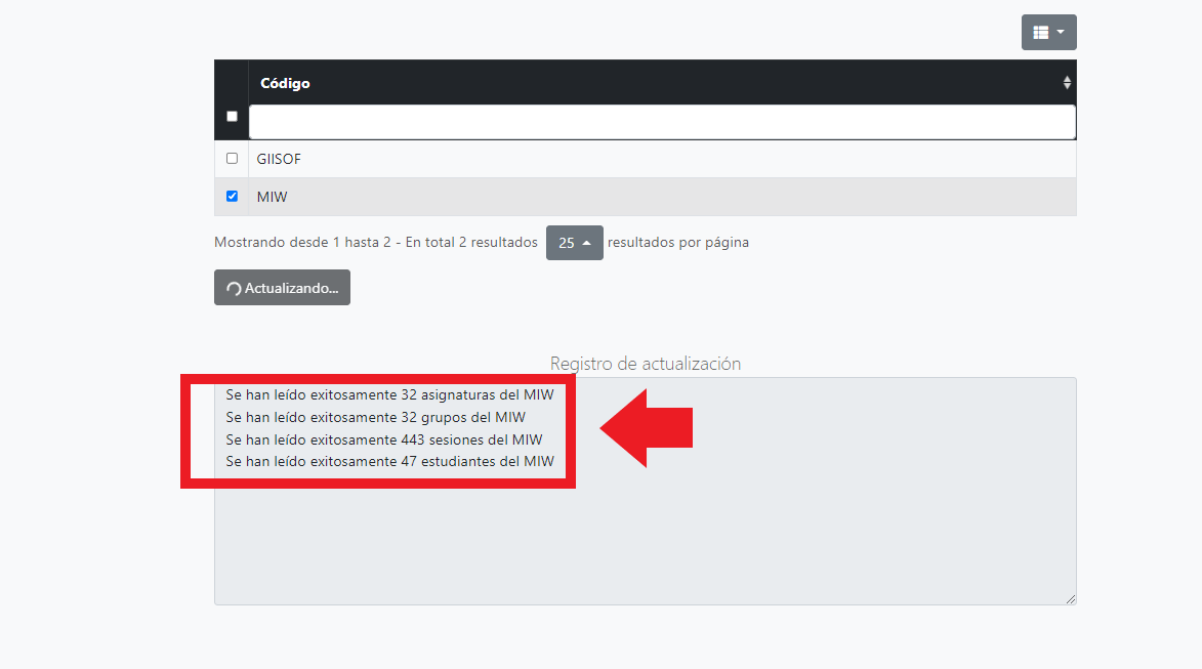


Figure 98: Design. Event Logger

6.1.1.1.2 DESIGN

The class diagram of Logger is shown in Figure 99: Design. Logger. Observer Pattern.

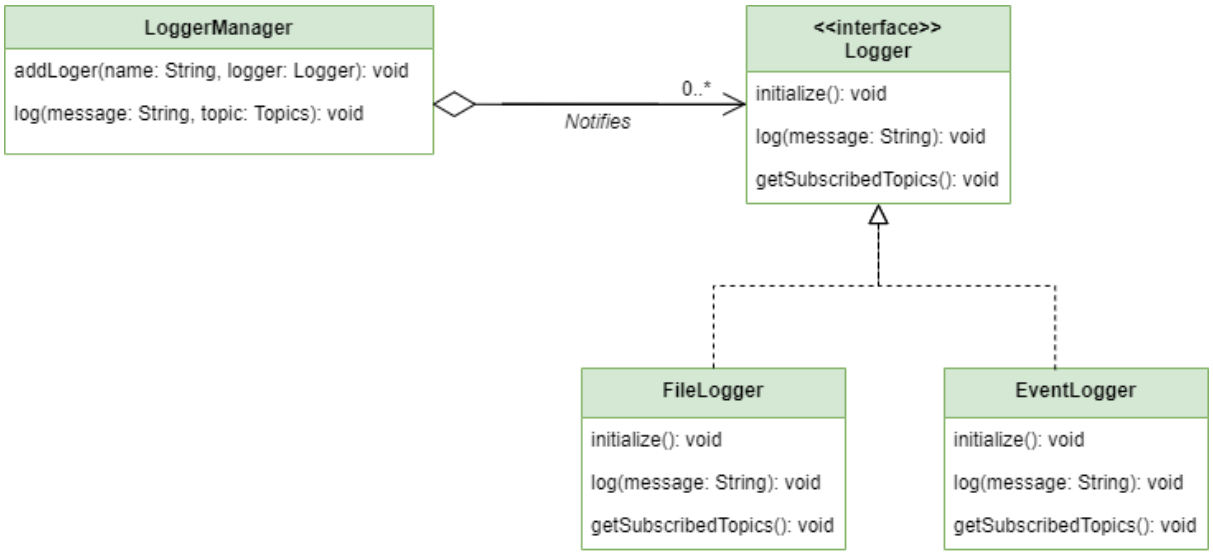


Figure 99: Design. Logger. Observer Pattern

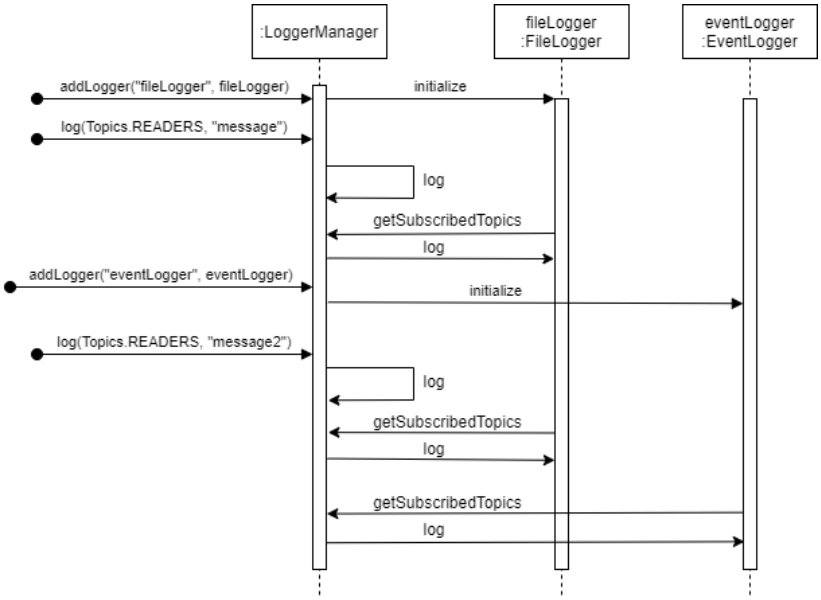


Figure 100: Design. Logger. Sequence Diagram

The sequence diagram of Logger is shown in Figure 100: Design. Logger. Sequence Diagram. Note that, in the case illustrated, fileLogger and eventLogger have "READERS" as subscribedTopic, i.e., "getSubscribeTopics" returns READERS for both Loggers indicating that they are interested in logging information related to READERS topic.

6.1.1.1.3 PARTICIPANTS

The participants of the pattern are listed below:

- **Subject**: LoggerManager.
- **Observer**: Logger.
- **ConcreteObserver**: FileLogger, EventLogger.

6.1.2 TRANSPORT LAYER

6.1.2.1 READERS

6.1.2.1.1 MOTIVATION

Readers module is responsible for **reading information from the School of Computer Science information sources** [3]–[5], **parsing** such information and **storing** it in the system.

The information sources are **diverse**: some of them are local **Excel** files, others are **HTML** documents, **Calendar** files, etc.

The readers are going to be triggered under two scenarios:

- When an ADMINISTRATOR updates the Database. It is **triggered immediately**.
- When an ADMINISTRATOR schedules an update selecting periodicity. It is **triggered when current time matches selected periodicity**.

6.1.2.1.2 DESIGN

The class diagram of Readers is shown in Figure 101: Design. Reader. Command Pattern.

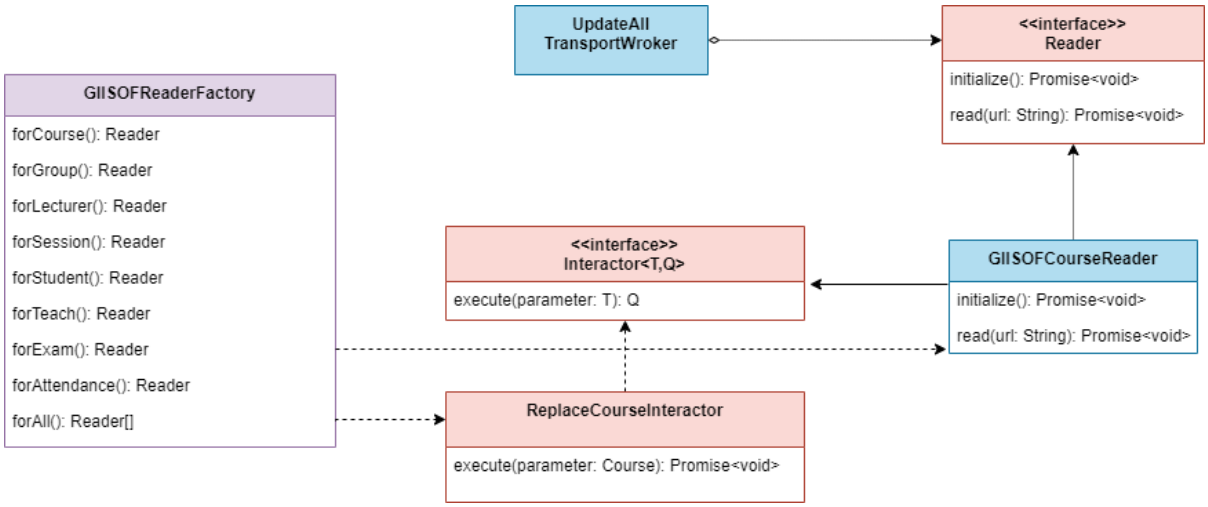


Figure 101: Design. Reader. Command Pattern

UpdateAllTransportWorker is triggered when an ADMINISTRATOR has scheduled an update and current time matches selected periodicity, it contains all Readers and executes them. GIISOFReaderFactory is responsible for creating Readers for GIISOF Academic Degree, associating the Reader with its correspondent interactor, ReplaceCourseInteractor.

To handle the variety of source types, the "load" feature is decoupled from the Reader itself creating a **strategy** that performs loading for each "source type" (see Figure 102: Design. Reader. Strategy Pattern).

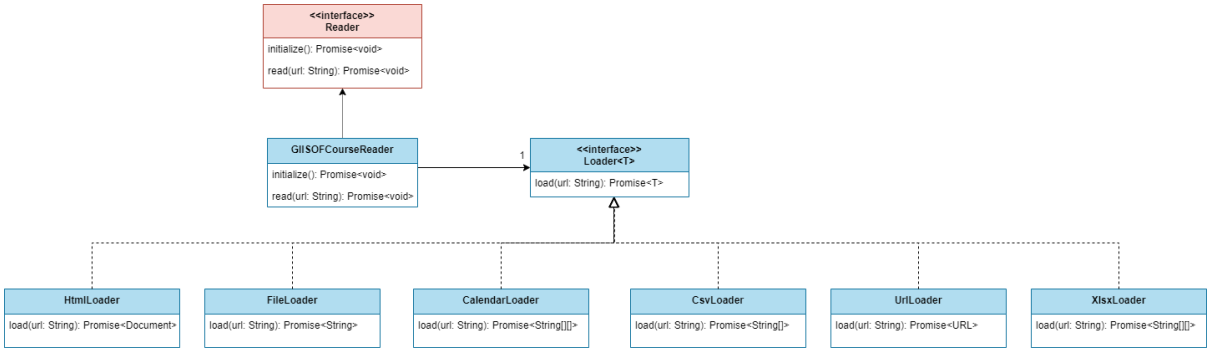


Figure 102: Design. Reader. Strategy Pattern

In some cases, like reading Students, the same process needs to be executed twice: one for each semester (1st semester and 2nd semester). On the other hand, the second semester only needs to be read when it has started.

In other words, imagine that second semester starts on 10/01/2022.

- If the update process is triggered on 03/01/2022, only students related to the first semester has to be read.

- If the update process is triggered on 11/01/2022, students related to the first and second semester has to be read. Each read process following the same business logic.

To accomplish that, we create another Reader: **Semester Reader**. It is modelled as a **Decorator** Pattern, as shown in Figure 103: Design. Reader. Decorator Pattern.

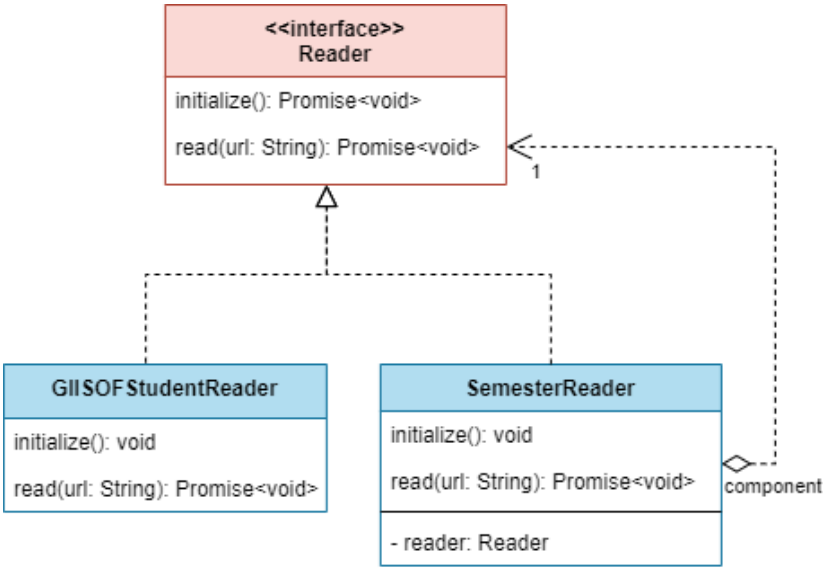


Figure 103: Design. Reader. Decorator Pattern

In other cases, such as in the reading of groups for the GIISOF Academic Degree, a **Decorator Pattern** is also used.

The same website is the source of all **groups and sessions** of the Software Engineering Degree. However, we would like to read all groups and sessions once (not repeat the same process) but store all groups first and then all sessions, since sessions depend on groups.

The same thing happens for students, the same website is the source of **students and attendances**, attendances depend on students, students need to be stored before attendances and we would like to load/read/parse the website once.

The solution is shown in Figure 104: Design. Reader. Decorator Pattern.

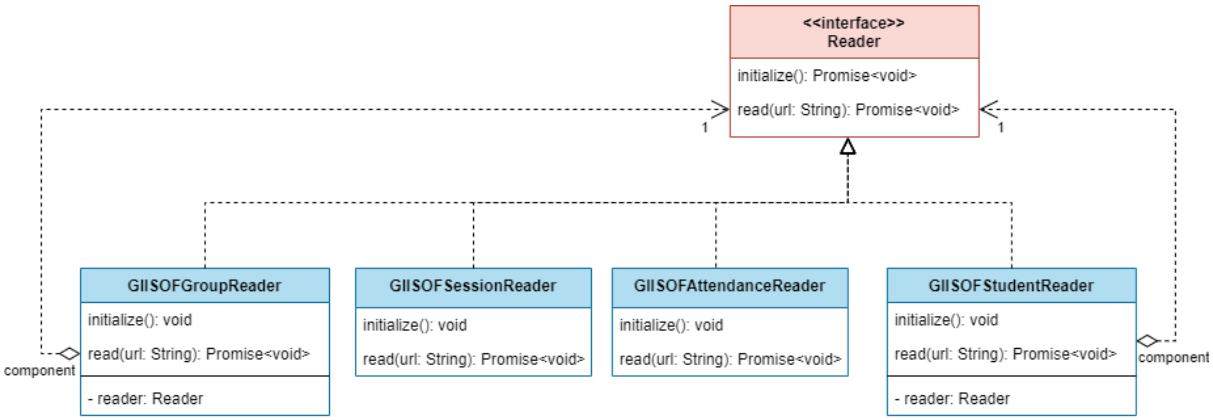


Figure 104: Design. Reader. Decorator Pattern

Finally, since reading process follows a **series of well-defined steps**:

1. Initialization
2. Reading
3. Storing

We used a **Template Method** for defining those steps execution in the established order (see Figure 105: Design. Reader. Template Method Pattern).

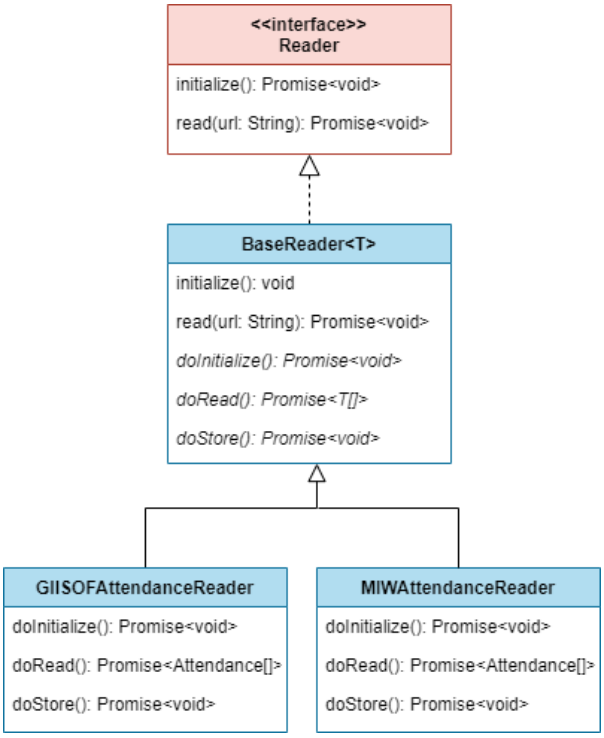


Figure 105: Design. Reader. Template Method Pattern

6.1.2.1.3 PARTICIPANTS

The participants of the Command Pattern are listed below:

- **Client**: GIISOFReaderFactory.
- **Invoker**: UpdateAllTransportWorker.
- **Command**: Reader.
- **ConcreteCommand**: GIISOFCourseReader.
- **Receiver**: ReplaceCourseInteractor.

The participants of the Strategy Pattern are listed below:

- **Context**: GIISOFCourseReader.
- **Strategy**: Loader.
- **ConcreteStrategies**: HtmlLoader, FielLoader, CalendarLoader, CsvLoader, UrlLoader, XlslxLoader.

The participants of the Decorator Pattern for Semester Reader are listed below:

- **Component**: Reader.
- **ConcreteComponent**: GIISOFStudentReader.

- **ConcreteDecorator**: SemesterReader.

The participants of the Decorator Pattern for Group and Student Readers are listed below:

- **Component**: Reader.
- **ConcreteComponents**: GIISOFSessionReader, GIISOFAttendaceReader.
- **ConcreteDecorators**: GIISOFGroupReader, GIISOFStudentReader.

The participants of the Template Method Pattern are listed below:

- **AbstractClass**: BaseReader.
- **ConcreteClasses**: GIISOFAttendanceReader, MIWAttendanceReader.

6.1.2.2 WORKERS

6.1.2.2.1 MOTIVATION

For heavy-computational tasks such as Sending PUSH Notifications or Updating the Database, we use thread workers to lighten the workload of EIISERVER. These are part of the Transport Layer and encapsulated inside strategies, as shown in Figure 106: Design. Workers. Strategy Pattern, used by Route Controllers of Express.

6.1.2.2.2 DESIGN

The class diagram of TransportWorker is shown in Figure 106: Design. Workers. Strategy Pattern.

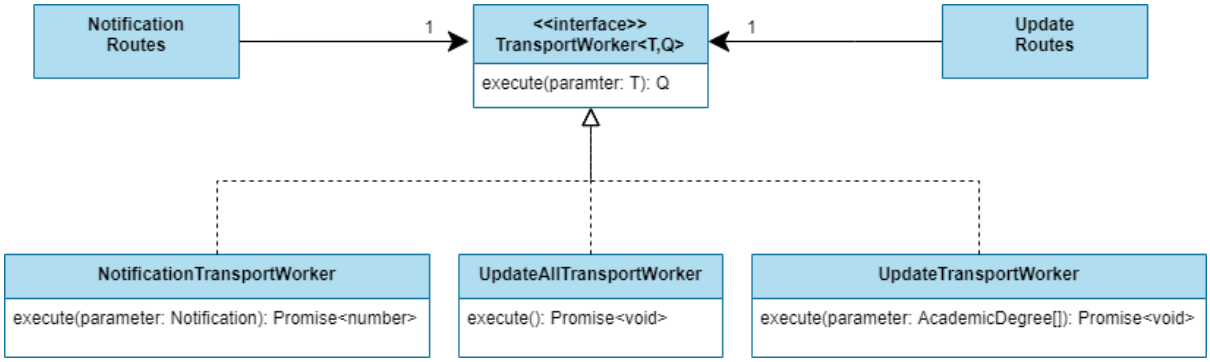


Figure 106: Design. Workers. Strategy Pattern

Another remarkable factor to point out is that Transport Workers enables the communication between Main Thread and Worker Thread and vice versa.

An example of communication can be found in UpdateTransportWorker which communicates with MainThread each time a message is logged. MainThread, when communicated, sends an event to the web client that triggered the update process with the log message (as explained in Logger). Note that a WorkerThread cannot send an event to the web client, since it is placed in another thread.

This is represented in Figure 107: Design. Workers. Sequence Diagram.

1. A user makes a POST request to update the database.
2. Update Routes Controller executes UpdateTransportWorker.
3. UpdateTransportWorker spawns a Thread that performs the heavy-computational work (updating the database by executing all GIISOFReaders).

4. When GIISOFCourseReader has read all courses, Thread posts a message to its parent (UpdateTransportWorker).
5. UpdateTransportWorker handles the message by logging it using LoggerManager.
6. LoggerManager will send an Event to web client (see Logger).

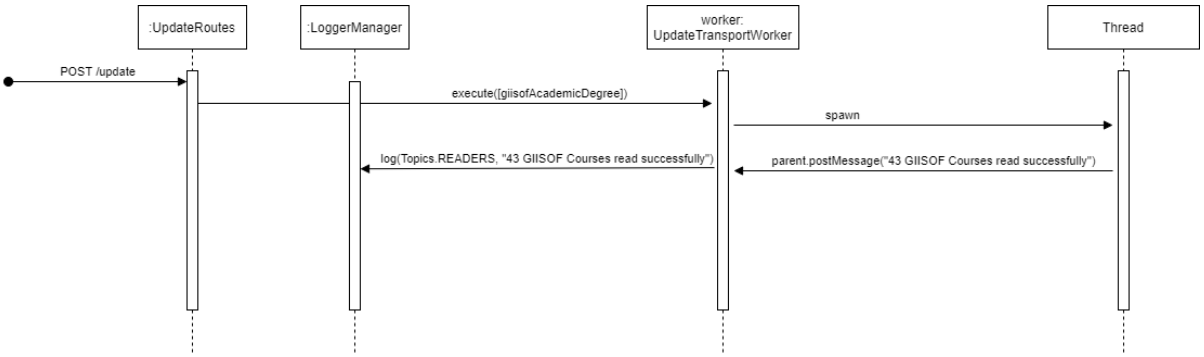


Figure 107: Design. Workers. Sequence Diagram

6.1.2.2.3 PARTICIPANTS

The participants of the Strategy Pattern are listed below:

- **Context**: Notification Routes, Update Routes.
- **Strategy**: TransportWorker.
- **ConcreteStrategies**: NotificationTransportWorker, UpdateTransportWorker, UpdateAllTransportWorker.

6.1.2.3 ERROR HANDLERS

6.1.2.3.1 MOTIVATION

Error handling, following **Node Best Practices** [14], was centralized in Transport Layer. Core Layer as well as Datasource Layer **throw Application Errors depending on no error handling strategy or method**.

Thrown Errors are caught and handled on Route Controllers. Two types of handling are contemplated:

- **Application Error Handling**: when an error occurs, the user authenticated on the web client is redirected to an URL and an error is shown.
- **JSON Error Handling**: when an error occurs, an Error is return in JSON format. Useful for the exposed REST API of EIISERVER.

6.1.2.3.2 DESIGN

The class diagram of ErrorHandler is shown in Figure 108: Design. Error Handlers. Strategy Pattern.

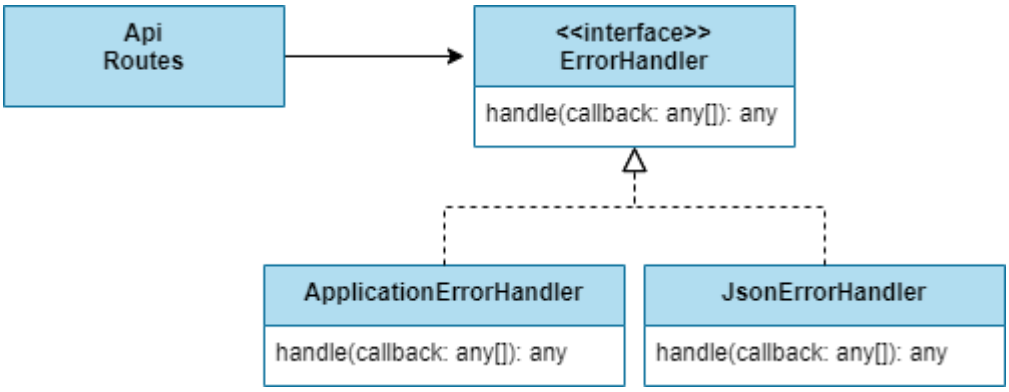


Figure 108: Design. Error Handlers. Strategy Pattern

ErrorHandler, handles all errors thrown within the execution of callbacks. Callbacks are meant to be middlewares used within API Routes.

6.1.2.3.3 PARTICIPANTS

The participants of the Strategy Pattern are listed below:

- **Context**: Api Routes.
- **Strategy**: ErrorHandler.
- **ConcreteStrategies**: ApplicationErrorHandler, JsonErrorHandler.

6.1.3 CORE LAYER

6.1.3.1 INTERACTORS

6.1.3.1.1 MOTIVATION

Core Layer (Interactors) is responsible for connecting **Transport Layer** with **Datasource Layer** executing some business logic. We would like to **decouple the invoker**, Routes of Transport Layer, **from the receiver**, Datasources of Datasource Layer.

The interactors will be executed, for instance:

- Whenever a user authenticated on EIIAPP makes a request to EIISERVER.
- Whenever a user of EIISERVER makes a request.
- Whenever a scheduled process is executed.

6.1.3.1.2 DESIGN

The class diagram of Interactors is shown in Figure 109: Design. Interactors. Command Pattern.

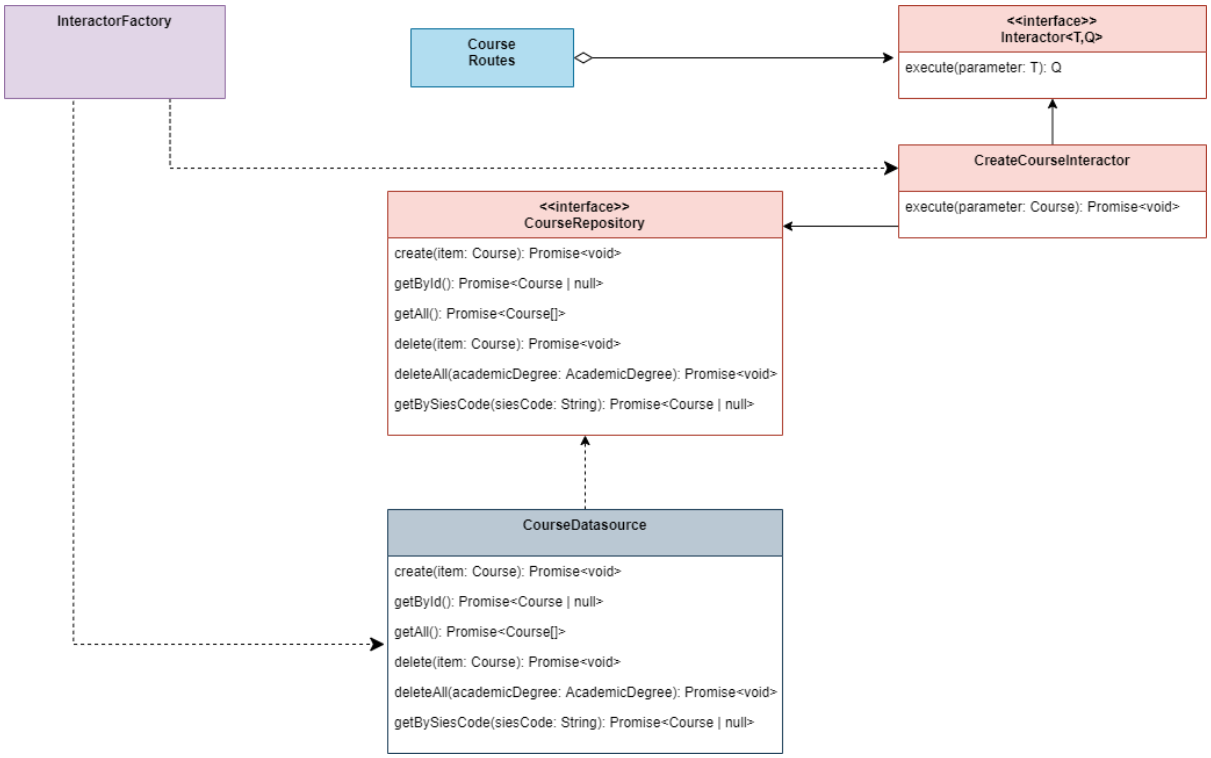


Figure 109: Design. Interactors. Command Pattern

Command Pattern Collaborations can be seen on Figure 110: Design. Interactors. Sequence Diagram. CourseRoutes stores the Command that is retrieved from Interactor Factory. When a user makes a POST request to add a new Course, stored Command is executed.

Note the **time break** between storing the Command (forCreateCourseInteractor) and executing it (execute).

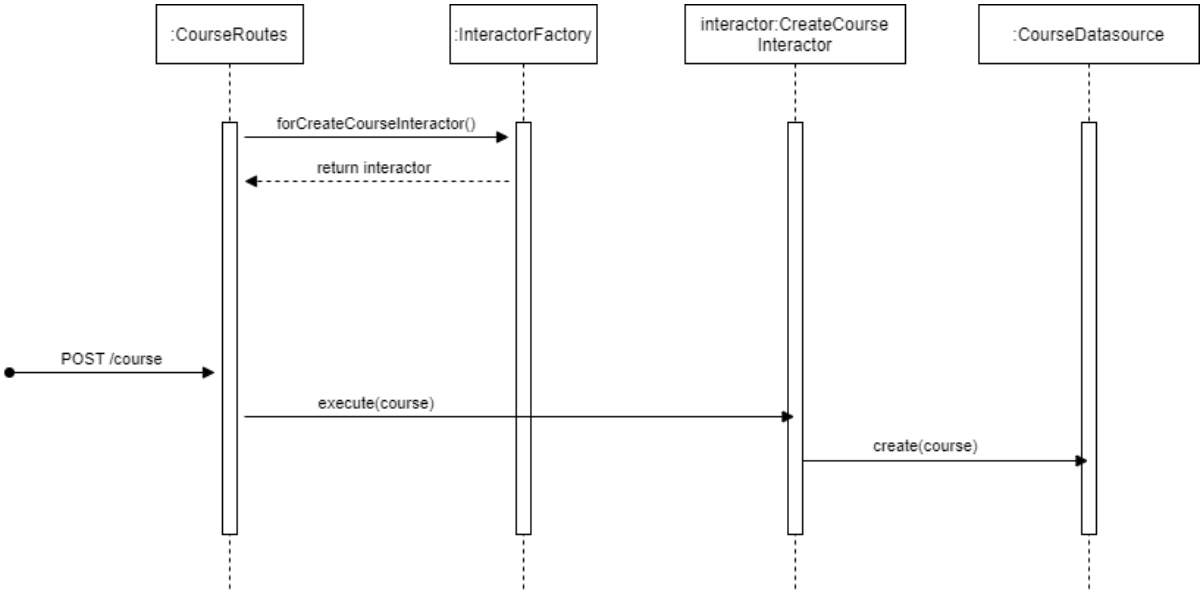


Figure 110: Design. Interactors. Sequence Diagram

6.1.3.1.3 PARTICIPANTS

The participants of the Command Pattern are listed below:

- **Client**: InteractorFactory.
- **Invoker**: CourseRoutes.
- **Command**: Interactor.
- **ConcreteCommand**: CreateCourseInteractor.
- **Receiver**: CourseDatasource.

6.1.4 DATASOURCE LAYER

As shown in Figure 120: Class Diagram. EIISERVER. **Datasources** are decoupled from Core Layer by implementing Interfaces: Repositories. **Each Datasource implements a Repository**.

6.1.4.1 UPDATE DATASOURCE

6.1.4.1.1 MOTIVATION

Update Datasource, schedules the updates of the database attending to one of these periodicities: **Daily** (the update process is triggered once a day), **Weekly** (the update process is triggered once a week), **Never** (the update process is never triggered).

6.1.4.1.2 DESIGN

To tackle this problem, a Strategy Pattern was used (see Figure 111: Design. Update Datasource. Strategy Pattern).

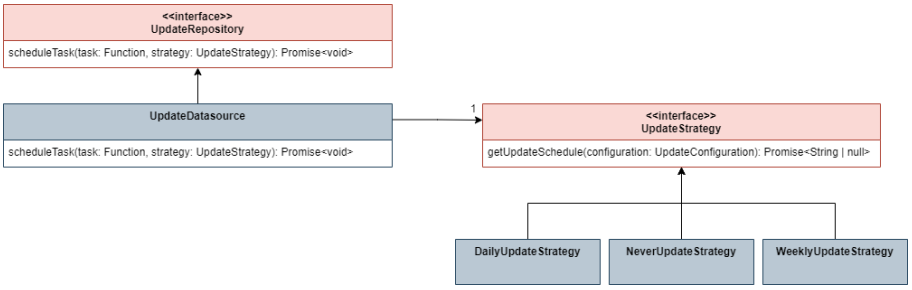


Figure 111: Design. Update Datasource. Strategy Pattern

6.1.4.1.3 PARTICIPANTS

The participants of the Strategy Pattern are listed below:

- **Context**: UpdateDatasource.
- **Strategy**: UpdateStrategy.
- **ConcreteStrategies**: DailyUpdateStrategy, WeeklyUpdateStrategy, NeverUpdateStrategy.

6.1.5 EXTENSION METHODS

We have used **extension methods** in EIISERVER to avoid using "Utility Classes" (static classes with a bunch of non-related methods). As Figure 112: Design. EIISERVER Extension Methods shows, those utility methods are arranged in multiple interfaces that are used throughout the entire system.

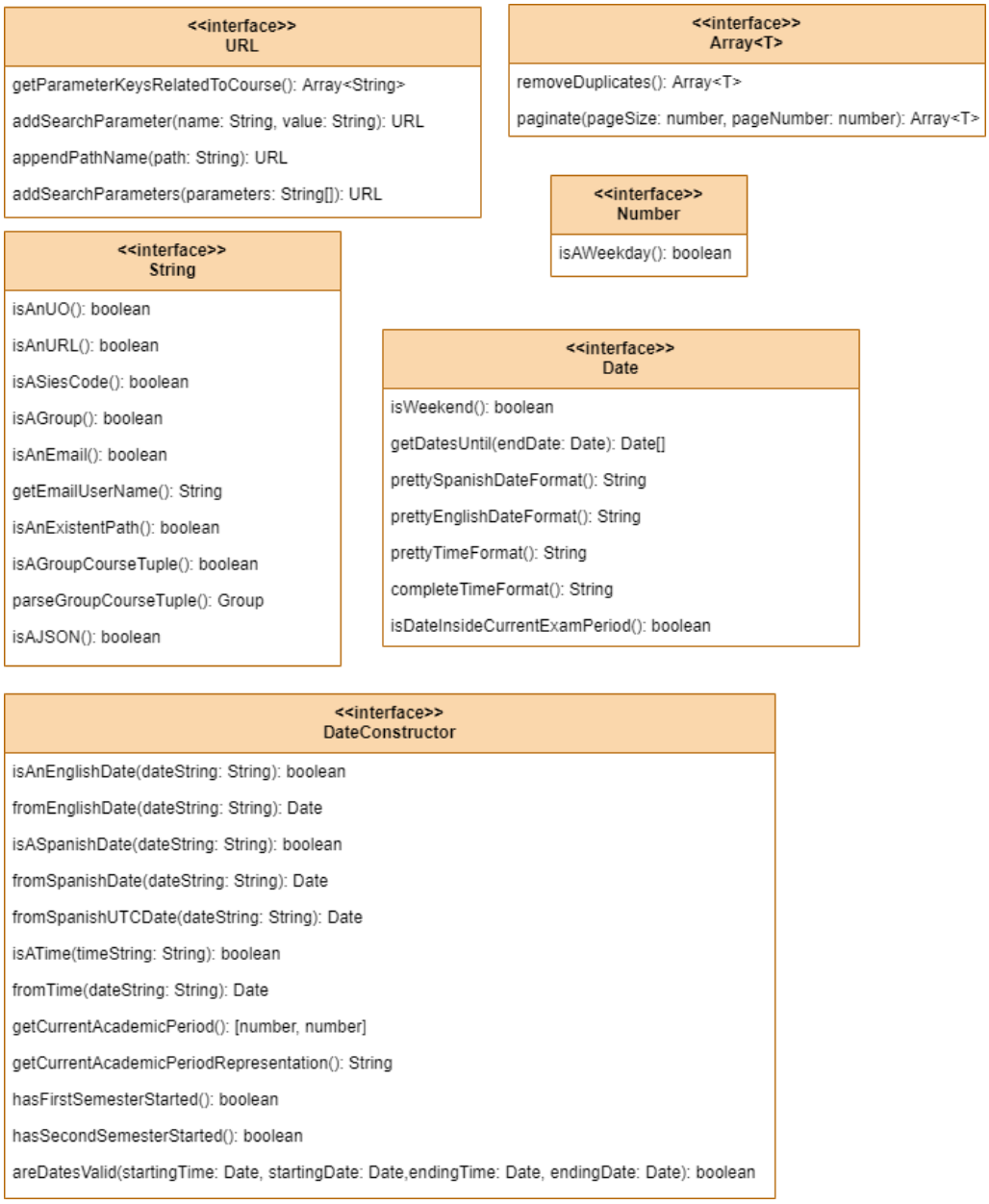


Figure 112: Design. EIISERVER Extension Methods

6.1.6 FACTORIES

6.1.6.1 MOTIVATION

Factories (the module in EIISERVER system) are responsible for creating concrete objects and exposing them by their interfaces.

ErrorHandlerFactory, RepositoryStrategyFactory, LoggerFactory, LoaderFactory, InteractorFactory, RepositoryFactory and WorkerFactory are static classes (**not Design Patterns**).

On the other hand, regarding Readers module, both Academic Degrees have their collection of Readers. As Table 31: Design. GIISOF Readers vs MIW Readers shows, each Academic Degree has

the same number and type of Readers, but each one, implements a different reading process: GIISOFCourseReader parses a JSON file, while MIWCourseReaders parses an Excel file.

GIISOF READERS	MIW READERS
GIISOFAttendanceReader	MIWAttendanceReader
GIISOFCourseReader	MIWCourseReader
GIISOFGroupReader	MIWGroupReader
GIISOFLecturerReader	MIWLecturerReader
GIISOFSessionReader	MIWSessionReader
GIISOFStudentReader	MIWStudentReader
GIISOFTeachReader	MIWTeachReader
GIISOFExamReader	MIWExamReader

Table 31: Design. GIISOF Readers vs MIW Readers

6.1.6.2 DESIGN

To tackle this problem, we have used a **Factory Method**, as Table 32: Design. Reader Factory. Factory Method shows. The method that is named "forAll" returns all Readers that the factory creates in an Array (used by the Update Database module that updates one or more Academic Degrees).

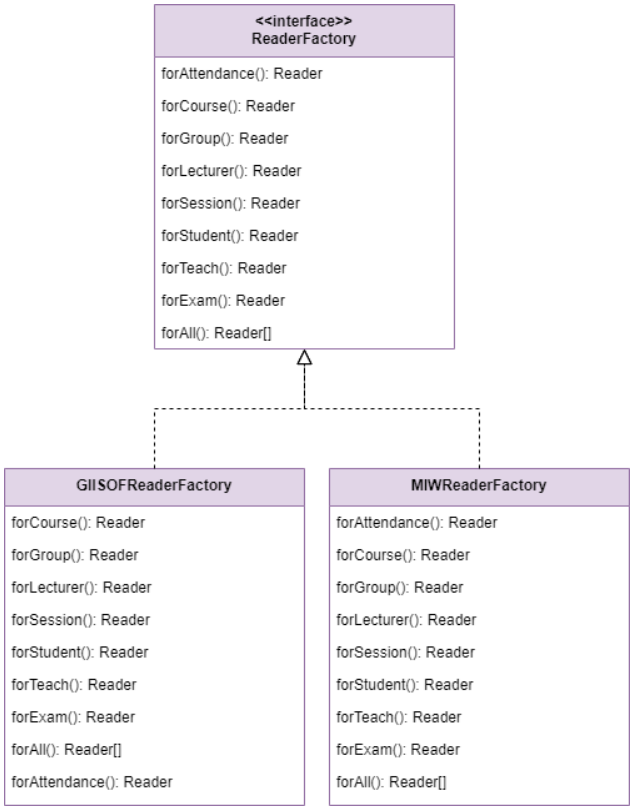


Table 32: Design. Reader Factory. Factory Method

6.1.6.3 PARTICIPANTS

The participants of the Factory Method Pattern are listed below:

- **Creator**: ReaderFactory.
- **ConcreteCreators**: GIISOFReaderFactory, MIWReaderFactory.
- **Product**: Reader.

- **ConcreteProducts**: GIISOFAttendanceReader, GIISOFCourseReader, GIISOFGroupReader, GIISOFLecturerReader, GIISOFSessionReader, GIISOFStudentReader, GIISOFTeachReader, GIISOFExamReader, MIWAttendanceReader, MIWCourseReader, MIWGroupReader, MIWLecturerReader, MIWSessionReader, MIWStudentReader, MIWTeachReader, MIWExamReader.

6.2 EIIAPP

6.2.1 CORE

6.2.1.1 SERVICES

6.2.1.1.1 MOTIVATION

Services in EIIAPP represent the execution of an action. For instance, the exportation of an event to a Calendar, the Update of the Database or a Request made to EIISERVER. All Services follow the contract indicated by "Service".

Nevertheless, most services execute a network request that follow a series of steps:

1. Make a request to a route
2. Checking response status
3. Decoding response if status is OK

On the other hand, we would like to let a user update the information stored on the database attending to one of these periodicities:

- Daily (the update process is triggered once a day).
- Weekly (the update process is triggered once a week).
- Never (the update process is never triggered).

The logic of the update process is the same for all periodicities.

6.2.1.1.2 DESIGN

This was accomplished with the creation of a BaseApiService modelled as a **Template Method**, as Figure 113: Design. Services shows.

In addition, getting the sessions and exams followed the same steps mentioned before as well as well other ones for decoding the response:

1. Check response decoded is iterable
2. Initialize repository
3. Decode events that appears on respone
4. Insert events in repository

That is why the same pattern is applied again creating another **Template Method** called "BaseEventsApiService" that defines the steps listed before.

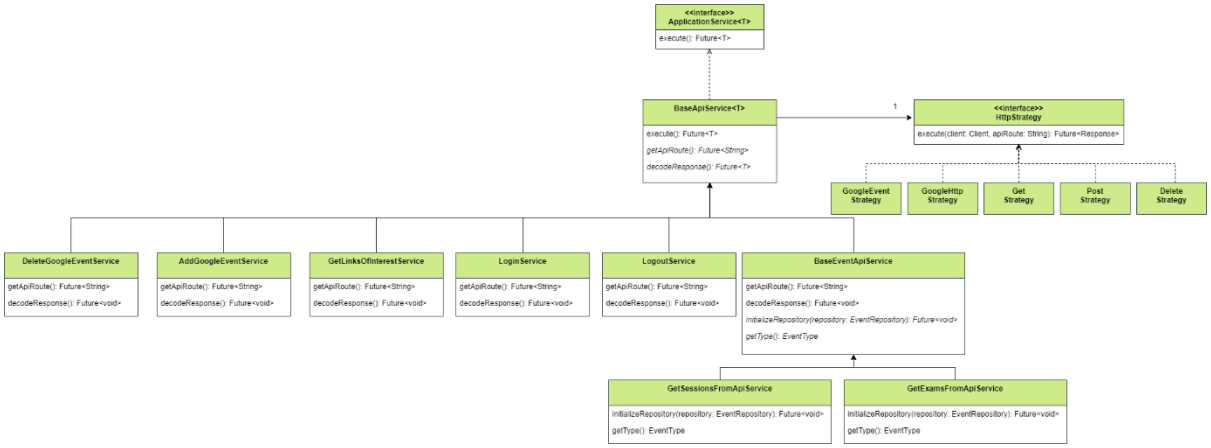


Figure 113: Design. Services

The request method is also decoupled from the Service, creating a **strategy** ("HttpStrategy") that performs each request (see Figure 113: Design. Services).

On the other hand, a Template Method is used to model all the update variants (see Figure 114: Design. Services. Template Method).

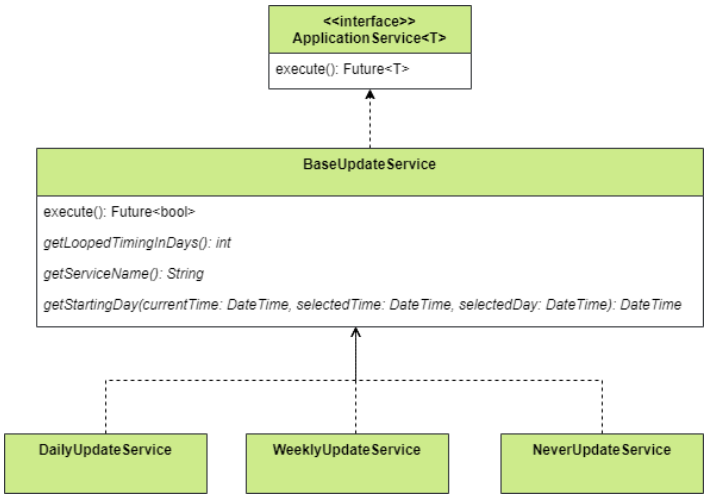


Figure 114: Design. Services. Template Method

6.2.1.1.3 PATICIPANTS

The participants of the Template Method Pattern are listed below:

- **AbstractClass**: BaseApiService, as well as BaseEventApiService.
- **ConcreteClasses**: DeleteGoogleEventService, AddGoogleEventService, GetLinksOfInterestService, LoginService, LogoutService, as well as GetSessionsFromApiService and GerExamsFromApiService.

The participants of the Strategy Pattern are listed below:

- **Context**: BaseApiService.
- **Strategy**: HttpStrategy.
- **ConcreteStrategies**: GoogleEventStrategy, GoogleHttpStrategy, GetStrategy, PostStrategy, DeleteStrategy.

The participants of the Template Method Pattern (for the update process) are listed below:

- **AbstractClass**: BaseUpdateService.
- **ConcreteClasses**: DailyUpdateService, WeeklyUpdateService and NeverUpdateService.

6.2.2 DATASOURCES

6.2.2.1 MOTIVATION

As shown in Figure 121: Class Diagram. EIIAPP. **Datasources** are decoupled from Core Layer by implementing Interfaces: Repositories. **Each Datasource implements a Repository**.

Nevertheless, we would like two types of Datasources: one "synchronous" and another "asynchronous".

- The **asynchronous** one, will open and close the connection with the database within each transaction. In other words, on each transaction the database will be dumped to disk.
- The **synchronous** one, will keep the connection alive throughout the lifecycle of the EIIAPP instance. **Database will only be dumped to disk when the application lifecycle has reached the Detached/Killed state**.

The **synchronous datasource** will be used for notifications **to lighten the workload** of EIIAPP: getting notifications, storing a notification when it has arrived and marking all notifications as read **without closing the connection** (dumping to disk). The datasource will keep the notifications on memory and whenever the state of a notification state change, all listeners to the Notification Datasource will be notified using Provider (as explained in Definition of the technological architecture). The **asynchronous datasource** will be used for the rest of Datasources.

6.2.2.2 DESIGN

The class diagram of Datasource Module is illustrated, thereby, in Figure 115: Design. EIIAPP. Datasources.

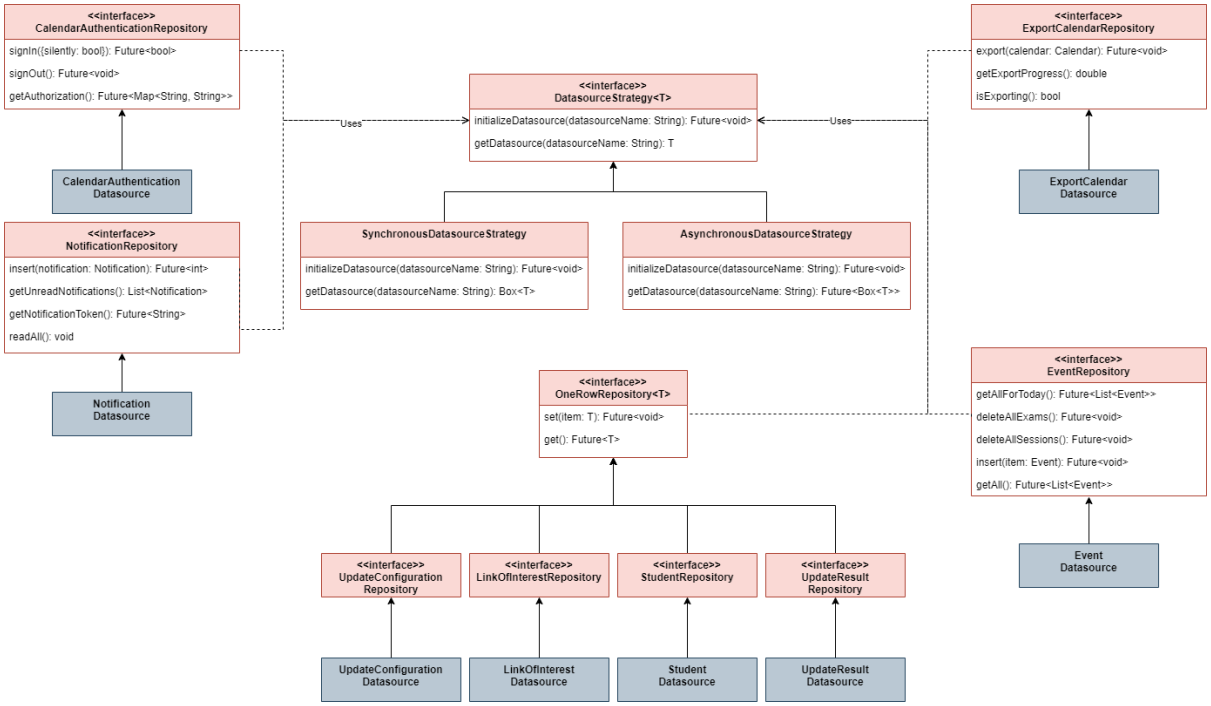


Figure 115: Design. EIIAPP. Datasources

6.2.2.3 PARTICIPANTS

The participants of the Strategy Pattern are listed below:

- **Context**: Each Datasource.
- **Strategy**: DatasourceStrategy.
- **ConcreteStrategies**: SynchronousDatasourceStrategy, AsynchronousDatasourceStrategy.

6.2.3 UI

6.2.3.1 WIDGETS

Following **Flutter Guidelines** [15], Views of EIIAPP system are compound of "widgets", immutable descriptions of part of User Interfaces (see Figure 5: Selection of the Architecture. EIIAPP. Provider Architecture).

Style Guidelines state that Views should add functionality by composing widgets (not inheriting), **acting as full-fledged Decorators**. All EIIAPP widgets extend "Widget" class and are responsible for rendering a piece of the UI with its "build" method (see Figure 116: Design. EIIAPP. Widgets).

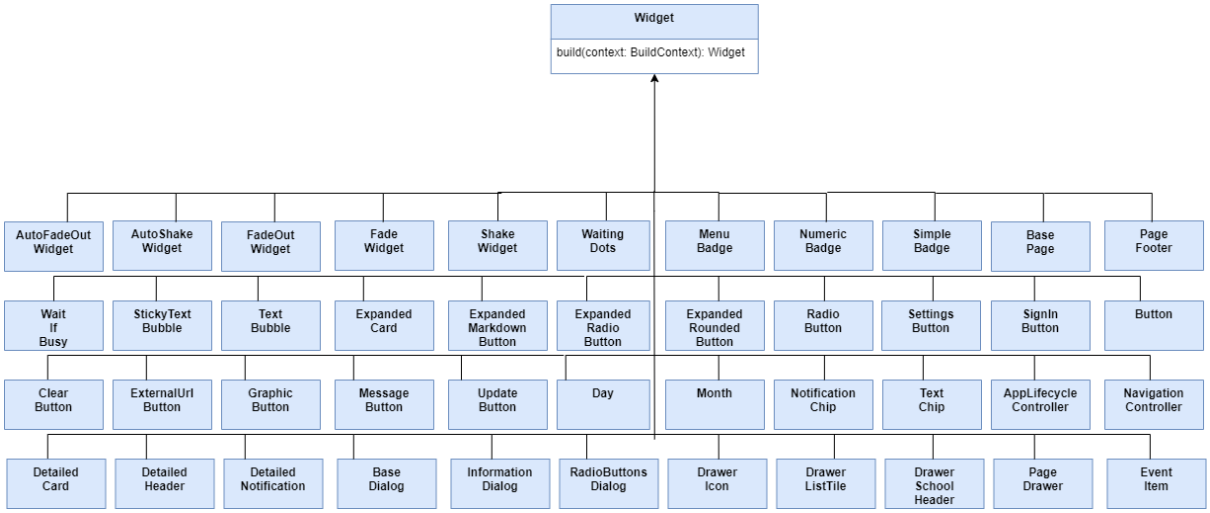


Figure 116: Design. EIIAPP. Widgets

6.2.3.2 VIEWS

6.2.3.2.1 MOTIVATION

Just like Widgets Module, Views also inherit "Widget" class (see Figure 117: Design. EIIAPP. Views). Each View is associated to one ViewModel that **consumes**. Each time a ViewModel has made a change, the associated View is notified.

On the other hand, **ExportCalendarView** must behave differently under these circumstances:

- The user is not authenticated with Google
- The user is already authenticated
- The user has selected a calendar and exportation to such calendar is being prepared
- The exportation is in process

6.2.3.2.2 DESIGN

The class diagram of Views is shown in Figure 117: Design. EIIAPP. Views. **BaseView** is responsible for wrapping each View inside a Consumer along with its ViewModel and **ExportCalendarView**, represents its state with "ExportCalendarState".

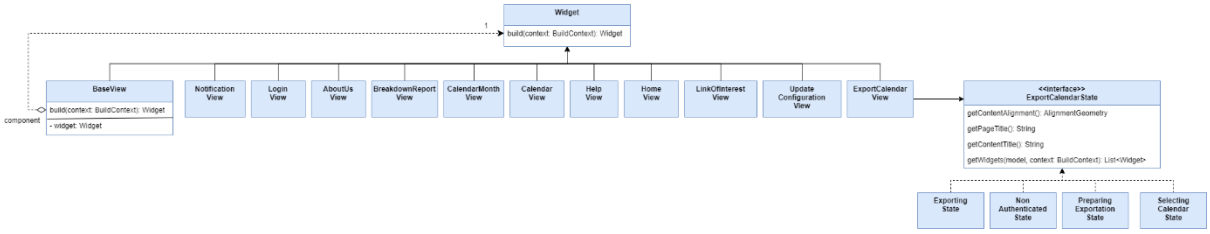


Figure 117: Design. EIIAPP. Views

6.2.3.2.3 PARTICIPANTS

The participants of the Decorator Pattern are listed below:

- **Component**: Widget.

- **ConcreteComponents**: NotificationView, LoginView, BreakdownReportView, CalendarMonthView, CalendarView, HomeView, LinkOfInterestView, UpdateConfigurationView and ExportCalendarView.
- **ConcreteDecorator**: BaseView.

The participants of the State Pattern are listed below:

- **Context**: ExportCalendarView.
- **State**: ExportCalendarStatw.
- **ConcreteStates**: ExportingState, NonAuthenticatedState, PreparingExportationState and SelectingCalendarState.

6.2.4 EXTENSION METHODS

As we have explained for EIISERVER, extension methods have also been used for EIIAPP for the same reasons (see Figure 118: Design. EIIAPP Extension Methods).

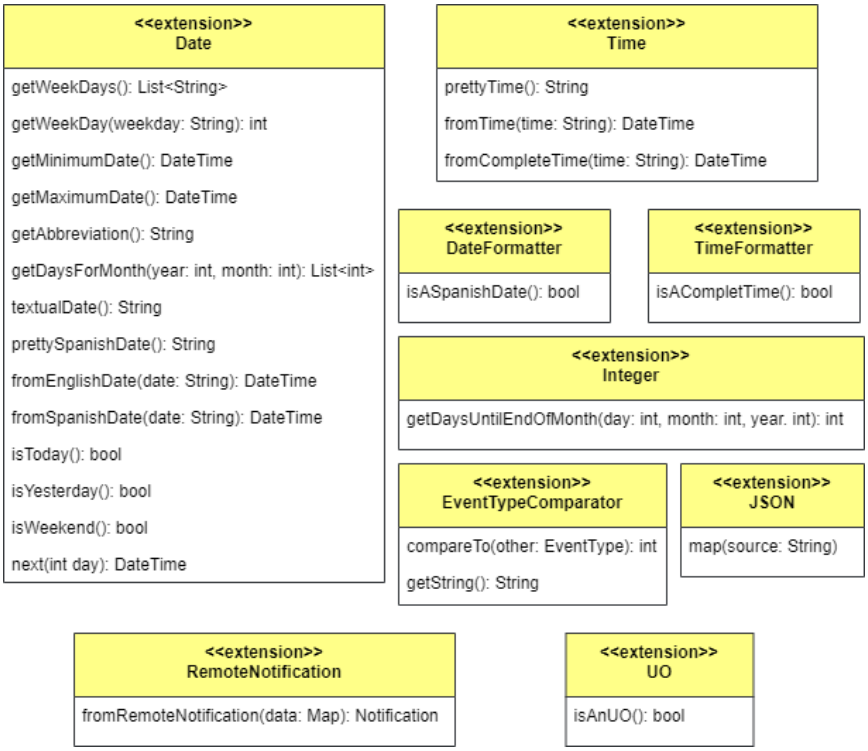


Figure 118: Design. EIIAPP Extension Methods

6.2.5 LOCATORS

Locators module is responsible for instantiating all classes of EIIAPP. It is modelled as a dependency injector. For instance, locator<GetEventsFromApiService>() would return the instance of GetEventsFromApiService class that was previously registered.

As Figure 119: Design. Locators shows, one Dependency Injector is created for the normal use of the application and other for testing purposes (which injects mock implementations).

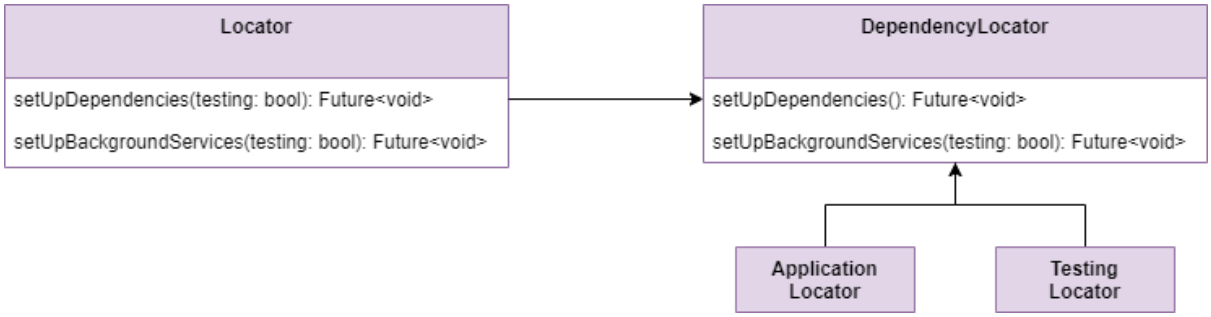


Figure 119: Design. Locators

6.3 CLASS DESIGN

6.3.1 CLASS DIAGRAM

Note that **the diagrams of this section** are also provided as **part of the contents delivered** together with this document (see Contents Delivered).

The class diagram of EIISERVER is shown in Figure 120: Class Diagram. EIISERVER. Classes belonging to the **core layer** are painted in a **light pink** colour, to the **transport** in a **light blue** colour, to the **datasource** in a **dark grey**, **factories** in a **light purple**, **extension methods** in a **yellow colour** and **loggers** in a **green** one. Note that some minor dependencies were not painted in order not to litter the diagram with lots of arrows.

The class diagram of EIIAPP is shown in Figure 121: Class Diagram. EIIAPP. Only some widgets were represented to illustrate these classes without collapsing the diagram. Classes belonging to the "**service**" directory are painted on **lime**, extension methods on **yellow**, locators on **light purple**, views and widgets on **light blue**, view models on **orange**, repositories on **pink** and datasources on **grey**.

To take a closer look into the dependencies and design of the classes of both systems, please see Design of the information system.

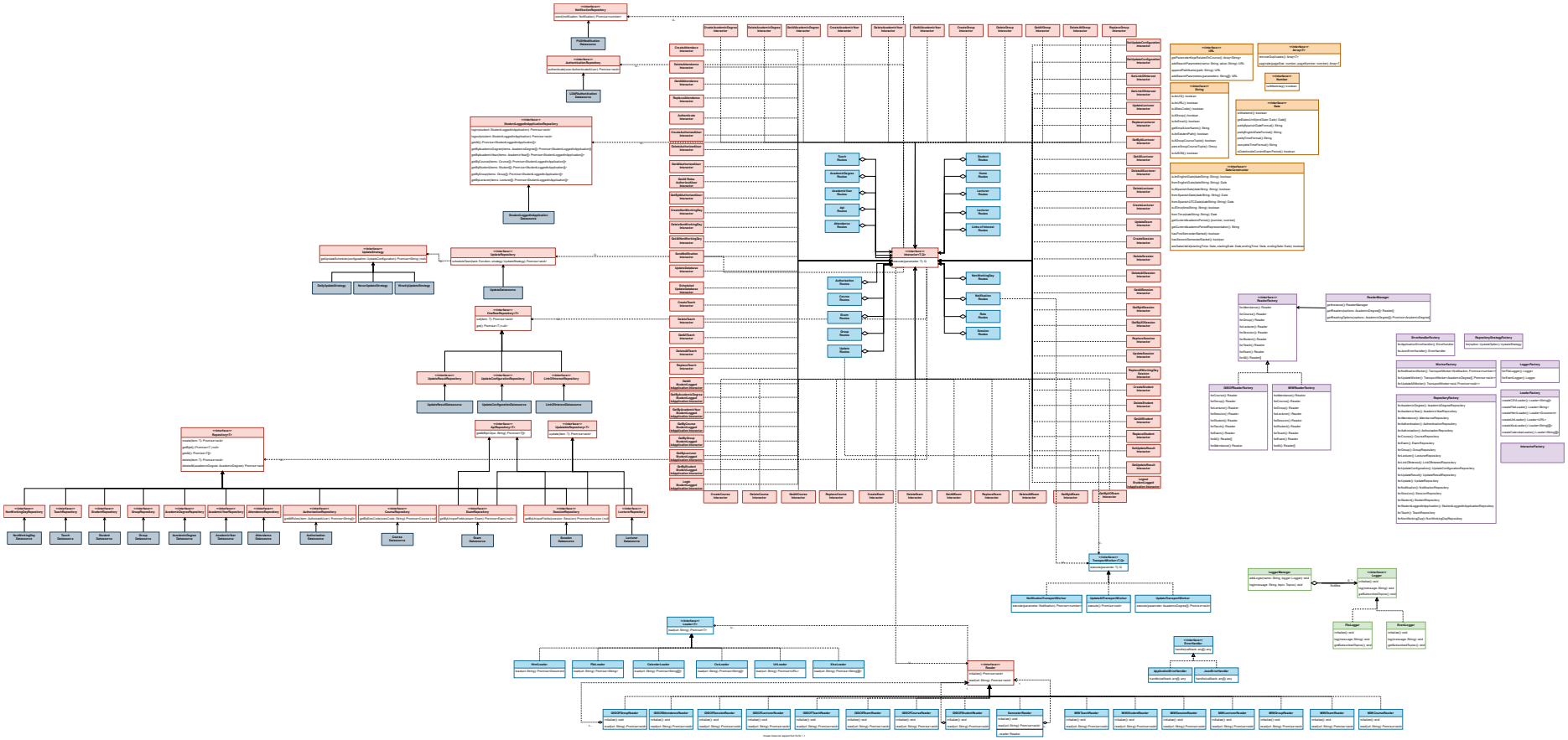


Figure 120: Class Diagram. EIISERVER

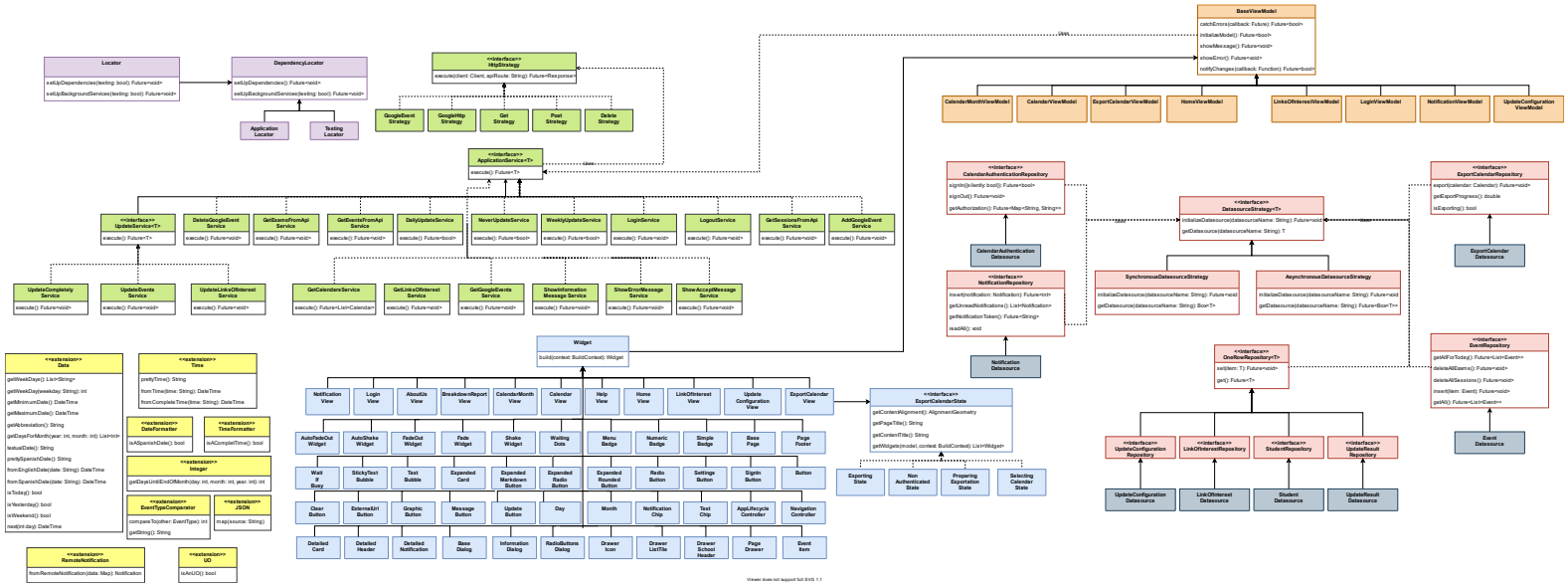


Figure 121: Class Diagram. EIIAPP

6.4 SYSTEM MODULE ARCHITECTURE DESIGN

6.4.1 SYSTEM MODULE DESIGN

The **Package Model View of EIIAPP** is illustrated on Figure 122: System Module Architecture Design. EIIAPP. Package Model View.

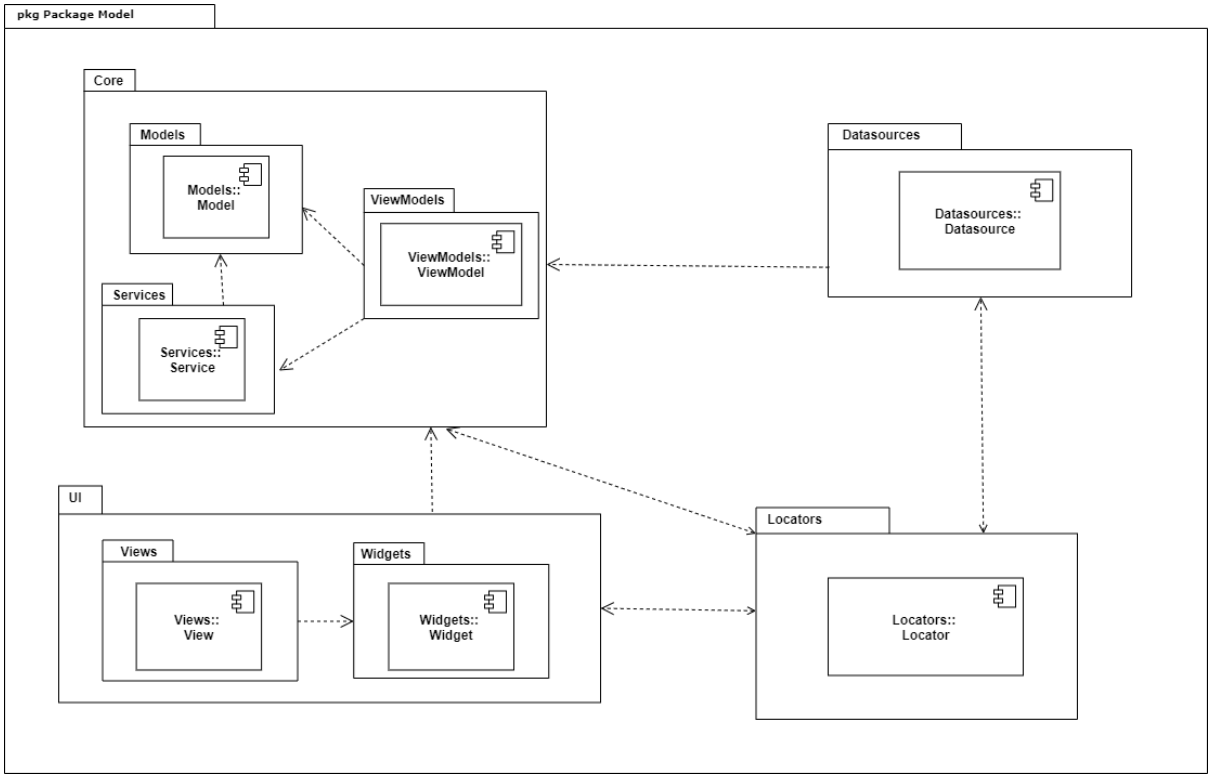


Figure 122: System Module Architecture Design. EIIAPP. Package Model View

On Figure 123: System Module Architecture Design. EIISERVER. Package Model View, the **Package Model View of EIISERVER** can be seen.

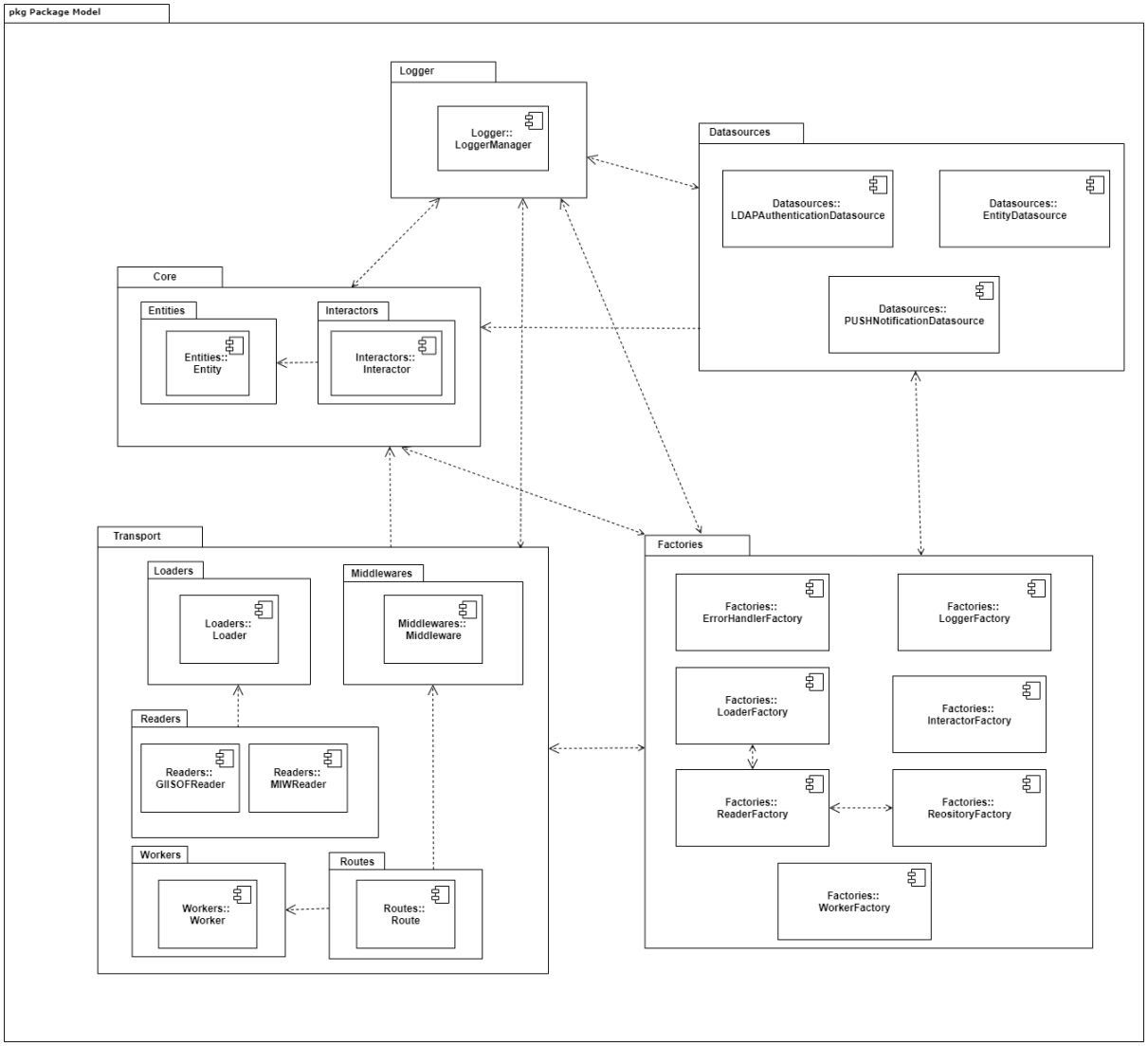


Figure 123: System Module Architecture Design. EIISERVER. Package Model View

6.4.2 DESIGN OF COMMUNICATIONS

The **network topology** that shows communications between modules and subsystems is detailed in Figure 124: Design of Communications. Topology. To see more information about these communications, please see Identification of Analysis Subsystems.

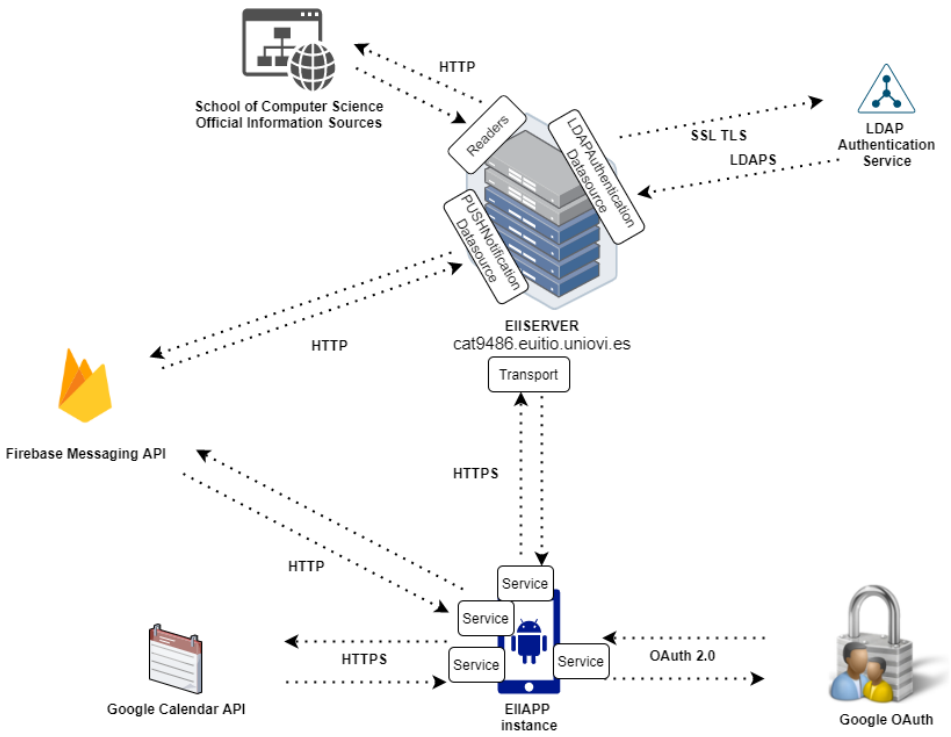


Figure 124: Design of Communications. Topology

6.5 PHYSICAL DATA DESIGN

In this section the use of two different technologies for data persistence is explained. One for the **EIISERVER** and another for **EIIAPP**.

6.5.1 DESCRIPTION OF THE DBMS USED

As far as **EIISERVER** is concerned, **PostgreSQL** has been used for persistence. The main reason behind the choice of this **RDBMS** has been the existence of a great number of relations between the entities, in comparison with the entities of **EIIAPP**.

The use of PostgreSQL was also **commented with the clients**, Luis Antonio Vinuesa Martínez and Fernando Álvarez García (School of Computer Science) and after evaluating the number of students enrolled in the Software Engineering Degree and the Master in Web Engineering in the current academic period, we concluded that PostgreSQL was suitable for performance.

On the other hand, for the **EIIAPP** we have use **Hive**, which is, according to its authors [16], "a lightweight and blazing fast key-value database written in pure Dart". Some of the reasons behind the use of Hive is:

- The **lack of relationships** between the entities of EIIAPP
- The need for a **local database** (as we have documented before, the users of EIIAPP need to access all its information offline).
- The need for a **database that works on Isolates**. As we have stated in previous chapters, some processes such as the receival of a notification needs to be run on an Isolate. Some technologies as **sqflite** [10], are do not work on Isolates (are not intend for Isolates).

To map the entities to persistence and from persistence to entities, **TypeAdapter** is used. It is based on the use of **annotations** to map the fields of our **entities** (see entity layer in Figure 6: Selection of the Architecture. EIISERVER. Clean Architecture).

The placement of **annotations inside the entities** (core of our EIIAPP system) litters the code (**strongly dependence between models and persistence**) but **facilitates the development** for EIIAPP system.

6.5.1.1 VERSIONS

The versions of the technologies used in both systems, EIIAPP and EIISERVER, are listed below:

- PostgreSQL 13.4
- pgAdmin 4
- Hive 2.0.4

6.5.2 INTEGRATION OF THE DBMS IN OUR SYSTEM

EIISERVER, as we have stated before, is deployed on a Virtual Machine of the School of Computer Science using Windows Server. For the integration of PostgreSQL, we start PostgreSQL service locally and connect to it from the deployed system. For its management we use pgAdmin. More information is detailed in User Manuals.

On the other hand, **Hive** is a local database. It will be located on each user's device and connected locally from each **EIIAPP** system instance.

6.5.3 RELATIONAL MODEL

The Relational Model of EIISERVER is illustrated in Figure 125: Relational Model. EIISERVER.

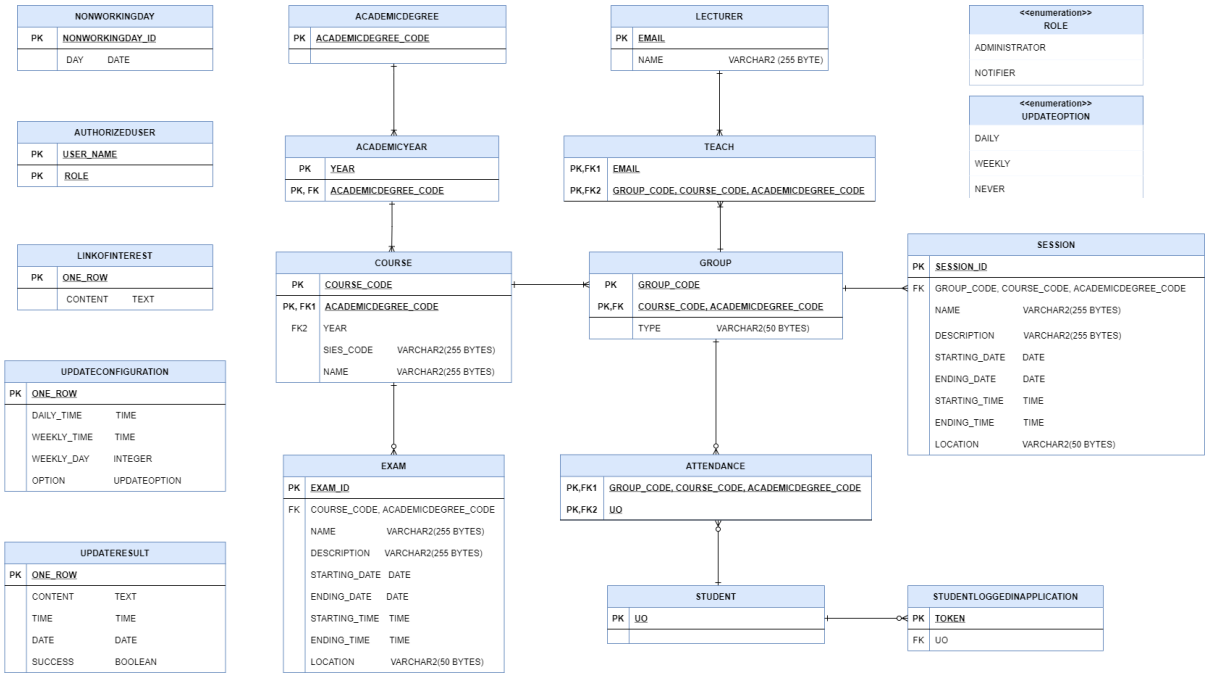


Figure 125: Relational Model. EIISERVER

Note that for EIIAPP we have used Hive (as stated in Description of the DBMS Used) and there are no relationships between entities. The diagram is illustrated in Figure 126: Relational Model. EIIAPP.

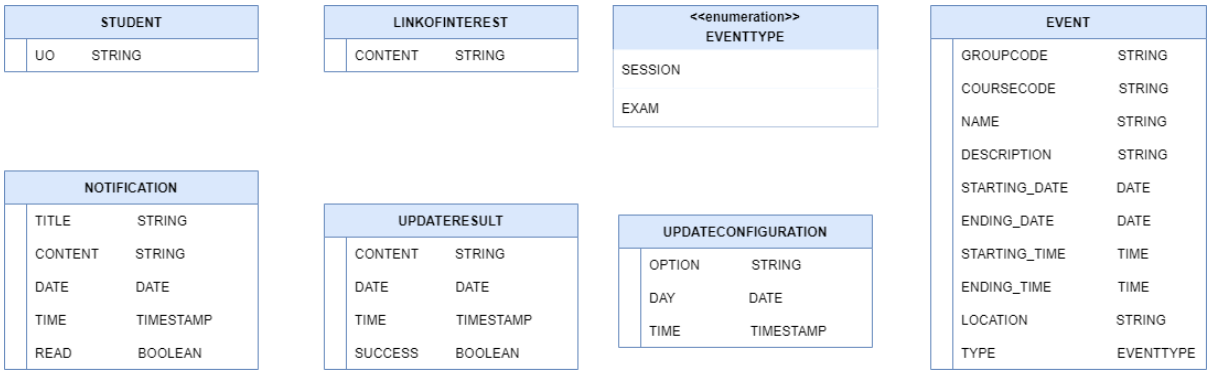


Figure 126: Relational Model. EIIAPP

6.6 TECHNICAL SPECIFICATION OF THE TESTING PLAN

6.6.1 EIISERVER

6.6.1.1 TESTING TECHNIQUES

A key part of EIISERVER are the **Readers**, located on the **transport layer** of the system (see Figure 6: Selection of the Architecture. EIISERVER. Clean Architecture). These classes are responsible for reading all information exposed on the official sources [3]–[5], and parsing it, determining which is correct/valid and should be decoded into an **Entity** and which not.

On the other hand, **Checkers**, located also on the transport layer, have been analysed and testing techniques have been applied.

Testing techniques have been applied to **all Readers and all Checkers** but here is only one shown, GIISOF Attendance Reader, to illustrate the process and to avoid repetition.

6.6.1.1.1 GIISOF ATTENDANCE READER

6.6.1.1.1.1 EQUIVALENCE PARTITIONING

- **Inputs**
  - Course Code
    - Provided
    - Not Provided
  - Group Code
    - Follows Group Code Format
    - Does not Follow Group Code Format
  - UO
    - Valid UO
    - Not Valid UO
- **Outputs**
  - Attendance Entity Decoded with all its fields
  - Attendance Entity Not Decoded

6.6.1.1.1.2 BASE CHOICE

Course Code	Group Code	UO	Attendance Entity
Provided	Follows Group Code Format	Valid	Decoded
Not Provided	Follows Group Code Format	Valid	Not Decoded
Provided	Does not Follow Group Code Format	Valid	Not Decoded
Provided	Follows Group Code Format	Not Valid	Not Decoded

Table 33: GIISOF Attendance Reader. Base Choice

6.6.1.2 UNIT TESTING

All unit tests have been automatized with **Jest** [17]. We also make use **of Jest-Each** to parameterise our tests, as Figure 127: Unit testing. EIISERVER. Jest-Each shows.

```
each`
    test                                                      | course                                                                              | expected
    ${"name is ok, siesCode is ok, year is ok, code is ok"}      | ${{"code": "AL", "name": "Álgebra lineal", "siesCode": "GIISOF01-1-002", "year": 0}}      | ${1}
    ${"name is not ok, siesCode is ok, year is ok, code is ok"}  | ${{"code": "AL", "siesCode": "GIISOF01-1-002", "year": 1}}                             | ${0}
    ${"name is ok, siesCode is not ok, year is ok, code is ok"}  | ${{"code": "AL", "name": "Álgebra lineal", "year": 1}}                                 | ${0}
    ${"name is ok, siesCode is ok, year is not ok, code is ok"}  | ${{"code": "AL", "name": "Álgebra lineal", "siesCode": "GIISOF01-1-002", "year": "year"}} | ${0}
    ${"name is ok, siesCode is ok, year is ok, code is not ok"}  | ${{"name": "Álgebra lineal", "siesCode": "GIISOF01-1-002", "year": 1}}                  | ${0}
    `

    .test( title: 'when $test, interactor should be called $expected times', test: async ({course, expected}) => {
```

Figure 127: Unit testing. EIISERVER. Jest-Each

All tests, as far as possible, were design following the pattern: **initialize** - **execute** – **expect** and were self-documented on the code.

6.6.1.3 ACCEPTANCE TESTING

For acceptance testing, we used **Cucumber** [18] with **Gherkin**. The tests followed the pattern: **Given – When – Then** and the following features and scenarios were designed:

6.6.1.3.1 DATABASE FEATURE

The following scenarios were designed for Database Feature: Table 34: Acceptance Testing. EIISERVER. Administrator modifies database adding entities, Table 35: Acceptance Testing. EIISERVER. Administrator modifies database editing entities, Table 36: Acceptance Testing. EIISERVER. Administrator manually updates database and Table 37: Acceptance Testing. EIISERVER. Administrator schedules an update.

The **scenarios** were designed to cover the **main actions of an Administrator**: adding an Entity, editing an Entity, updating the database, and scheduling an update.

Administrator modifies database adding entities	
Given	A user that logs in as "UO123456"
When	Administrator adds an Academic Degree
Then	Added Academic Degree should be shown

Table 34: Acceptance Testing. EIISERVER. Administrator modifies database adding entities

Administrator modifies database editing entities	
Given	A user that logs in as "UO123456"
When	Administrator edits a Lecturer
Then	Edited Lecturer should be shown

Table 35: Acceptance Testing. EIISERVER. Administrator modifies database editing entities

Administrator manually updates database	
Given	A user that logs in as "UO123456"
When	Administrator updates the database
Then	A message indicating that the database was updated should be shown

Table 36: Acceptance Testing. EIISERVER. Administrator manually updates database

Administrator schedules an update	
Given	A user that logs in as "UO123456"
When	Administrator schedules a "DAILY" update
Then	A message indicating that the update was scheduled should be shown

Table 37: Acceptance Testing. EIISERVER. Administrator schedules an update

6.6.1.3.2 SENDING NOTIFICATIONS FEATURE

The following scenario was designed for Sending Notifications Feature: Table 38: Acceptance Testing. EIISERVER. Notifier sends a notification to an Academic Degree.

The **scenario** was designed to cover the **main action of a Notifier**: sending a broadcast notification.

Notifier sends a notification to an Academic Degree	
Given	A user that logs in as "UO654321"
When	Notifier accesses "/notifications/academicdegrees" to send a message
Then	A message indicating that the notification was sent should be shown

Table 38: Acceptance Testing. EIISERVER. Notifier sends a notification to an Academic Degree

6.6.1.3.3 RESTRCITED ROUTES

The following scenarios were designed for Restricted Routes Feature: Table 39: Acceptance Testing. EIISERVER. Administrator logs in and accesses a restricted route, Table 40: Acceptance Testing. EIISERVER. Notifier logs in and accesses a restricted route and Table 41: Acceptance Testing. EIISERVER. Notifier logs in and changes its role.

The **scenarios** were designed to cover the **main restrictions that EIISERVER shall implement**: restricting Administrator actions to Notifiers and vice versa.

Administrator logs in and accesses a restricted route	
Given	A user that logs in as "UO123456"
When	He accesses "/notifications/academicdegrees"
Then	A message should be shown

Table 39: Acceptance Testing. EIISERVER. Administrator logs in and accesses a restricted route

Notifier logs in and accesses a restricted route	
Given	A user that logs in as "UO654321"
When	He accesses "/academicdegrees"
Then	A message should be shown

Table 40: Acceptance Testing. EIISERVER. Notifier logs in and accesses a restricted route

Notifier logs in and changes its role	
Given	A user that logs in as "UO654321"
When	He changes its role to "ADMINISTRADOR" and accesses "/academicdegrees"
Then	Academic Degrees page should be shown

Table 41: Acceptance Testing. EIISERVER. Notifier logs in and changes its role

6.6.2 EIIAPP

For EIIAPP testing, **Flutter Testing Documentation** [19] was followed. All categories, as named by Flutter, were tested: Unit testing, Widget testing and Integration testing.

6.6.2.1 UNIT TESTING

For Unit Testing, we have used **flutter_test**, external dependencies were mocked using **mockito** and tests were parameterized. All tests were self-documented on the code.

6.6.2.2 WIDGET TESTING

For Widget Testing, we have also used **flutter_test** to test the interaction between the user and the User Interface, i.e., testing the **Widgets** of the UI layer. All tests were self-documented on the code.

6.6.2.3 INTEGRATION TESTING

For Integration Testing, **integration_test** was used to test EIIAPP completely on a simulator as well as on a real device (see Technologies Used). **When** are integration tests passed, designed and developed is detail in Initial Planning. WBS. The main integration tests, among others, designed covered the following:

6.6.2.3.1 LOGIN VIEW

- The redirection to Login View when user is not authenticated on EIIAPP
- The redirection to Home View after logs in correctly
- The lack of redirection when login process fails were tested

6.6.2.3.2 HOME VIEW

- The display of today events
- The update of today events after clicking update

6.6.2.4 NOTIFICATION VIEW

- The display of the notifications
- The search by words
- The search by date

6.6.2.5 UPDATE CONFIGURATION VIEW

- The configuration of updates

6.6.2.6 EXPORT CALENDAR VIEW

- The redirection to Google Login when user is not authenticated
- The redirection to Select Calendars when user is authenticated

6.6.2.6.1 LINKS OF INTEREST VIEW

- The display of the links of interest
- The update of the links of interest after clicking update

6.6.2.6.2 CALENDAR VIEW

- The display of the events
- The update of the events after clicking update
- The increase/decrement of the selected month

6.6.2.6.3 CALENDAR MONTH VIEW

- The display of the selected month when view is accessed
- The increase/decrement of the selected year

6.6.3 TECHNOLOGIES USED FOR TESTING

6.6.3.1 EIISERVER

- WebStorm 2020.3.2
  - Build #WS-203.7148.54, built on January 25, 2021
  - For educational use only.
  - Runtime version: 11.0.9.1+11-b1145.77 amd64
  - VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.
  - Windows 10 10.0
  - GC: ParNew, ConcurrentMarkSweep
  - Memory: 1954M
  - Cores: 12
- Google Chrome Navigator
  - Version: 94.0.4606.71
- cucumber-pretty: 6.0.0
- cucumber-tsflow: 4.0.0-rc.1
- chromedriver: 93.0.1
- jest: 26.6.3
- jest-each: 26.6.2
- ts-jest: 26.5.5

6.6.3.2 EIIAPP

- Devices used for Integration Testing
  - Emulator (Pixel 2 API 29)
    - Android SDK built for x86
    - Build number: QSR1.190920.001
    - Android version: 10
  - Real Device (Realme C11)
    - Android version: 10
    - RAM: 2,00 GB
    - Model: RMX2185
- integration_test

    - sdk: flutter
- flutter_driver
    - sdk: flutter
- flutter_test
    - sdk: flutter
- test: any

6.6.4 RUN TESTS

6.6.4.1 RUN EIISERVER UNIT TESTS

To run the tests, open the Command Line Console and change your location to the root of the EIISERVER Project. Then execute the following command: **npm test**.

6.6.4.2 RUN EIISERVER ACCEPTANCE TESTS

To run the tests, open the Command Line Console and change your location to the root of the EIISERVER Project. Then execute the following command: **npm run-script start-cucumber**.

Open another Command Line Console and change your location to the root of the EIISERVER Project. Then execute the following command: **npm run-script cucumber**. Please use a version of **Google Chrome Navigator** compatible with the **chromedriver** version specified in EIISERVER.

6.6.4.3 RUN EIIAPP UNIT AND WIDGET TESTS

To run the tests, open the Command Line Console and change your location to the root of the EIIAPP Project. Then execute the following command: **flutter test**.

6.6.4.4 RUN EIIAPP INTEGRATION TESTS

To run the tests, select the device from the Visual Studio Code Bottom Bar (see Figure 128: Run EIIAPP integration tests. Select device). Please, use a device with the specifications specified in EIIAPP.

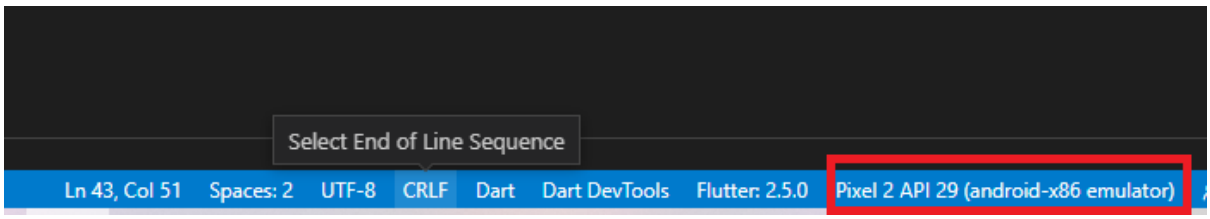


Figure 128: Run EIIAPP integration tests. Select device

Then, open the Command Line Console and change your location to the root of the EIIAPP Project. Then execute the following command: **flutter test integration_test**.

7 CONSTRUCTION OF THE INFORMATION SYSTEM

7.1 PREPARATION OF THE GENERATION AND CONSTRUCTION ENVIRONMENT

7.1.1 STANDARDS AND NORMS

- For the design and development of **EIISERVER**, **Node best practices** [14] and **Google TypeScript Style Guide** [20] have been followed.
- For the design and development of **EIIAPP**, **Dart Guidelines for development** [21] and **Effective Flutter Testing** [12].
- For the design and development of the **user interfaces** of the project, **Material Design** [12].
- For the contents of this document, **MetricaV3** [9].

7.1.2 PROGRAMMING LANGUAGES

- **JavaScript**
  - Version: NodeJS 8.4.371.19-node.18
- **TypeScript**
  - Version: 4.3.4
- **HTML5**
- **Dart**
  - Version: 2.14.0

7.1.3 TOOLS AND PROGRAMS USED

- **WebStorm 2020.3.2**
  - Build #WS-203.7148.54, built on January 25, 2021
  - For educational use only.
  - Runtime version: 11.0.9.1+11-b1145.77 amd64
  - VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.
  - Windows 10 10.0
  - GC: ParNew, ConcurrentMarkSweep
  - Memory: 1954M
  - Cores: 12
- **Visual Studio Code**
  - Version: 1.61.0
  - OS: Windows_NT x64 10.0.19042
- **Windows Server 2016**
  - Version: 1607
- **Internet Information Services (IIS)**
  - Version: 10.0.14393.0
- **Google Chrome**
  - Version: 94.0.4606.71
- **Git**
  - Version: 2.31.1.windows.1

7.2 EXECUTION OF EIISERVER TESTS

In this section, the **most remarkable aspects** about EIISERVER and EIIAPP tests execution will be pointed out, illustrating how testing helped us improving our systems.

7.2.1 EXECUTION OF EIISERVER UNIT TESTS

Code coverage of EIISERVER, generated by the command specified in several previous sections, is shown in Figure 129: Execution of EIISERVER Unit Tests. Code Coverage.

All files

96.93% Statements 5369/5539 **87.52%** Branches 1431/1635 **94.62%** Functions 1530/1617 **97.56%** Lines 3806/3901

Figure 129: Execution of EIISERVER Unit Tests. Code Coverage

These numbers are the result of the application of the selected architecture, dependency injections, mocks as well as other design principles (all they explained throughout this document) that allowed us to reach and cover **96.96%** of EIISERVER statements, for instance.

The execution of the Reader Tests uncovered a bug present in our code:

- Sessions and Exams stored **null** on dates if they were not provided. If no date is provided, it makes no sense to store an event.

Another bug discovered, particularly in Datasources and Interactors modules, was the following one:

- If we write a sentence like the one shown in Figure 130: Execution of EIISERVER Tests. Typescript If Statement.
    - It will return false if "item.academicDegreeCode" is undefined or null.
    - It will **return false** if "item.academicDegreeCode" is numeric and **its value is zero**.

In that case, for **numeric values**, that statement will fail for **0**. We needed to change if statements to the one shown in Figure 131: Execution of EIISERVER Tests. Typescript If Statement (version 2) for numeric fields.

```
if(item.academicDegreeCode)
```

Figure 130: Execution of EIISERVER Tests. Typescript If Statement

```
if(!isNaN(obj.year))
```

Figure 131: Execution of EIISERVER Tests. Typescript If Statement (version 2)

7.2.2 EXECUTION OF EIISERVER ACCEPTANCE TESTS

An improvement was applied after the execution of Restricted Routes Feature:

- If the user accessed a restricted route, an error in JSON format was returned.
- This could be suitable for EIISERVER API, but not for its web client.
- Whenever a user accesses a restricted route is now redirected to an Error Page like the one shown in Figure 132: Execution of EIISERVER Tests. Error Page.

Figure 132: Execution of EIISERVER Tests. Error Page

7.3 EXECUTION OF EIIAPP TESTS

7.3.1 EXECUTION OF EIIAPP UNIT AND WIDGET TESTS

The most remarkable bugs found are listed below:

- A user **could go back to Login View after logging in** by pressing Back Button on its device. Now if the user is in Home View and presses back it exits the app.
- A user **could constantly press Update Button** (located in Home, Calendar and Link of Interest View). System overflows and crashes. Now if the system is updating, the "press" is ignored, and no other request is sent.
- After clicking Update Button, the user **needed to re-enter the Page to see the changes**. Now Page is refreshed after clicking Updating without needing to re-enter.
- When a notification arrived, **if no date and time fields are provided, notification was discarded**. Now, if no date and time fields are provided, current date and time are used instead.
- If a **notification arrives when the app is on background**, **notification badge** (see Figure 133: Execution of EIIAPP Tests. Notification Badge) **is not updated**. Now, app lifecycle is listened whenever its state is changed (when it changes from background to foreground, for instance).

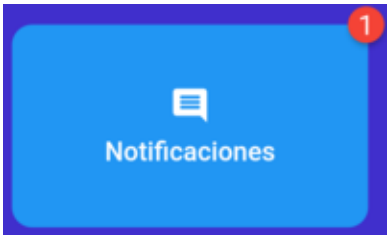


Figure 133: Execution of EIIAPP Tests. Notification Badge

7.3.2 EXECUTION OF EIIAPP INTEGRATION TESTS

An improvement was applied after the execution of Integration Tests:

- On Notification View, if user is on the bottom of the View, Scroll-Down Floating Action Button is not visible. But if a new notification arrives (see Figure 134: Execution of EIIAPP Tests. Scroll-Down Floating Action Button), this one is placed at the very bottom and **Scroll-Down Floating Action Button should now be visible again** (there is a lower notification).

Scroll View Changes were listened, and it was determined if the Floating Action was needed to be showed.

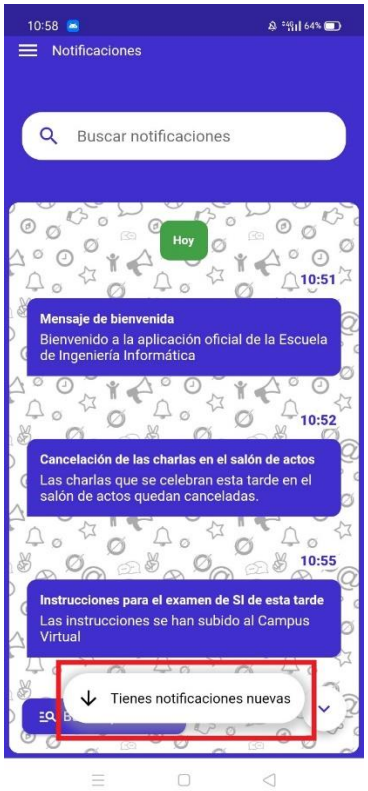


Figure 134: Execution of EIIAPP Tests. Scroll-Down Floating Action Button

7.4 USER MANUALS

All manuals are placed in the indicated directory (see Contents Delivered).

8 INTRODUCTION AND ACCEPTANCE OF THE SYSTEM

8.1 ESTABLISHMENT OF THE INTRODUCTION PLAN

EIISERVER will be deployed **on the School of Computer Science premises**, particularly, on the Datacenter of the previously mentioned client.

To achieve this, a virtual machine will be used with the following features:

- Windows machine, **virtually located on the Datacenter** of the School of Computer Science
- **Microsoft Windows Server 2016**

Microsoft Internet Information Services (IIS) will be used to host the contents that conform EIISERVER, along with **IISNode** [22].

EIISERVER will use **HTTPS protocol**, installing a **certificate** provided by the client, and will be **hosted in** a subdomain of **"uniovi.es" domain** whose complete name will be given by the name of the previously mentioned virtual machine.

Port 443 shall be opened, making **EIISERVER accessible outside** from the School of Computer Science intranet (note that EIIAPP shall communicate with EIISERVER API).

Additionally, as mentioned throughout this document, EIISERVER will use **University of Oviedo's LDAP Authentication Service** to authenticate users of EIISERVER and the **official sources of information of the School of Computer Science** [3]–[5]to read and parse information.

On the other hand, EIIAPP will be released on **Google Play**, using the School of Computer Science official account.

8.2 UPLOADING DATA TO THE OPERATING ENVIRONMENT

The necessary upload of data to EIISERVER system is detailed in **Maintenance and Deployment Manual for EIIProject** (*Maintenance and Deployment Manual*), section "Initial Data". The manual is located under the directory specified in Contents Delivered.

8.3 PRESENTATION AND APPROVAL OF THE SYSTEM AND GOING INTO PRODUCTION

On the one hand, EIISERVER system is **in production** and **deployed** on https://cat9486.euitio.uniovi.es/. On the other hand, EIIAPP system is **released** on Google Play Store.

9 ANNEXES

9.1 RISK MANAGEMENT PLAN

9.1.1 METHODOLOGY

To carry out an efficient risk management, **Boehm Methodology** [23] will be used. This methodology is divided in two main phases:

- **Risk Assessment**:
  - **Risk Identification**: during this step, a list of potential risks for the project will be generated.
  - **Risk Analysis**: during this step, probability, impact, and risk levels will be measured.
  - **Risk Prioritization**: the list of risks will be prioritized attending to its importance for the project.
- **Risk Control**:
  - **Risk Management Plan**: a plan for each significative risk will be generated.
  - **Risk Resolution**: risk plan will be executed.
  - **Risk Monitoring**: risks will be monitored.

9.1.2 TOOLS AND TECHNIQUES

To collect information, the following techniques have been used:

9.1.2.1 BRAINSTORMING

Even if it is recommended to have a group of people for the application of this technique, it has been used to collect quickly and specify the ideas that have appeared throughout the session performed by the unique member of the Development Team.

9.1.2.2 EXPERIENCE FROM LAST PROJECTS

Considering that a "real project" was monitored in the **DPPI course of the School of Computer Science** [24], all the experience gained from that project was used and applied for the context of this project.

9.1.3 RISK CATEGORIES

Each identified risk will be accompanied by one of the **categories** shown in Figure 135: Risk Management Plan. Categories.

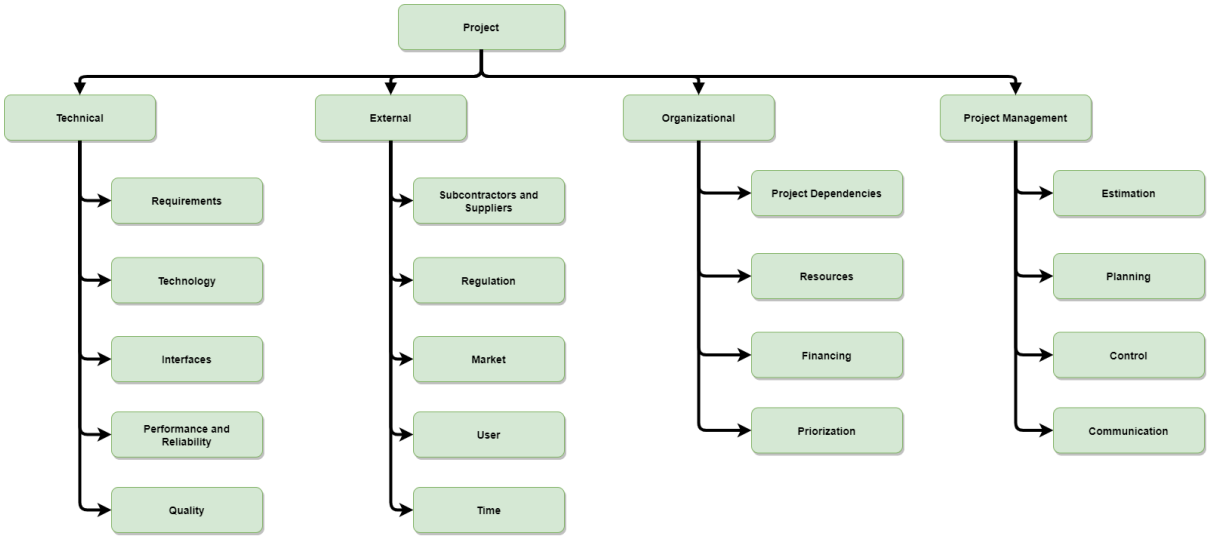


Figure 135: Risk Management Plan. Categories

9.1.4 PROBABILITY AND IMPACT DEFINITIONS

To define the **probability of a risk**, Table 42: Risk Management Plan. Probability will be taken in consideration.

Label	Range	Value used in the Probability/Impact Matrix
Rare	**[0%..20%]**	**10%**
Unlikely	**(20%..40%]**	**30%**
Possible	**(40%..60%]**	**50%**
Likely	**(60%..80%]**	**70%**
Almost Certain	**(80%..100%]**	**90%**

Table 42: Risk Management Plan. Probability

To define the impact of a (negative) risk, Table 43: Risk Management Plan. Impact will be used. With the use of the previously mentioned table, "**real cases are aimed to be exemplified** to discern the impact that we will assign to the risk studied.

Impact					
Project Objectives	Relative or numerical scales				
	Negligible (5%)	Low (10%)	Medium (20%)	High (40%)	Extreme (80%)
Cost	Cost increment of 0-5%	Cost increment of 5-10%	Cost increment of 10-20%	Cost increment of 20-40%	Cost increment is greater than 40%
Time	Time increment of 0-5%	Time increment of 5-10%	Time increment of 10-20%	Time increment of 20-40%	Time increment is greater than 40%
Scope	Scope reduction does not affect the client	Scope reduction affects, at least, an indirect stakeholder of the project	Scope reduction makes impossible documenting the whole project	Scope reduction makes impossible completing, at least, one	Scope reduction makes the project not being accepted by the client

			(completing all documentation tasks of the project)	module (highly tested, fulfilling all requirements, environment prepared for the production)	
Quality	Quality reduction does not affect the client	Quality reduction affects, at least, an indirect stakeholder of the project	Quality reduction makes impossible documenting the whole testing plan of the project. Quality reduction makes impossible designing and developing, at least, one of the test types of the project (unit testing, acceptance testing, among others)	Quality reduction makes impossible completing, at least, one module (highly tested, fulfilling all requirements, environment prepared for the production)	Quality reduction makes the project not being accepted by the client

Table 43: Risk Management Plan. Impact

9.1.5 PROBABILITY/IMPACT MATRIX

The probability/impact matrix of the project is illustrated in Table 44: Risk Management Plan. Probability/Impact Matrix.

Probability					
0,90	0,05	0,09	0,18	0,36	0,72
0,70	0,04	0,07	0,14	0,28	0,56
0,50	0,03	0,05	0,10	0,20	0,40
0,30	0,02	0,03	0,06	0,12	0,24
0,10	0,01	0,01	0,02	0,04	0,08
	0,05 Negligible	0,10 Low	0,20 Medium	0,40 High	0,80 Extreme
	Impact				

Table 44: Risk Management Plan. Probability/Impact Matrix

9.1.6 RISK TOLERANCE

Risk tolerance is set to **0.50**.

9.1.7 DOCUMENTATION FORMAT

All risks must be recorded in an **electronic document**. For each risk, the following **fields** must be provided:

- Name or risk identifier
- Brief description
- Category
- Probability
- Impact
- Risk response
- Strategy

9.1.8 MONITORING

For **each risk**, the following steps must be carried out:

- Keep the list of risk updated according to these fields:
    - Name or risk identifier
    - Brief description
    - Category
    - Probability
    - Impact
    - Risk response
    - Strategy
- Keep the contingency plans updated

9.1.9 CONTINGENCY PLANS

All risks whose impact is over tolerance (see Risk Tolerance) will have an associated contingency plan. In this case, "Bureaucratic processes that delay the project" risk will have the following plan:

The main actors that promote this risk are the **School of Computer Science** (responsible for the approval and acceptance of the project) and **Google** (responsible for the verification of EIIAPP system).

To mitigate this risk, reduce its impact, and reduce its probability, **communications with the client** will be held along the whole project (related with "Requirements Inflation"). The aim of this communications (via email) is to get the client information throughout the project execution and nuance all requirements and preferences.

All tasks related to **project management** (included Project Acceptance and Approval) will take the **12%** of total time of the project. These number was taken from the laboratory lectures of DPPI Course [24], and that are based on the experience.

In addition, the project is intended to be presented, ideally, in June and EIIProject planning on May (see Initial Planning. WBS), leaving weeks as contingency.

The task related to Google verification process ("Export Events Datasource") will also be incremented in time at the start of EIIProject (see Wrong planning of Datasource modules).

Nonetheless, if not enough, "**time**" constraint will be incremented in favour of "**quality**", as illustrated in the "Project Management Triangle" [24], i.e., **presenting EIIProject later in time**.

9.2 FROM THE ARCHITECTURE TO WBS

In this section we will describe the process followed to build the Work Breakdown Structure (WBS). The reference point is the architecture, more concretely the **Package Model View** of **EIISERVER** and **EIIAPP** (Figure 123: System Module Architecture Design. EIISERVER. Package Model View and Figure 122: System Module Architecture Design. EIIAPP. Package Model View, respectively).

9.2.1 BREAKDOWN OF SOFTWARE ACTIVITIES. EIISERVER

The first approach to the Product Breakdown Structure (PBS) of EIISERVER is shown in Figure 136: From the architecture to WBS. EIISERVER. First PBS.

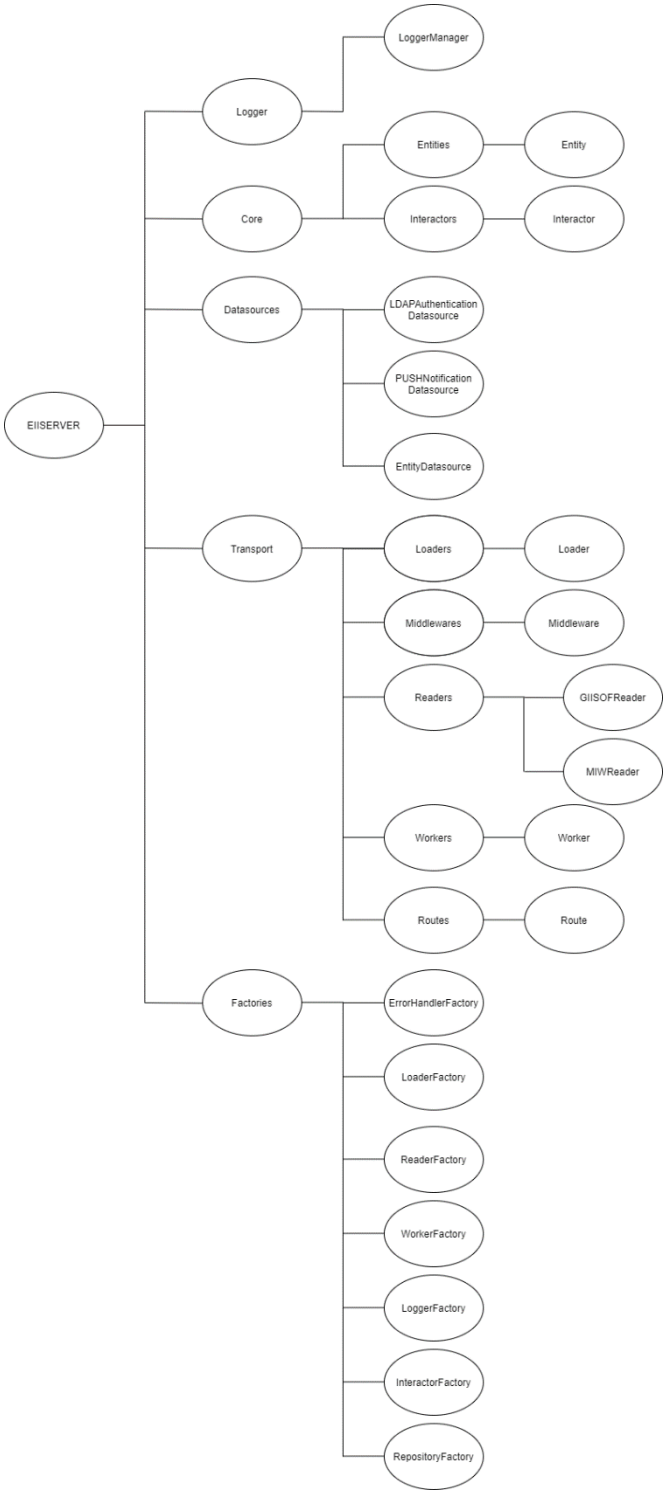


Figure 136: From the architecture to WBS. EIISERVER. First PBS

The first WBS/PBS of EIISERVER is depicted in Figure 137: From the architecture to WBS. EIISERVER. First WBS/PBS.

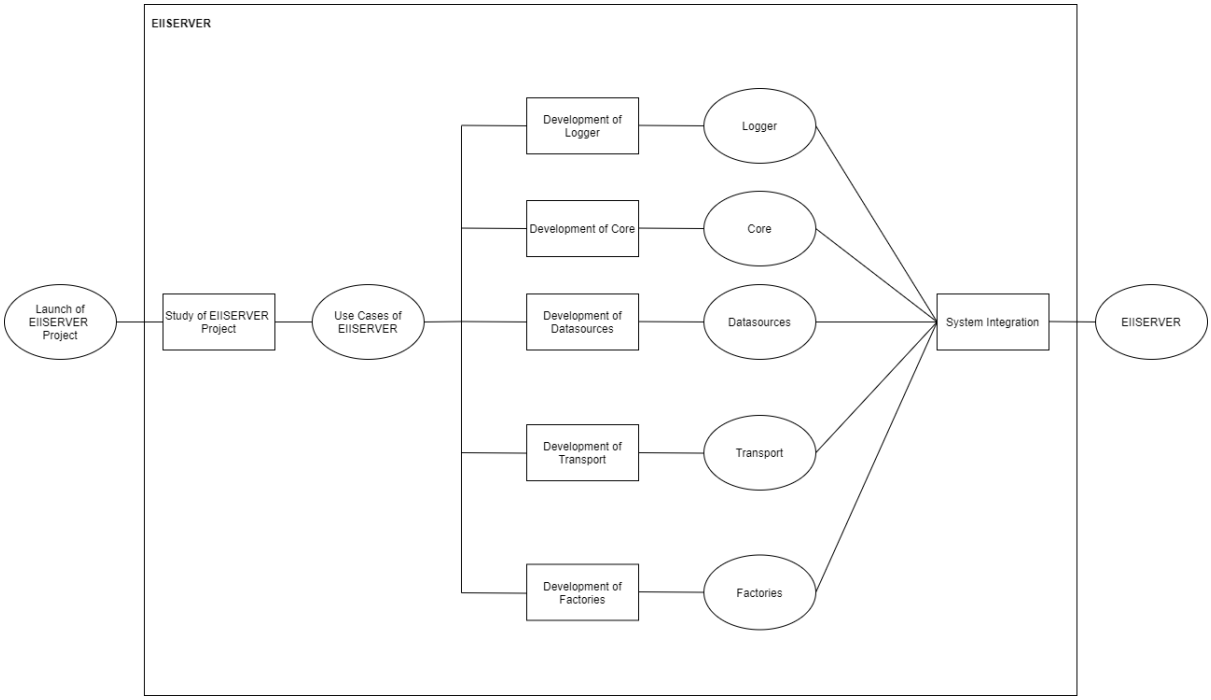


Figure 137: From the architecture to WBS. EIISERVER. First WBS/PBS

The second iteration of the construction of the WBS/PBS of EIISERVER is illustrated in [Figure 138: From the architecture to WBS. EIISERVER. Second WBS/PBS](#).

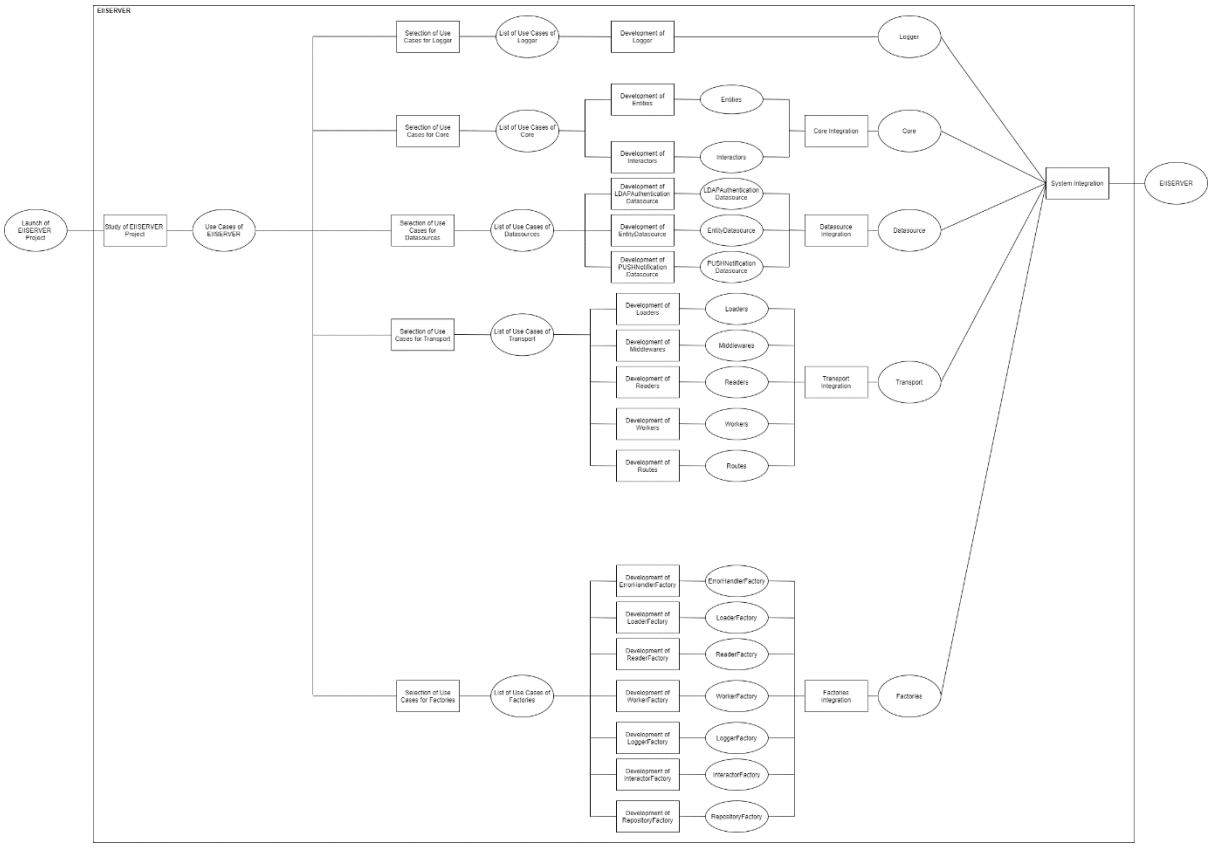


Figure 138: From the architecture to WBS. EIISERVER. Second WBS/PBS

After all iterations, the resultant PBS is the one shown in Figure 139: From the architecture to WBS. EIISERVER. PBS.

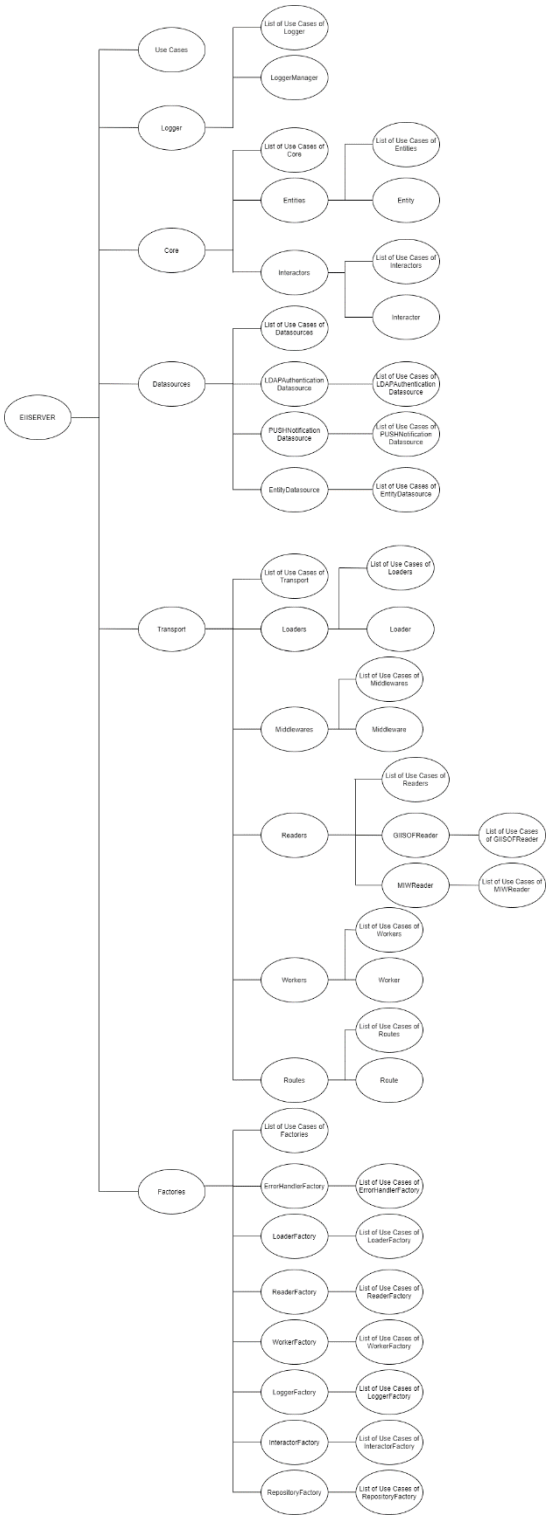


Figure 139: From the architecture to WBS. EIISERVER. PBS

In the same way, the resultant WBS is shown in Figure 140: From the architecture to WBS. EIISERVER. WBS.

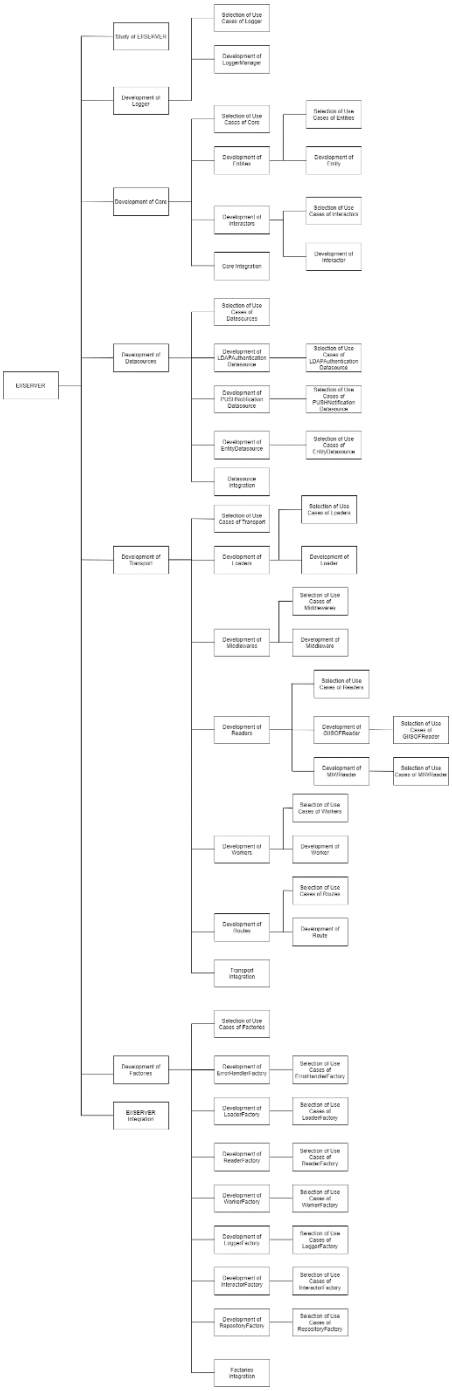


Figure 140: From the architecture to WBS. EIISERVER. WBS

9.2.2 BREAKDOWN OF SOFTWARE ACTIVITIES. EIIAPP

The first approach to the Product Breakdown Structure (PBS) of EIIAPP is shown in Figure 141: From the architecture to WBS. EIIAPP. First PBS.

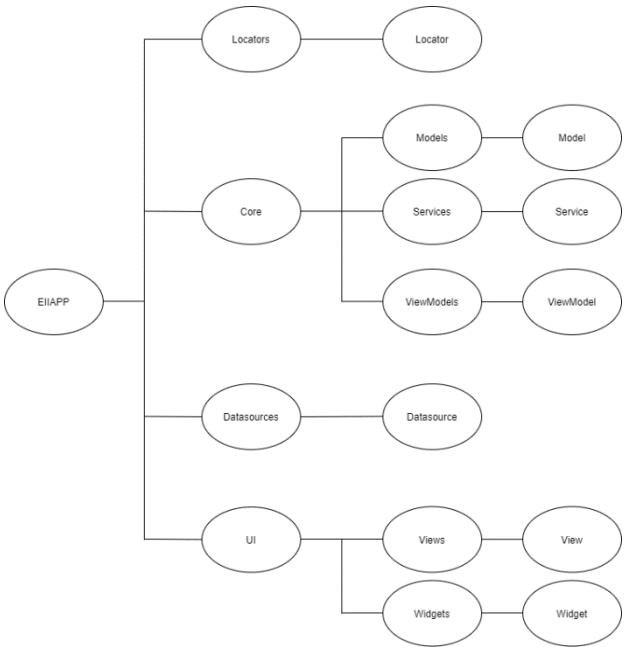


Figure 141: From the architecture to WBS. EIIAPP. First PBS

The first WBS/PBS of EIIAPP is depicted in Figure 142: From the architecture to WBS. EIIAPP. First WBS/PBS.

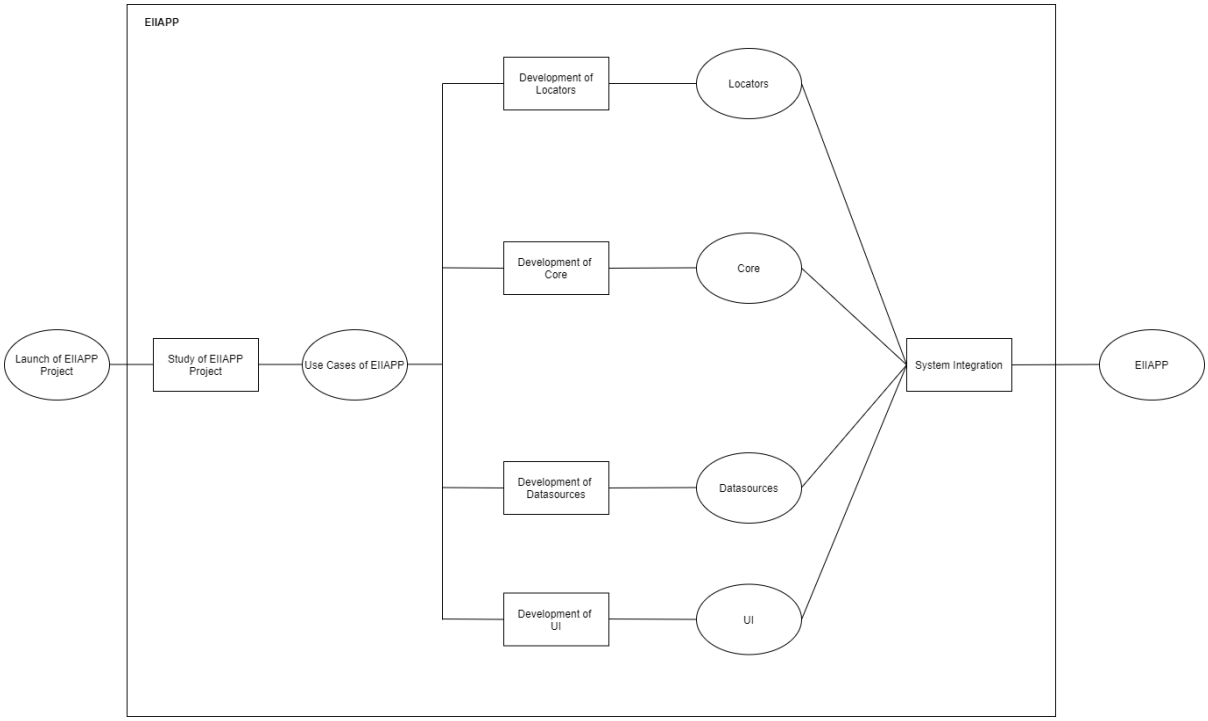


Figure 142: From the architecture to WBS. EIIAPP. First WBS/PBS

The second iteration of the construction of the WBS/PBS of EIIAPP is illustrated in Figure 143: From the architecture to WBS. EIIAPP. Second WBS/PBS.

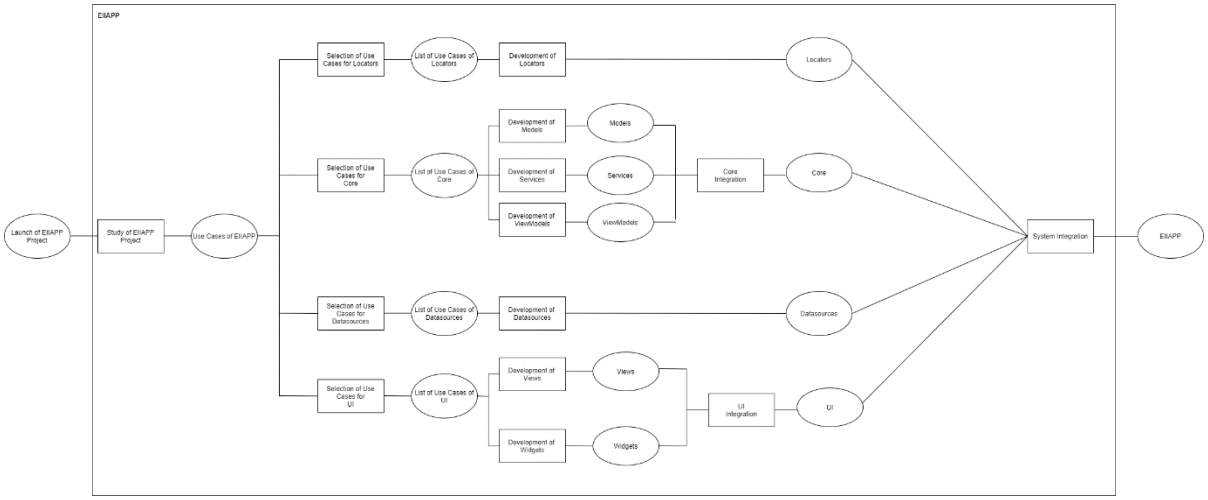


Figure 143: From the architecture to WBS. EIIAPP. Second WBS/PBS

After all iterations, the resultant PBS is the one shown in Figure 144: From the architecture to WBS. EIIAPP. PBS.

Figure 144: From the architecture to WBS. EIIAPP. PBS

In the same way, the resultant WBS is shown in [Figure 145: From the architecture to WBS. EIIAPP. WBS](#).

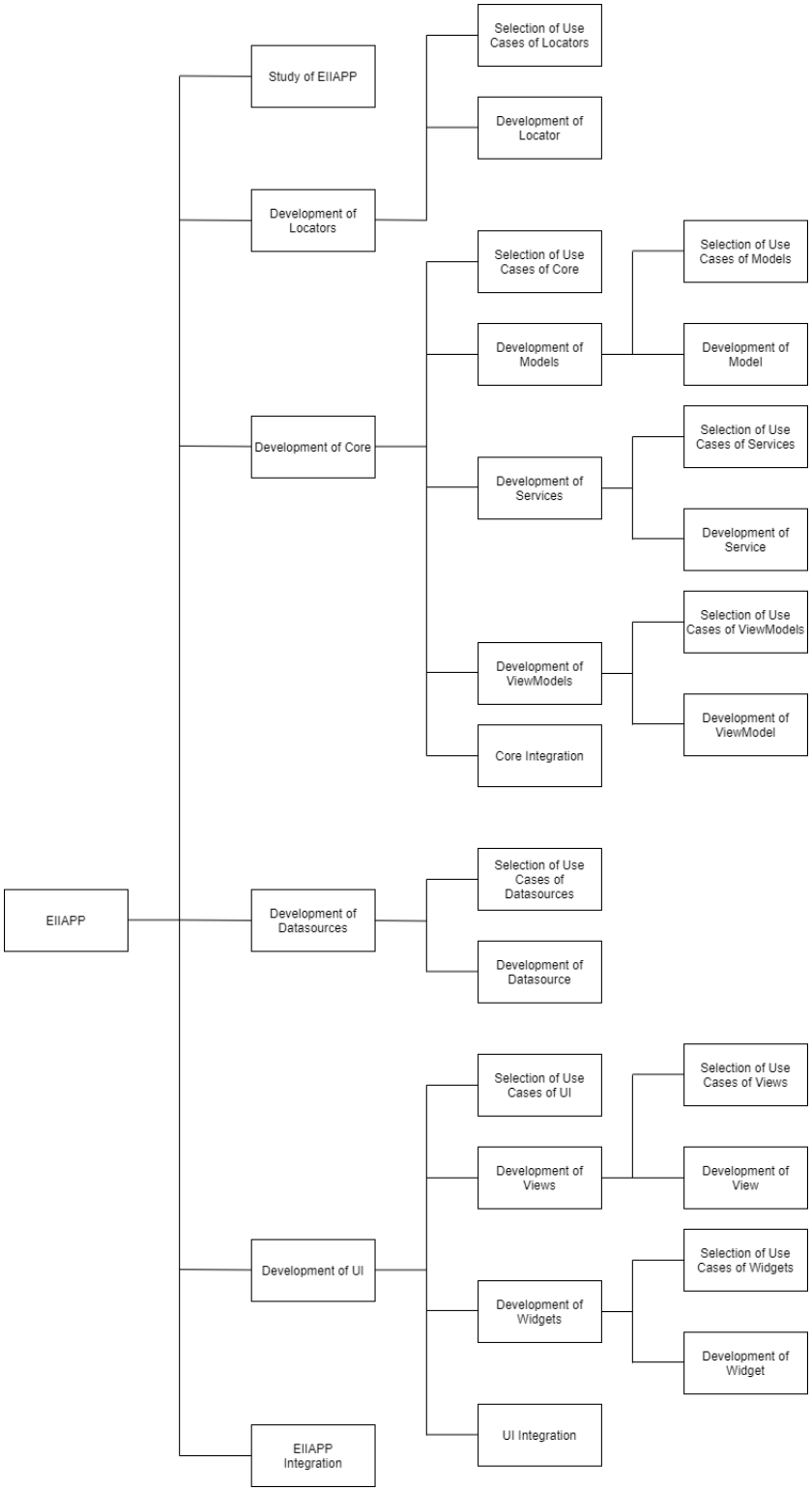


Figure 145: From the architecture to WBS. EIIAPP. WBS

9.2.3 BREAKDOWN OF HARDWARE ACTIVITIES

Note that **EIISERVER** is deployed on the premises of the client, School of Computer Science, so the infrastructure is provided by the School. In other words, **the server where the system is deployed is in charge of the School**.

On the other hand, **EIIAPP** is an application released on the **School's account of Google Play Store** and installed by the students themselves on their mobile devices.

The **WBS** is illustrated in Figure 147: From the architecture to WBS. WBS (Hardware) and the **PBS** in Figure 146: From the architecture to WBS. PBS (Hardware).

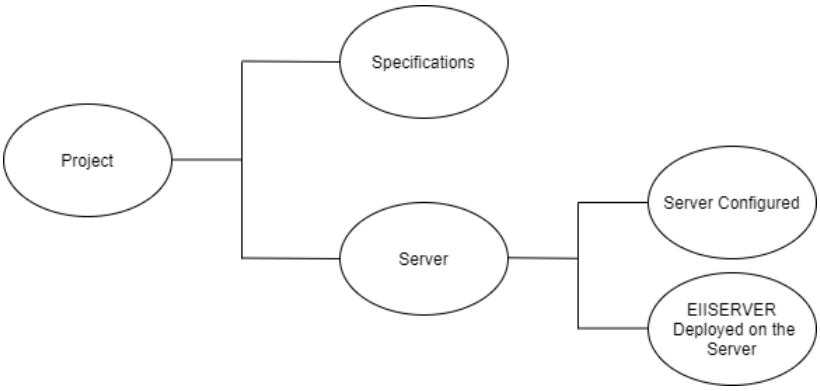


Figure 146: From the architecture to WBS. PBS (Hardware)

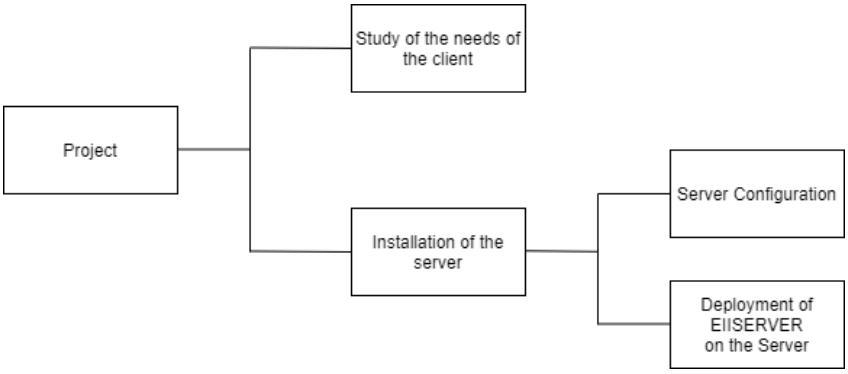


Figure 147: From the architecture to WBS. WBS (Hardware)

9.3 ESTIMATION OF SIZE AND EFFORT

To estimate the size and effort to build both systems, EIIAPP and EIISERVER, we will start from the architecture designed and documented on Definition of the technological architecture. To tackle this section, we will follow the same process taught in the **DPPI course of the School of Computer Science** [24].

Note that this process should be applied for each module of the project, but in this case, given the fact that the developed project is **not as complex as the HIS Project illustrated on the DPPI course** [24], we will take **EIIAPP** as one "module" and **EIISERVER** as another "module".

9.3.1 EIISERVER

9.3.1.1 MODULE DEFINITION

9.3.1.1.1 FUNCTIONALITIES

- Authenticates users against LDAP Service.
- Stores information related to the Software Engineering Degree and Master in Web Engineering.
    - Allows a user to create an entity.
    - Allows a user to delete an entity.
    - Allows a user to read all entities.
    - Allows a user to update an entity.
- Updates the stored information.
- Schedules updates.
- Sends notifications to EIIAPP.

9.3.1.1.1.1 ARCHIVES

The system will need the following archives:

- Academic Degrees Register
- Academic Years Register
- Attendances Register
- Authenticated Users Register
- Authorized Users Register
- Courses Register
- Exams Register
- Groups Register
- Lecturers Register
- Link of Interest Register
- Non-working days Register
- Notifications Register
- Sessions Register
- Students Register
- Students Logged in Application Register
- Teaches Register
- Update Configurations Register
- Update Results Register
- LDAP Users Register

These are detailed below:

Item	Size	Type
Code	255	Alphanumeric

Table 45: EIISERVER. Academic Degrees Archive

Item	Size	Type
Code	255	Alphanumeric
Year	2	Numeric

Table 46: EIISERVER. Academic Years Archive

Item	Size	Type
Group Code	255	Alphanumeric
Course Code	255	Alphanumeric
Academic Degree Code	255	Alphanumeric
UO	255	Alphanumeric

Table 47: EIISERVER. Attendances Archive

Item	Size	Type
Username	255	Alphanumeric
Password	255	Alphanumeric
Organization	255	Alphanumeric

Table 48: EIISERVER. Authenticated Users Archive

Item	Size	Type
Username	255	Alphanumeric
Role	255	Alphanumeric

Table 49: EIISERVER. Authorized Users Archive

Item	Size	Type
Academic Degree Code	255	Alphanumeric
Course Code	255	Alphanumeric
Name	255	Alphanumeric
SIES Code	255	Alphanumeric
Year	2	Numeric

Table 50: EIISERVER. Courses Archive

Item	Size	Type
Exam Code	255	Numeric
Course Code	255	Alphanumeric
Name	255	Alphanumeric
Description	255	Alphanumeric
Starting Date	12	Date
Ending Date	12	Date
Starting Time	24	DateTime
Ending Time	24	DateTime
Location	255	Alphanumeric

Table 51: EIISERVER. Exams Archive

Item	Size	Type
Group Code	255	Alphanumeric
Course Code	255	Alphanumeric
Academic Degree Code	255	Alphanumeric
Type	255	Alphanumeric

Table 52: EIISERVER. Groups Archive

Item	Size	Type
Email	255	Alphanumeric
Name	255	Alphanumeric

Table 53: EIISERVER. Lecturers Archive

Item	Size	Type
Content	65000	Alphanumeric

Table 54: EIISERVER. Link of Interest Archive

Item	Size	Type
Non-working Day Code	255	Numeric
Day	12	Date

Table 55: EIISERVER. Non-working Days Archive

Item	Size	Type
Title	255	Alphanumeric
Content	255	Alphanumeric
Receivers	255	Alphanumeric

Table 56: EIISERVER. Notifications Archive

Item	Size	Type
Session Code	255	Numeric
Group Code	255	Alphanumeric
Course Code	255	Alphanumeric
Academic Degree Code	255	Alphanumeric
Name	255	Alphanumeric
Description	255	Alphanumeric
Starting Date	12	Date
Ending Date	12	Date
Starting Time	24	DateTime
Ending Time	24	DateTime
Location	255	Alphanumeric

Table 57: EIISERVER. Sessions Archive

Item	Size	Type
UO	255	Alphanumeric

Table 58: EIISERVER. Students Archive

Item	Size	Type
UO	255	Alphanumeric
Token	255	Alphanumeric

Table 59: EIISERVER. Students Logged in Application Archive

Item	Size	Type
Group Code	255	Alphanumeric
Course Code	255	Alphanumeric
Academic Degree Code	255	Alphanumeric
Email	255	Alphanumeric

Table 60: EIISERVER. Teaches Archive

Item	Size	Type
Daily Time	24	DateTime
Weekly Time	24	DateTime
Weekly Day	2	Numeric
Option	255	Alphanumeric

Table 61: EIISERVER. Update Configurations Archive

Item	Size	Type
Date	24	DateTime
Time	24	DateTime
Content	255	Alphanumeric
Success	1	Boolean

Table 62: EIISERVER. Update Results Archive

Item	Size	Type
Username	255	Alphanumeric
Password	255	Alphanumeric

Table 63: EIISERVER. LDAP Users Archive

9.3.1.1.2 COUNTING PROCESS

9.3.1.1.2.1 INPUTS

- Register Academic Degree Screen
- Register Academic Year Screen
- Register Attendance Screen
- Register Authorized User Screen
- Register Course Screen
- Register Exam Screen
- Register Group Screen
- Register Lecturer Screen
- Edit Link of Interest Screen
- Register Non-working day Screen
- Send Notification Screen
- Register Session Screen
- Register Student Screen
- Register Teaches Screen
- Schedule Update Screen
- Update Screen

9.3.1.1.2.2 OUTPUTS

- Exams
- Link of Interest
- Sessions
- Students

9.3.1.1.2.3 QUERIES

- Query Academic Degrees
- Query Academic Years
- Query Attendances
- Query Authorized Users
- Query Courses

- Query Exams
- Query Groups
- Query Lecturers
- Query Non-working days
- Query Sessions
- Query All Students
- Query All Students that belong to an Academic Degree
- Query All Students that belong to an Academic Year
- Query All Students that belong to a Course
- Query All Students that belong to a Group
- Query All Students that belong to a Lecturer
- Query Teaches

9.3.1.1.2.4 LOGICAL FILES

- Academic Degrees Register
- Academic Years Register
- Attendances Register
- Authenticated Users Register
- Authorized Users Register
- Courses Register
- Exams Register
- Groups Register
- Lecturers Register
- Link of Interest Register
- Non-working days Register
- Notifications Register
- Sessions Register
- Students Register
- Students Logged in Application Register
- Teaches Register
- Update Configurations Register
- Update Results Register

9.3.1.1.2.5 EXTERNAL INTERFACES

- LDAP Users Register

9.3.1.1.3 CLASSIFICATION OF FUNCTIONS

To classify the functions, we will use the table detailed on **DPPI course** [24], and shown on Table 64: Estimation of size and effort. Weight Factor.

Measure parameter	Weight Factor		
	BASIC	MEDIUM	COMPLEX
Number of Inputs	3	4	6
Number of Outputs	4	5	7
Number of Logical Files	7	10	15
Number of Queries	3	4	6
Number of External Interfaces	5	7	10

Table 64: Estimation of size and effort. Weight Factor

- 16 inputs (Medium)
- 4 outputs (Medium)

- 18 logical files (Medium)
- 17 queries (Basic)
- 1 external interface (Basic)

Which is translated, applying the formula shown in Figure 148: Estimation of size and effort. Function Counter Formula, into **CF** = **320**

$$CF = \sum_{i=1}^{5} \sum_{j=1}^{3} w_{ij} x_{ij}$$

Figure 148: Estimation of size and effort. Function Counter Formula

9.3.1.1.3.1 CALCULATION OF THE ADJUSTMENT FACTOR

Factor	Score (0 -5)
Does the system require reliable backups and recovery?	**2**
Is data communication required?	**4**
Are there any distributed processing functions?	**0**
Is performance critical?	**2**
Would the system run in an existing and heavily used operating environment?	**5**
Does the system require interactive data entry?	**5**
Does interactive data entry require input transactions to be carried out on multiple screens or operations?	**0**
Are logical files updated interactively?	**0**
Are the inputs, outputs, files, or requests complex?	**0**
Is the internal processing complex?	**1**
Is the code designed to be reusable?	**5**
Are conversion and installation included in the design?	**4**
Has the system been designed to support multiple installations in different organizations?	**0**
Has the application been designed to facilitate changes and to be easily used by the user?	**5**

Table 65: EIISERVER. Calculation of the adjustment factor

In that case, $\mathbf{c_k}$ = 0.65 + 0.01 * 33 = **0.98**, and **PF** = **313,6**.

9.3.2 EIIAPP

9.3.2.1 MODULE DEFINITION

9.3.2.1.1 FUNCTIONALITIES

- Authenticates users against EIISERVER.
- Authenticates users against Google OAuth.
- Receives notifications from EIISERVER.
- Allows a user to consult Links of Interest Page.
- Allows a user to consult events.
- Allows a user to export events.
- Allows a user to report a breakdown.
- Allows a user to consult notifications.
- Updates the stored information.
- Schedules updates.

9.3.2.1.1.1 ARCHIVES

The system will need the following archives:

- Events Register
- Link of Interest Register
- Notifications Register
- Students Register
- Update Configurations Register
- Update Results Register

These are detailed below:

Item	Size	Type
Group Code	255	Alphanumeric
Course Code	255	Alphanumeric
Name	255	Alphanumeric
Description	255	Alphanumeric
Starting Date	12	Date
Ending Date	12	Date
Starting Time	24	DateTime
Ending Time	24	DateTime
Location	255	Alphanumeric

Table 66: EIIAPP. Events Archive

Item	Size	Type
Content	65000	Alphanumeric

Table 67: EIIAPP. Link of Interest Archive

Item	Size	Type
Title	255	Alphanumeric
Content	255	Alphanumeric
Receivers	255	Alphanumeric

Table 68: EIIAPP. Notifications Archive

Item	Size	Type
UO	255	Alphanumeric

Table 69: EIIAPP. Students Archive

Item	Size	Type
UO	255	Alphanumeric
Token	255	Alphanumeric

Table 70: EIIAPP. Students Logged in Application Archive

Item	Size	Type
Daily Time	24	DateTime
Weekly Time	24	DateTime
Weekly Day	2	Numeric
Option	255	Alphanumeric

Table 71: EIIAPP. Update Configurations Archive

Item	Size	Type
Date	24	DateTime
Time	24	DateTime
Content	255	Alphanumeric
Success	1	Boolean

Table 72: EIIAPP. Update Results Archive

9.3.2.1.2 COUNTING PROCESS

9.3.2.1.2.1 INPUTS

- Links of Interest Screen
- Events Screen
- Notifications Screen
- Export Events Screen
- Report Breakdown Screen
- Schedule Update Screen
- Update Screen

9.3.2.1.2.2 OUTPUTS

- Events

9.3.2.1.2.3 QUERIES

- Query Events
- Query Links of Interest
- Query Notifications

9.3.2.1.2.4 LOGICAL FILES

- Notifications Register
- Update Configurations Register
- Update Results Register

9.3.2.1.2.5 EXTERNAL INTERFACES

- Events Register
- Link of Interest Register
- Students Register

9.3.2.1.3 CLASSIFICATION OF FUNCTIONS

To classify the functions, we will use the table detailed on **DPPI course** [24], and shown on Table 64: Estimation of size and effort. Weight Factor.

- 7 inputs (Basic)
- 1 output (Basic)
- 3 logical files (Basic)
- 3 queries (Basic)
- 3 external interfaces (Basic)

Which is translated, applying the formula shown in Figure 148: Estimation of size and effort. Function Counter Formula, into **CF = 70**

9.3.2.1.3.1 CALCULATION OF THE ADJUSTMENT FACTOR

Factor	Score (0 -5)
Does the system require reliable backups and recovery?	**2**
Is data communication required?	**4**
Are there any distributed processing functions?	**0**
Is performance critical?	**3**
Would the system run in an existing and heavily used operating environment?	**5**
Does the system require interactive data entry?	**3**
Does interactive data entry require input transactions to be carried out on multiple screens or operations?	**0**
Are logical files updated interactively?	**0**
Are the inputs, outputs, files, or requests complex?	**0**
Is the internal processing complex?	**1**
Is the code designed to be reusable?	**5**
Are conversion and installation included in the design?	**4**
Has the system been designed to support multiple installations in different organizations?	**0**
Has the application been designed to facilitate changes and to be easily used by the user?	**5**

Table 73: EIIAPP. Calculation of the adjustment factor

In that case, $\mathbf{c_k}$ = 0.65 + 0.01 * 32 = **0.97**, and **PF** = **67.9**.

9.3.3 EFFORT

The effort estimation of EIISERVER and EIIAPP is shown in Table 74: EIISERVER. Effort Estimation and Table 75: EIIAPP. Effort Estimation, respectively.

Effort Estimation of EIISERVER	
Most Likely	200
Maximum	250
Minimum	150

Table 74: EIISERVER. Effort Estimation

Effort Estimation of EIIAPP	
Most Likely	120
Maximum	180
Minimum	100

Table 75: EIIAPP. Effort Estimation

In this case, the value considered is obtained by calculating the arithmetic mean of 4 times the most probable, the minimum and the maximum. Note that we have only one person in the Development Team and just two modules. $\mathbf{E_{EIISERVER}}$ = 200 an $\mathbf{E_{EIIAPP}}$ = 126.

9.4 BUDGETING

In this annex, budgeting will be deconstructed and analysed.

9.4.1 ENTERPRISE DEFINITION

In this section, **Enterprise Definition** will be detailed. **Note that all profiles that appear here are the ones that Adrián Vaz Sánchez will play on the planning and execution of EIIProject**: Software Engineer, Software Architect, among others. All numbers have been taken from **PayScale** [25], setting Spain as job location.

Enterprise definition is structured in tables between Table 76: Budgeting. Enterprise Definition. Summary and Table 80: Budgeting. Enterprise Definition.

Direct plus Indirect Costs	227.360,25 €
Desired benefits (25%)	56.840,06 €
Billing needs	284.200,31 €

Table 76: Budgeting. Enterprise Definition. Summary

Indirect Costs

Service	Monthly Cost	Annual cost
Cleaning	30,00 €	360,00 €
Renting fees	60,00 €	720,00 €
Electricity consumption (except consumption for production)	50,00 €	600,00 €
Fuel consumption for heating	30,00 €	360,00 €
Communications expenses	10,00 €	120,00 €
Expenses on office supplies	5,00 €	60,00 €
TOTAL		2.220,00 €

Table 77: Budgeting. Enterprise Definition. Indirect Costs

Costs of the means of production

Equipment/License	Units	Price	Total Cost	Annual Cost	Type	Time Limit
Development Equipment	1	50,00 €	50,00 €	12,50 €	Amortization	4
Development Licenses (Enterprise, Project, etc.)	1	32,75 €	32,75 €	32,75 €	Rental	-
Telecommunication Licenses (Microsoft Teams)	1	10,50 €	126,00 €	126,00 €	Rental	-
Office Licenses (Office 365 E5)	1	5,75 €	69,00 €	69,00 €	Rental	-
TOTAL				240,25 €		

Table 78: Budgeting. Enterprise Definition. Costs of the means of production

Price / hour (cost and sale)

Number	CONCEPT	AMOUNT
1	Direct Costs Total	141.570,00 €
2	Indirect Costs Total	85.790,25 €
3	Sum of direct and indirect costs	227.360,25 €
4	Desired benefits (25%)	56.840,06 €
5	Total Cost (sum of direct costs, indirect costs and benefits)	284.200,31 €
6	Billing based on production hours and calculated hourly prices	286.220,50 €
7	Margin between Total Cost and Billing (ratio between 5 and 6)	0,71%

Table 79: Budgeting. Enterprise Definition. Price/hour (cost and sale)

Enterprise Definition	Staff Productivity	Productive and non-productive hours	Price / hour (cost and sale)

Profile	Nº	Gross Annual Salary	Annual Salary Cost	TOTAL	Prod (%)	Direct Cost	IC (%)	Indirect Cost	Hours/ year	Productive hours / year (per person)	Total productive hours	Price/ hour	Billing	Price / hour (without benefits)
Project Manager	1	30.000,00 €	39.000,00 €	39.000,00 €	0,00 %	0,00 €	100,00%	39.000,00 €	-	-	-	-	-	-
Software Engineer	1	21.000,00 €	27.300,00 €	27.300,00 €	75,00 %	20.475,00 €	25,00%	6.825,00 €	1784	1338	1338	32,50 €	43.485,00 €	26,00 €
Software Analyst	1	25.000,00 €	32.500,00 €	32.500,00 €	75,00 %	24.375,00 €	25,00%	8.125,00 €	1784	1338	1338	37,50 €	50.175,00 €	30,00 €
Requirements Engineer	1	21.000,00 €	27.300,00 €	27.300,00 €	75,00 %	20.475,00 €	25,00%	6.825,00 €	1784	1338	1338	32,50 €	43.485,00 €	26,00 €
Software Architect	1	27.000,00 €	35.100,00 €	35.100,00 €	75,00 %	26.325,00 €	25,00%	8.775,00 €	1784	1338	1338	43,75 €	58.537,50 €	35,00 €
Developer	1	18.000,00 €	23.400,00 €	23.400,00 €	80,00 %	18.720,00 €	20,00%	4.680,00 €	1784	1427	1427	17,50 €	24.976,00 €	14,00 €
Test Engineer	1	16.000,00 €	20.800,00 €	20.800,00 €	75,00 %	15.600,00 €	25,00%	5.200,00 €	1784	1338	1338	25,00 €	33.450,00 €	20,00 €
Tester	1	15.000,00 €	19.500,00 €	19.500,00 €	80,00 %	15.600,00 €	20,00%	3.900,00 €	1784	1427	1427	22,50 €	32.112,00 €	18,00 €
TOTAL	**8**			**224.900,00 €**		**141.570,00 €**		**83.330,00 €**			**9544**	**211,25 €**	**286.220,50 €**	**169,00 €**

Table 80: Budgeting. Enterprise Definition

9.4.2 HARDWARE (INSTALLATION)

In this section, **Hardware** Budget Entry is detailed (see Table 81: Budgeting. Hardware (Installation)). Note that only Installation takes place (no Hardware acquisition, due to previously mentioned reasons throughout this document).

I1	I2	I3	I4	Description	Amount	Units	Price	Subtotal (3)	Subtotal (2)	Total
01				**Hardware Installation**						**2.210,00 €**
	001			**Study of the needs of the client**					1.300,00 €	
		01		Requirements Engineer	20	hours	32,50 €	650,00 €		
		02		Software Engineer	20	hours	32,50 €	650,00 €		
	002			**Installation of the server**					910,00 €	
		01		**Server Configuration**						
			01	Software Engineer	12	hours	32,50 €	390,00 €		
		02		**Deployment of EIISERVER on the server**						
			01	Software Engineer	16	hours	32,50 €	520,00 €		
Total										**2.210,00 €**

Table 81: Budgeting. Hardware (Installation)

9.4.3 SYSTEM PLANNING

System Planning Entry is detailed on Table 82: Budgeting. System Planning.

I1	I2	I3	Description	Amount	Units	Price	Subtotal (3)	Subtotal (2)	Total
01			**Planning of the information system**						2.233,75 €
	001		**Definition and Organization of the PSI**					0,00 €	
		01	Project Manager	13	hours	0,00 €	0,00 €		
	002		**Study of relevant information**					562,50 €	
		01	Software Analyst	15	hours	37,50 €	562,50 €		
02			**Definition of the technological architecture**						
	001		**Identification of the needs of the technological infrastructure**					306,25 €	

		01	Software Architect	7	hours	43,75 €	306,25 €		
	002		**Selection of the architecture**					875,00 €	
		01	Software Architect	20	hours	43,75 €	875,00 €		
03			**Feasibility Study of the system**					305,00 €	
		01	Project Manager	4	hours	0,00 €	0,00 €		
		02	Software Architect	4	hours	43,75 €	175,00 €		
		03	Software Engineer	4	hours	32,50 €	130,00 €		
04			**Planning and management of the project**						
	001		**Identification of the stakeholders**					87,50 €	
		01	Project Manager	5	hours	0,00 €	0,00 €		
		02	Requirements Engineer	2	hours	43,75 €	87,50 €		
	002		**OBS**					0,00 €	
		01	Project Manager	8	hours	0,00 €	0,00 €		
	003		**PBS**					0,00 €	
		01	Project Manager	8	hours	0,00 €	0,00 €		
	004		**WBS**					97,50 €	
		01	Software Engineer	3	hours	32,50 €	97,50 €		
		02	Project Manager	15	hours	0,00 €	0,00 €		
	005		**Risks**					0,00 €	
		01	Project Manager	16	hours	0,00 €	0,00 €		
	006		**Budget**					0,00 €	
		01	Project Manager	20	hours	0,00 €	0,00 €		

05			**Project Closure**					0,00 €	
		01	Project Manager	16	hours	0,00 €	0,00 €		
Total									**2.233,75 €**

Table 82: Budgeting. System Planning

9.4.4 SYSTEM STUDY

System Study is detailed in this section (see tables from Table 83: Budgeting. System Study. Summary to Table 87: Budgeting. System Study (Introduction and Acceptance)).

System Study		
Cod	**Item**	**Total**
01	**System Analysis**	18.767,50 €
02	**System Design**	15.550,00 €
03	**System Construction**	15.060,00 €
04	**System Introduction and Acceptance**	130,00 €
Total Cost		**49.507,50 €**

Table 83: Budgeting. System Study. Summary

System Analysis

I1	I2	I3	I4	I5	Description	Amount	Units	Price	Subtotal (2)	Subtotal (1)	Total
01					**System Analysis**						18.767,50 €
	001				**System Definition**					650,00 €	
				01	Requirements Engineer	10	hours	32,50 €	325,00 €		
				02	Software Engineer	10	hours	32,50 €	325,00 €		
	002				**Requirements Establishment**					1.040,00 €	
				01	Requirements Engineer	16	hours	32,50 €	520,00 €		
				02	Software Engineer	16	hours	32,50 €	520,00 €		
	003				**EIISERVER Module Identification**					11.887,50 €	
		01			**Logger Module Analysis**						
			01		**LoggerManager Module Analysis**						
				01	Software Analyst	12	hours	37,50 €	450,00 €		
		02			**Datasources Module Analysis**						
			01		**LDAPAuthenticationDatasource Module Analysis**						
				01	Software Analyst	30	hours	37,50 €	1.125,00 €		
			02		**PUSHNotificationDatasource Module Analysis**						

				01	Software Analyst	30	hours	37,50 €	1.125,00 €		
			03		**EntityDatasource Module Analysis**						
				01	Software Analyst	30	hours	37,50 €	1.125,00 €		
		03			**Core Module Analysis**						
			01		**Entities Module Analysis**						
				01	Software Analyst	8	hours	37,50 €	300,00 €		
			02		**Interactors Module Analysis**						
				01	Software Analyst	20	hours	37,50 €	750,00 €		
		04			**Transport Module Analysis**						
			01		**Loaders Module Analysis**						
				01	Software Analyst	10	hours	37,50 €	375,00 €		
			02		**Middlewares Module Analysis**						
				01	Software Analyst	10	hours	37,50 €	375,00 €		
			03		**GIISOF Readers Module Analysis**						
				01	Software Analyst	30	hours	37,50 €	1.125,00 €		
			04		**MIW Readers Module Analysis**						
				01	Software Analyst	30	hours	37,50 €	1.125,00 €		

			05		**Workers Module Analysis**						
				01	Software Analyst	28	hours	37,50 €	1.050,00 €		
			06		**Routes Module Analysis**						
				01	Software Analyst	28	hours	37,50 €	1.050,00 €		
		05			**Factories Module Analysis**						
			01		**ErrorHandlerFactory Module Analysis**						
				01	Software Analyst	7	hours	37,50 €	262,50 €		
			02		**LoggerFactory Module Analysis**						
				01	Software Analyst	7	hours	37,50 €	262,50 €		
			03		**LoaderFactory Module Analysis**						
				01	Software Analyst	7	hours	37,50 €	262,50 €		
			04		**ReaderFactory Module Analysis**						
				01	Software Analyst	7	hours	37,50 €	262,50 €		
			05		**InteractorFactory Module Analysis**						
				01	Software Analyst	7	hours	37,50 €	262,50 €		
			06		**RepositoryFactory Module Analysis**						
				01	Software Analyst	8	hours	37,50 €	300,00 €		

			07		**WorkerFactory Module Analysis**						
				01	Software Analyst	8	hours	37,50 €	300,00 €		
	004				**EIIAPP Module Identification**					2.287,50 €	
		01			**Core Module Analysis**						
			01		**Models Module Analysis**						
				01	Software Analyst	7	hours	37,50 €	262,50 €		
			02		**Services Module Analysis**						
				01	Software Analyst	7	hours	37,50 €	262,50 €		
			03		**ViewModels Module Analysis**						
				01	Software Analyst	7	hours	37,50 €	262,50 €		
		02			**Datasources Module Analysis**						
				01	Software Analyst	16	hours	37,50 €	600,00 €		
		03			**UI Module Analysis**						
			01		**Views Module Analysis**						
				01	Software Analyst	8	hours	37,50 €	300,00 €		
			02		**Widgets Module Analysis**						
				01	Software Analyst	8	hours	37,50 €	300,00 €		

		04			**Locators Module Analysis**						
				01	Software Analyst	8	hours	37,50 €	300,00 €		
	005				**Class Analysis**					560,00 €	
				01	Software Analyst	8	hours	37,50 €	300,00 €		
				02	Software Engineer	8	hours	32,50 €	260,00 €		
	006				**Data modelling**					1.072,50 €	
				01	Software Engineer	33	hours	32,50 €	1.072,50 €		
	007				**Defining User Interfaces**					520,00 €	
				01	Software Engineer	16	hours	32,50 €	520,00 €		
	008				**Testing Plan Specification**					750,00 €	
				01	Test Engineer	30	hours	25,00 €	750,00 €		

Table 84: Budgeting. System Study (Analysis)

System Design

I1	I2	I3	I4	I5	Description	Amount	Units	Price	Subtotal (2)	Subtotal (1)	Total
01					**System Design**						15.550,00 €
	001				**Class Design**					227,50 €	
				01	Software Engineer	7	hours	32,50 €	227,50 €		
	002				**EIISERVER Use Case Design**					10.172,50 €	
		01			**Selection of Use Cases of Logger**						
			01		**Selection of Use Cases of LoggerManager**						
				01	Software Engineer	28	hours	32,50 €	910,00 €		
		02			**Selection of Use Cases of Datasources**						
			01		**Selection of Use Cases of LDAPAuthenticationDatasource**						
				01	Software Engineer	30	hours	32,50 €	975,00 €		
			02		**Selection of Use Cases of PUSHNotificationDatasource**						
				01	Software Engineer	30	hours	32,50 €	975,00 €		
			03		**Selection of Use Cases of EntityDatasource**						
				01	Software Engineer	30	hours	32,50 €	975,00 €		
		03			**Selection of Use Cases of Core**						
			01		**Selection of Use Cases of Entities**						

				01	Software Engineer	26	hours	32,50 €	845,00 €		
			02		**Selection of Use Cases of Interactors**						
				01	Software Engineer	12	hours	32,50 €	390,00 €		
		04			**Selection of Use Cases of Transport**						
			01		**Selection of Use Cases of Loader**						
				01	Software Engineer	10	hours	32,50 €	325,00 €		
			02		**Selection of Use Cases of Middlewares**						
				01	Software Engineer	12	hours	32,50 €	390,00 €		
			03		**Selection of Use Cases of GIISOF Readers**						
				01	Software Engineer	20	hours	32,50 €	650,00 €		
			04		**Selection of Use Cases of MIW Readers**						
				01	Software Engineer	20	hours	32,50 €	650,00 €		
			05		**Selection of Use Cases of Workers**						
				01	Software Engineer	18	hours	32,50 €	585,00 €		
			06		**Selection of Use Cases of Routes**						
				01	Software Engineer	28	hours	32,50 €	910,00 €		
		05			**Selection of Use Cases of Factories**						
			01		**Selection of Use Cases of ErrorHandlerFactory**						
				01	Software Engineer	7	hours	32,50 €	227,50 €		
			02		**Selection of Use Cases of LoggerFactory**						
				01	Software Engineer	7	hours	32,50 €	227,50 €		

			03		**Selection of Use Cases of LoaderFactory**						
				01	Software Engineer	7	hours	32,50 €	227,50 €		
			04		**Selection of Use Cases of ReaderFactory**						
				01	Software Engineer	7	hours	32,50 €	227,50 €		
			05		**Selection of Use Cases of InteractorFactory**						
				01	Software Engineer	7	hours	32,50 €	227,50 €		
			06		**Selection of Use Cases of RepositoryFactory**						
				01	Software Engineer	7	hours	32,50 €	227,50 €		
			07		**Selection of Use Cases of WorkerFactory**						
				01	Software Engineer	7	hours	32,50 €	227,50 €		
	003				**EIIAPP Use Case Design**					2.665,00 €	
		01			**Selection of Use Cases of Core**						
			01		**Selection of Use Cases of Models**						
				01	Software Engineer	12	hours	32,50 €	390,00 €		
			02		**Selection of Use Cases of Services**						
				01	Software Engineer	12	hours	32,50 €	390,00 €		
			03		**Selection of Use Cases of ViewModels**						
				01	Software Engineer	14	hours	32,50 €	455,00 €		
		02			**Selection of Use Cases of Datasources**						
				01	Software Engineer	10	hours	32,50 €	325,00 €		
		03			**Selection of Use Cases of UI**						

			01		**Selection of Use Cases of Views**						
				01	Software Engineer	12	hours	32,50 €	390,00 €		
			02		**Selection of Use Cases of Widgets**						
				01	Software Engineer	14	hours	32,50 €	455,00 €		
		04			**Selection of Use Cases of Locators**						
				01	Software Engineer	8	hours	32,50 €	260,00 €		
	004				**System module architecture design**					875,00 €	
				01	Software Architect	20	hours	43,75 €	875,00 €		
	005				**Physical Data Design**					910,00 €	
				01	Software Engineer	28	hours	32,50 €	910,00 €		
	006				**Technical Specification of the Testing Plan**					700,00 €	
				01	Test Engineer	28	hours	25,00 €	700,00 €		

Table 85: Budgeting. System Study (Design)

System Construction

I1	I2	I3	I4	I5	Description	Amount	Units	Price	Subtotal (2)	Subtotal (1)	Total
01					**System Construction**						15.060,00 €
	001				**Generation of the code of the components and procedures of EIISERVER**					8.050,00 €	
		01			**Development of Logger**						
			01		**Development of LoggerManager**						
				01	Developer	32	hours	17,50 €	560,00 €		
		02			**Development of Datasources**						
			01		**Development of LDAPAuthenticationDatasource**						
				01	Developer	38	hours	17,50 €	665,00 €		
			02		**Development of PUSHNotificationDatasource**						
				01	Developer	37	hours	17,50 €	647,50 €		
			03		**Development of EntityDatasource**						
				01	Developer	28	hours	17,50 €	490,00 €		
		03			**Development of Core**						
			01		**Development of Entities**						
				01	Developer	2	hours	17,50 €	35,00 €		
			02		**Development of Interactors**						
				01	Developer	63	hours	17,50 €	1.102,50 €		
		04			**Development of Transport**						

			01		**Development of Loader**						
				01	Developer	28	hours	17,50 €	490,00 €		
			02		**Development of Middlewares**						
				01	Developer	36	hours	17,50 €	630,00 €		
			03		**Development of GIISOF Readers**						
				01	Developer	47	hours	17,50 €	822,50 €		
			04		**Development of MIW Readers**						
				01	Developer	26	hours	17,50 €	455,00 €		
			05		**Development of Workers**						
				01	Developer	5	hours	17,50 €	87,50 €		
			06		**Development of Routes**						
				01	Developer	68	hours	17,50 €	1.190,00 €		
		05			**Development of Factories**						
			01		**Development of ErrorHandlerFactory**						
				01	Developer	7	hours	17,50 €	122,50 €		
			02		**Development of LoggerFactory**						
				01	Developer	7	hours	17,50 €	122,50 €		
			03		**Development of LoaderFactory**						
				01	Developer	7	hours	17,50 €	122,50 €		
			04		**Development of ReaderFactory**						
				01	Developer	7	hours	17,50 €	122,50 €		

			05		**Development of InteractorFactory**						
				01	Developer	7	hours	17,50 €	122,50 €		
			06		**Development of RepositoryFactory**						
				01	Developer	7	hours	17,50 €	122,50 €		
			07		**Development of WorkerFactory**						
				01	Developer	8	hours	17,50 €	140,00 €		
	002				**Generation of the code of the components and procedures of EIIAPP**					4.235,00 €	
		01			**Development of Core**						
			01		**Development of Models**						
				01	Developer	12	hours	17,50 €	210,00 €		
			02		**Development of Services**						
				01	Developer	45	hours	17,50 €	787,50 €		
			03		**Development of ViewModels**						
				01	Developer	45	hours	17,50 €	787,50 €		
		02			**Development of Datasources**						
				01	Developer	46	hours	17,50 €	805,00 €		
		03			**Development of UI**						
			01		**Development of Views**						
				01	Developer	37	hours	17,50 €	647,50 €		
			02		**Development of Widgets**						

				01	Developer	45	hours	17,50 €	787,50 €		
		04			**Development of Locators**						
				01	Developer	12	hours	17,50 €	210,00 €		
	003				**EIISERVER Tests Execution**					900,00 €	
		01			**EIISERVER Unit tests execution**						
				01	Tester	20	hours	22,50 €	450,00 €		
		02			**EIISERVER Acceptance tests execution**						
				01	Tester	20	hours	22,50 €	450,00 €		
	004				**EIIAPP Tests Execution**					900,00 €	
		01			**EIIAPP Unit tests execution**						
				01	Tester	14	hours	22,50 €	315,00 €		
		02			**EIIAPP Widget tests execution**						
				01	Tester	13	hours	22,50 €	292,50 €		
		03			**EIIAPP Integration tests execution**						
				01	Tester	13	hours	22,50 €	292,50 €		
	005				**User Manuals Elaboration**					975,00 €	
				01	Software Engineer	30	hours	32,50 €	975,00 €		

Table 86: Budgeting. System Study (Construction)

System Introduction and Acceptance										
I1	I2	I3	I4	Description	Amount	Units	Price	Subtotal (3)	Subtotal (2)	Total
01				**System Introduction and Acceptance**						**130,00 €**
	001			**Establishment of the Introduction Plan**					0,00 €	
			01	Project Manager	14	hours	0,00 €	0,00 €		
	002			**Uploading Data to the Operating Environment**					130,00 €	
			01	Project Manager	12	hours	0,00 €	0,00 €		
			02	Software Engineer	4	hours	32,50 €	130,00 €		
	003			**Presentation and approval of the system and going into production**					0,00 €	
			01	Project Manager	12	hours	0,00 €	0,00 €		

Table 87: Budgeting. System Study (Introduction and Acceptance)

9.4.5 OTHER COSTS

Other Costs can be seen in Table 88: Budgeting. Other Costs.

I1	I2	I3	Description	Amount	Units	Price	Subtotal (3)	Subtotal (2)	Total
01			**Travel and Expense Allowance**						**30,00 €**
	001							30,00 €	
		01	Travel (50 Km to Project Presentation location)	50	Km	0,20 €	10,00 €		
		02	Expense Allowance (1 person x 1 day)	1	Expense Allowance	20,00 €	20,00 €		
Total									**30,00 €**

Table 88: Budgeting. Other Costs

9.4.6 COST BUDGET SUMMARY

Thereby, Cost Budget Summary is illustrated in Table 89: Budgeting. Cost Budget Summary.

Cod.	Entry	Total
01	**Hardware (Installation)**	**2.210,00 €**
02	**System Planning**	**2.233,75 €**
03	**System Study**	**49.507,50 €**
03	**Other Costs (Travel and Expense Allowance)**	**30,00 €**
Total		**53.981,25 €**

Table 89: Budgeting. Cost Budget Summary

9.5 FINAL BUDGETING

9.5.1 ENTERPRISE DEFINITION

The same one illustrated in Budgeting.

9.5.2 HARDWARE (INSTALLATION)

In this section, **Hardware** Budget Entry is detailed (see Table 90: Final Budgeting. Hardware (Installation)).

I1	I2	I3	I4	Description	Amount	Units	Price	Subtotal (3)	Subtotal (2)	Total
01				**Hardware Installation**						**2.015,00 €**
	001			**Study of the needs of the client**					455,00 €	
		01		Requirements Engineer	7	hours	32,50 €	227,50 €		
		02		Software Engineer	7	hours	32,50 €	227,50 €		
	002			**Installation of the server**					1.560,00 €	
		01		**Server Configuration**						
			01	Software Engineer	32	hours	32,50 €	1.040,00 €		
		02		**Deployment of EIISERVER on the server**						
			01	Software Engineer	16	hours	32,50 €	520,00 €		
Total										**2.015,00 €**

Table 90: Final Budgeting. Hardware (Installation)

9.5.3 SYSTEM PLANNING

The same one illustrated in Budgeting.

9.5.4 SYSTEM STUDY

System Analysis, System Design and System Introduction and Acceptance are the same ones illustrated in Budgeting. On the other hand, System Construction is detailed on Table 91: Final Budgeting. System Construction.

System Construction

I1	I2	I3	I4	I5	Description	Amount	Units	Price	Subtotal (2)	Subtotal (1)	Total
01					**System Construction**						17.472,50 €
	001				**Generation of the code of the components and procedures of EIISERVER**					9.747,50 €	
		01			**Development of Logger**						
			01		**Development of LoggerManager**						
				01	Developer	45	hours	17,50 €	787,50 €		
		02			**Development of Datasources**						
			01		**Development of LDAPAuthenticationDatasource**						
				01	Developer	45	hours	17,50 €	787,50 €		
			02		**Development of PUSHNotificationDatasource**						
				01	Developer	44	hours	17,50 €	770,00 €		
			03		**Development of EntityDatasource**						
				01	Developer	32	hours	17,50 €	560,00 €		
		03			**Development of Core**						
			01		**Development of Entities**						
				01	Developer	7	hours	17,50 €	122,50 €		
			02		**Development of Interactors**						
				01	Developer	90	hours	17,50 €	1.575,00 €		
		04			**Development of Transport**						
			01		**Development of Loader**						
				01	Developer	34	hours	17,50 €	595,00 €		
			02		**Development of Middlewares**						

				01	Developer	56	hours	17,50 €	980,00 €		
			03		**Development of GIISOF Readers**						
				01	Developer	52	hours	17,50 €	910,00 €		
			04		**Development of MIW Readers**						
				01	Developer	36	hours	17,50 €	630,00 €		
			05		**Development of Workers**						
				01	Developer	9	hours	17,50 €	157,50 €		
			06		**Development of Routes**						
				01	Developer	76	hours	17,50 €	1.330,00 €		
		05			**Development of Factories**						
			01		**Development of ErrorHandlerFactory**						
				01	Developer	7	hours	17,50 €	122,50 €		
			02		**Development of LoggerFactory**						
				01	Developer	1	hours	17,50 €	17,50 €		
			03		**Development of LoaderFactory**						
				01	Developer	7	hours	17,50 €	122,50 €		
			04		**Development of ReaderFactory**						
				01	Developer	7	hours	17,50 €	122,50 €		
			05		**Development of InteractorFactory**						
				01	Developer	7	hours	17,50 €	122,50 €		
			06		**Development of RepositoryFactory**						
				01	Developer	1	hours	17,50 €	17,50 €		
			07		**Development of WorkerFactory**						
				01	Developer	1	hours	17,50 €	17,50 €		

	002				**Generation of the code of the components and procedures of EIIAPP**					5.337,50 €	
		01			**Development of Core**						
			01		**Development of Models**						
				01	Developer	9	hours	17,50 €	157,50 €		
			02		**Development of Services**						
				01	Developer	55	hours	17,50 €	962,50 €		
			03		**Development of ViewModels**						
				01	Developer	50	hours	17,50 €	875,00 €		
		02			**Development of Datasources**						
				01	Developer	69	hours	17,50 €	1.207,50 €		
		03			**Development of UI**						
			01		**Development of Views**						
				01	Developer	50	hours	17,50 €	875,00 €		
			02		**Development of Widgets**						
				01	Developer	60	hours	17,50 €	1.050,00 €		
		04			**Development of Locators**						
				01	Developer	12	hours	17,50 €	210,00 €		
	003				**EIISERVER Tests Execution**					1.012,50 €	
		01			**EIISERVER Unit tests execution**						
				01	Tester	15	hours	22,50 €	337,50 €		
		02			**EIISERVER Acceptance tests execution**						
				01	Tester	30	hours	22,50 €	675,00 €		
	004				**EIIAPP Tests Execution**					562,50 €	

		01			**EIIAPP Unit tests execution**						
				01	Tester	4	hours	22,50 €	90,00 €		
		02			**EIIAPP Widget tests execution**						
				01	Tester	6	hours	22,50 €	135,00 €		
		03			**EIIAPP Integration tests execution**						
				01	Tester	15	hours	22,50 €	337,50 €		
	005				**User Manuals Elaboration**					812,50 €	
				01	Software Engineer	25	hours	32,50 €	812,50 €		

Table 91: Final Budgeting. System Construction

Finally, System Study Summary is detailed in Table 92: Final Budgeting. System Study.

System Study		
Cod	**Item**	**Total**
01	**System Analysis**	18.767,50 €
02	**System Design**	15.550,00 €
03	**System Construction**	17.472,50 €
04	**System Introduction and Acceptance**	32,50 €
Total Cost		**51.822,50 €**

Table 92: Final Budgeting. System Study

9.5.5 OTHER COSTS

The same ones illustrated in Budgeting.

9.5.6 COST BUDGET SUMMARY

Cost Budget Summary is illustrated in Table 93: Final Budgeting. Cost Budget Summary.

Cost Budget (Summary)		
Cod.	**Entry**	**Total**
01	**Hardware (Installation)**	**2.015,00 €**
02	**System Planning**	**2.233,75 €**
03	**System Study**	**51.822,50 €**
03	**Other Costs (Travel and Expense Allowance)**	**30,00 €**
Total		**56.101,25 €**

Table 93: Final Budgeting. Cost Budget Summary

9.6 EXTENSIONS

In this section, **future work** (extensions) will be commented. There were not done due to requirements inflation: the clients suggested these extensions when EIISERVER was already deployed and EIIAPP approved:

- Allowing a user authenticated on EIIAPP to see a plan of School's classrooms: When clicking on the location of an event presented in the application, "A-2-01", for example, EIIAPP shall open the following plan: http://gis.uniovi.es/GISUniovi/GeoLoc.do?codEspacio=01.01.01.00.P2.00.05.
- Allowing a user authenticated on EIIAPP to delete notifications
- Adding a notification subject to notifications:
    - If a user authenticated on EIISERVER sends a message to all students of CVVS course with "Title1" as title and "Content1" as content, a student of CVVS authenticated on EIIAPP shall receive the next notification:
        - Title: **[CVVS] Title1**
        - Content: **Content1**

10 REFERENCES

[1] J. M. Redondo López, "Creación y evaluación de plantillas para trabajos de fin de grado como buena práctica docente," *Revista de Innovación y Buenas Prácticas Docentes*, 2020.

[2] J. M. Redondo López, "Documentos-modelo para Trabajos de Fin de Grado/Master de la Escuela de Informática de Oviedo," *https://www.researchgate.net/publication/327882831_Plantilla_de_Proyectos_de_Fin_de_Carrera_de_la_Escuela_de_Informatica_de_Oviedo*, Jun. 2019.

[3] School of Computer Science, "Software Engineering Degree Exams," *https://calendar.google.com/calendar/ical/cmrcsvl23b1ah0h0ej9gj6h*.

[4] School of Computer Science, "Assignment of groups of Theory, Seminars and Laboratory Practices for Software Engineering Degree," *http://gobierno.euitio.uniovi.es/grado/gd/?y=21-22&t=s1*.

[5] School of Computer Science, "Timetables for Software Engineering Degree Courses," *http://gobierno.ingenieriainformatica.uniovi.es/grado/plan/?y=21-22&t=s1*.

[6] Robert C. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design: A Craftsman's Guide to Software Structure and Design*. Addison-Wesley; N.º 1, 2017.

[7] Netflix, "Ready for changes with Hexagonal Architecture," *https://netflixtechblog.com/ready-for-changes-with-hexagonal-architecture-b315ec967749*, Oct. 03, 2020.

[8] "NPMTrends," *https://www.npmtrends.com/typescript*.

[9] Gobierno de España, "Métrica v.3," *https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html*.

[10] Tekartik (Mobile & Web development), "sqflite," *https://pub.dev/packages/sqflite*.

[11] "PostgreSQL Documentation: Connections," *https://www.postgresql.org/docs/13/runtime-config-connection.html*.

[12] "Material Design," *https://material.io/*.

[13] E. H. R. J. R. V. J. Gamma, *Design Patterns: Elements of Reusable Object-Oriented Software*. 1994.

[14] "Node.js Best Practices," *https://github.com/goldbergyoni/nodebestpractices*.

[15] "Effective Dart: Documentation," *https://dart.dev/guides/language/effective-dart/documentation*.

[16] HiveDB, "Hive," *https://pub.dev/packages/hive*.

[17] Jest, "Jest (JavaScript Testing Framework)," *https://jestjs.io/es-ES/*.

[18] Cucumber, "Cucumber (BDD Testing & Collaboration Tools for Teams)," *https://cucumber.io/*.

[19] Flutter, "Flutter Testing," *https://flutter.dev/docs/testing*.

[20] Google, "Google TypeScript Style Guide," *https://google.github.io/styleguide/tsguide.html*.

[21] "Effective Dart: Style," *https://dart.dev/guides/language/effective-dart/style*.

[22] "IISNode," *https://github.com/Azure/iisnode*.

[23] B. Boehm, "Software Risk Management: Principles and Practices," Jan. 1991.

[24] A. A. and L. P. B. Juan Fuente, *Guía de Aprendizaje de la asignatura de Dirección y Planificación de Proyectos Informáticos*. 2020.

[25] "PayScale," *https://www.payscale.com/research/ES/Country=Spain/Salary*.

11 CONTENTS DELIVERED

The contents delivered are detailed in Table 94: Contents Delivered.

Directory	Content
./README.txt	Explains the structure detailed in this table
./EIISERVER	Root of EIISERVER development project directory structure (see Table 95: Contents Delivered. EIISERVER).
./EIIAPP	Root of EIIAPP development project directory structure (see Table 96: Contents Delivered. EIIAPP).
./DOCUMENTATION	Root of Documentation directory structure (see Table 97: Contents Delivered. Documentation).
./EXPLOIT	Contains the script used to generate the database (please see "Installation Manual")

Table 94: Contents Delivered

Directory	Content
certificates	Certificates used in EIISERVER
coverage	Coverage report and information
private	Configuration files and keys used in EIISERVER
resources	Files publicly exposed in EIISERVER
sql	SQL scripts used to initialize EIISERVER persistence (PostgreSQL).
src	Source code of EIISERVER
tests	All tests of EIISERVER
views	Html views used in EIISERVER

Table 95: Contents Delivered. EIISERVER

Directory	Content
android	Folder generated by Dart. Use to release EIIAPP on Android devices
assets	Images and icons used throughout EIIAPP
coverage	Coverage information
integration_test	Integration tests
ios	Folder generated by Dart. It would be used if EIIAPP was released on iOS devices.
lib	Source code of EIIAPP
test	All unit and widget tests
web	Folder generated by Dart. It would be used if EIIAPP was deployed.

Table 96: Contents Delivered. EIIAPP

Directory	Content
DIAGRAMS	Contains main diagrams displayed in this document
MANAGEMENT	Contains: • WBS – PBS illustrations displayed in this document • Budget (xlsx) • Planning (mpp)
MANUALS	Contains: • Execution Manual • Installation Manual • Maintenance and Deployment Manual • Programmer Manual • User Manual

Table 97: Contents Delivered. Documentation

12 EIIPROJECT CONCLUSIONS

12.1 CONCLUSIONS

EIISERVER, my fist deployed project for a real client, has been a turning point in my perception of Software Projects.

It has revealed that the theory we have learned in many courses of the School of Computer Science, specially **DPPI** [24], is present in all projects (included EIIProject). As Figure 149: Project Management Triangle shows, **Quality is not free** and needs a balance between Scope, Cost and Time, i.e., resources are limited.

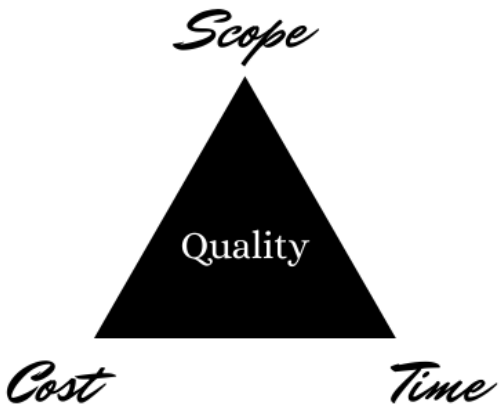


Figure 149: Project Management Triangle

On the other hand, as we have learned from **Requirements Engineering**, knowing **what we need to do** is the key part of every project that can lead us to the success as well as to the failure if no required attention is being paid.

And, ultimately, that is one of the **definitions of Project Management**: **leading our project to the success at all costs**.

This project, like any other, has suffered from lots of up and downs and all kind of situations that have documented in previous chapters, but the most remarkable fact is that, thanks to the lead of my tutors: **Luis Antonio Vinuesa Martínez** and **Fernando Álvarez García**, the knowledge gained from my time at the School and a good dose of courage and creativity to carry out this project, EIIProject has come to fruition, and I hope it leaves a mark on the university community of our School.

12.2 CONCLUSIONES

EIISERVER, mi primer proyecto desplegado para un cliente real, ha supuesto un punto de inflexión en mi percepción de los Proyectos Software.

Ha revelado que la teoría estudiada en varias asignaturas de la Escuela de Ingeniería Informática, especialmente **DPPI** [24], está presente en todos los proyectos (incluido EIIProject). Como muestra la Figura 1: Triángulo de la Gestión de Proyectos, **la Calidad no es gratuita** y necesita un equilibrio entre: Alcance, Coste y Tiempo, es decir, los recursos son limitados.

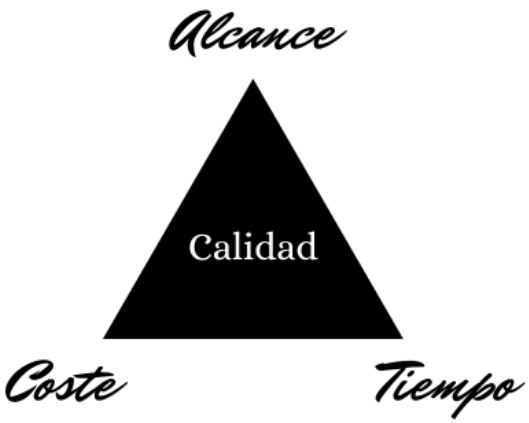


Figura 1: Triángulo de la Gestión de Proyectos

Por otra parte, como hemos aprendido de la **Ingeniería de Requisitos**, saber **qué hay que hacer** es la parte primordial de todo proyecto que puede llevarnos al éxito, así como al fracaso si no prestamos la atención requerida.

Y, fundamentalmente, esta es **una de las definiciones de la Gestión de Proyectos: guiar nuestro proyecto al éxito a toda costa**.

Este proyecto, como cualquier otro, ha sufrido diversos vaivenes y todo tipo de situaciones que han sido documentadas en capítulos anteriores, pero el hecho más reseñable es que, gracias al liderazgo de mis tutores: **Luis Antonio Vinuesa Martínez** y **Fernando Álvarez García**, el conocimiento obtenido de mi paso por la Escuela y una buena dosis de creatividad y esfuerzo para sacar adelante este proyecto, EIIProject ha llegado a buen término, y espero que deje una huella en la comunidad universitaria de nuestra Escuela.