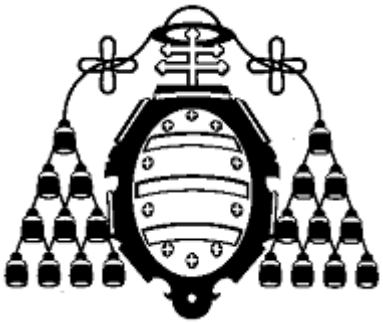


UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

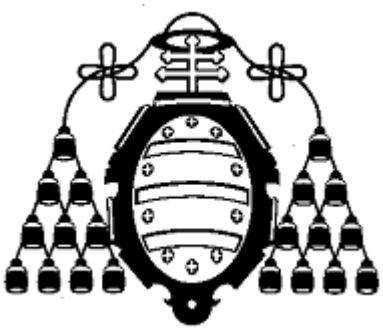
MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

HERRAMIENTA DE GESTIÓN DE EXPEDIENTES PARA EL CAT

ZEUS PÉREZ GARCÍA
JUNIO 2013

ÍNDICE



UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

HERRAMIENTA DE GESTIÓN DE EXPEDIENTES PARA EL CAT

DOCUMENTO Nº I

MEMORIA

ZEUS PÉREZ GARCÍA
JUNIO 2012

ÁREA DE TELEMÁTICA
TUTOR: XICU XABIEL GARCÍA PAÑEDA

ÍNDICE DE LA MEMORIA

ÍNDICE DE FÍGURAS DE LA MEMORIA

MEMORIA

1. Introducción

1.1. Identificación del proyecto

Título: Herramienta de Gestión de Expedientes para el CAT

Número de proyecto:

Área proponente: Departamento de Informática

Tutor: Xicu Xabiel García Pañeda

Autor: Zeus Pérez García

Fecha: Junio 2013

1.2. Planteamiento del problema

El proyecto está pensado para dar soporte a los servicios proporcionados por el Instituto Servicios Sociales, en concreto por el Centro de Accesibilidad y Ayudas Técnicas (CAT). Estos servicios se encargan de atender y gestionar solicitudes sobre demandas relativas a accesibilidad y ayudas técnicas durante todo el ciclo de vida de la solicitud.

Para la gestión de los servicios proporcionados por el CAT, se manejan tres tipos de elementos documentales principalmente, que permitirán el seguimiento, control, estudio y almacenamiento de toda la información relativa a las solicitudes presentadas. Estos tres elementos principales son: solicitudes, expedientes e intervenciones. A partir de una solicitud, se asignara al expediente de la persona beneficiaria de esa solicitud, creándose en caso de ser la primera solicitud de dicha persona. Además la solicitud creará un proceso con una intervención a personal experto de la Administración para el estudio de dicha solicitud a través de la cual se resolverá si aceptar o rechazar la solicitud. En la actualidad para realizar dicho procedimiento se usa principalmente expedientes en papel y una aplicación obsoleta y muy limitada, que no cubre las necesidades actuales para la gestión de estos servicios, siendo muy difícil la administración y consulta de la información almacenada.

El objetivo de esta aplicación es que a partir de una interfaz que permita introducir los datos necesarios mediante formularios, los trabajadores responsables del CAT puedan introducir y consultar la información que deseen según los requisitos existentes en la prestación de los servicios a gestionar.

En la introducción de información en el sistema también está presente el factor humano, por lo tanto pueden surgir otro tipo de inconvenientes como introducción de datos erróneos, duplicidad, errores en la toma de datos, etc. Todos estos percances no deseados se deberá tener en cuenta a la hora de diseñar el nuevo sistema para conseguir un resultado lo mas óptimo posible.

1.3. Objetivos del proyecto

El objetivo del proyecto es el diseño e implementación de un sistema que permita el almacenamiento y el acceso a la información relativa a la gestión de expediente según las necesidades del Centro de Accesibilidad y Ayudas Técnicas.

Para lograr este objetivo, se deberá diseñar una interfaz en la aplicación que permita obtener la información necesaria a través de formularios de entrada debidamente conectados con una base de datos diseñada para almacenar los datos necesarios relativos a la gestión de expedientes. El sistema debe permitir además la realización de funciones adicionales para mejorar el servicio que se da a los ciudadanos: funciones de consulta, estadísticas, control documental y generación de informes.
Además se deberá realizar con tecnologías que permitan la extender el sistema, ya que en un futuro puede extenderse dicho sistema para dar cobertura a más servicios ofrecidos por el Instituto de Cántabro de Servicios Sociales.

Todo esto debe estar diseñado de manera que resulte intuitivo y de fácil manejo para los usuarios, ya que el usuario principal al que está dirigido no tiene conocimientos de informática avanzados.

2. Descripción de los servicios ofrecidos

Para facilitar el entendimiento de la funcionalidad y un mejor uso del sistema, se procederá a explicar en rasgos generales los servicios ofrecidos por el Centro de Accesibilidad y Ayudas Técnicas que se pretenden gestionar.

2.1. Instituto de Servicios Sociales

Los Servicios Sociales se han modernizado estos últimos años y en la fase más actual han cambiado y se han descentralizado de manera que existen diversos organismos que dan soporte a estos. El centro que se pretende gestionar forma parte de un instituto de servicios sociales que es el encargado de seguir prestando los servicios sociales que hoy recibe la ciudadanía, además de la incorporación nuevos servicios y prestaciones que responderán a las nuevas necesidades de los ciudadanos.

Dicho instituto es además una herramienta para la coordinación entre distintos departamentos de servicios sociales, las entidades locales y la iniciativa privada en el desarrollo de unos nuevos servicios sociales modernos y entendidos como derecho de ciudadanía.

Para afrontar de la mejor manera posible el derecho a la protección social por los servicios sociales se requiere una Administración ágil que dé respuestas rápidas y eficaces a las necesidades de la población. El ISS es un paso decisivo en esta dirección: un instrumento para la gestión efectiva de unos servicios sociales modernos al servicio de la ciudadanía.

El ISS nace como consecuencia del nuevo sistema de servicios sociales como resultado de diversas leyes y directivas sobre Derechos y Servicios Sociales. Este nuevo modelo parte del principio de universalidad en el acceso y se articula por una cartera de servicios en la que se establecen las prestaciones del sistema público de servicios sociales.
Esta institución concreta el derecho de las personas situadas dentro de su territorio de demarcación a disfrutar de los servicios sociales y, por tanto, refleja el paso de un sistema asistencial de los servicios sociales, a un sistema de acceso universal con derechos concretos reconocidos por toda la población.

Para la gestión efectiva de los Servicios incluidos en dicha cartera y para ayudar en la rapidez de respuesta y efectividad, el ISS se apoya en soluciones informáticas diseñadas específicamente para cubrir los requisitos de cada servicio en concreto.

2.2. Centro de Accesibilidad y Ayudas Técnicas

El Centro de Accesibilidad y Ayudas Técnicas es un Centro dependiente del Instituto Servicios Sociales, y que pertenece a la red de centros de asesoramiento e información (CAI) que coordina el Centro Estatal de Autonomía Personal y Ayudas Técnicas (CEAPAT).Y forma parte de la red de Centros de Información y Productos de Apoyo (IPROA) del Ministerio de Sanidad y Política Social.

Es un centro que dedica sus actividades a potenciar actuaciones que favorezcan la calidad de vida mediante la autonomía personal, con un interés especial en personas que padezcan algún tipo de discapacidad y a personas mayores, apostando por la accesibilidad integral, el diseño para todos y la tecnología de apoyo para favorecer la vida independiente.

Para la prestación de los servicios propios del CAT, dicho centro se divide en las siguientes áreas:

- Coordinación y dirección
- Accesibilidad
- Productos de apoyo
- Comunicación
- Asesoramiento institucional
- Formación
- Gestión de programas

Los servicios prestados por el CAT son:

- Información y asesoramiento en productos de apoyo, con una exposición permanente de los productos más demandados.
- Asesoramiento para el acceso al ordenador y las nuevas tecnologías, contando con una unidad de demostración de equipos accesibles.
- Organización de jornadas de puertas abiertas, visitas guiadas y presentación de nuevos productos accesibles, para la difusión y el conocimiento de las últimas novedades en el campo de las adaptaciones para personas con discapacidad.
- Apoyo técnico para el cumplimiento de la legislación vigente en materia de eliminación de barreras arquitectónicas, urbanísticas, del transporte y la comunicación.
- Formación de profesionales y estudiantes en el ámbito de la autonomía personal.
- Proyectos de investigación en productos de apoyo, accesibilidad y eliminación de barreras.
- Apoyo a las familias, con la elaboración y distribución de material didáctico sobre cuidados y atención a personas dependientes, favoreciendo la formación dentro del grupo familiar sobre prevención, terapias y cuidados a familiares de personas con discapacidad o enfermos crónicos gravemente afectados.
- Documentación especializada en el campo de las ayudas técnicas, la autonomía personal, la tecnología de apoyo y la accesibilidad ofreciendo servicios de consulta y préstamo.

La forma de solicitud para la prestación de los servicios ofrecidos son:

- De forma presencial, previa petición de hora.
- De forma escrita.
- De forma telefónica o telemática.
- Para visitas guiadas a la exposición es necesario solicitar cita previa.
- Participación en actividades formativas organizadas por otras entidades.
- Organización de actividades formativas o divulgativas.

2.3. Sistema de Gestión de Expedientes

A continuación se describe el proceso de gestión de expedientes utilizado por el CAT para la prestación de los servicios:

El sistema necesita guardar toda la información relacionada con las solicitudes de prestación de servicios: datos de los solicitantes, datos de los beneficiarios, información sobre el expediente que relaciona a los beneficiarios con las solicitudes realizadas y las intervenciones asociadas a dichas solicitudes, datos de las solicitud y datos y documentación en caso de que exista de la/s intervenciones.

Inicialmente un beneficiario o solicitante no existen en el sistema hasta que hayan realizado una solicitud. En ese momento se crea el expediente relacionado a ese beneficiario guardando toda la información relativa al mismo. En caso de que el solicitante no exista también se guardará la información del mismo, pero no generará ningún expediente nuevo. Una vez creado se asociará la solicitud a dicho expediente, y posteriormente, cuando se valore la solicitud y se decida la intervención a realizar, se asociará dicha intervención, con todos sus cambios y su posible documentación asociada.
En caso de que el beneficiario ya exista, simplemente se asociará la nueva solicitud al expediente existente, añadiendo los datos de la intervención a realizar una vez se resuelva.

Todo esto atendiendo a unos requisitos que están explicados en la especificación de los requisitos del sistema.

3. Descripción del sistema

3.1. Introducción

La aplicación permite la gestión de expedientes a partir de datos introducidos a través de formularios. La aplicación organizará la información de manera adecuada para poder disponer de ella de la manera más efectiva posible, permitiendo al usuario realizar consultas, obtener estadísticas, gestionar los documentos relacionados y generar los informes necesarios para su almacenamiento en papel.

3.2. Requisitos del sistema

La aplicación almacenará los datos relativos a los expedientes permitiendo conservar la integridad de los mismos y la disponibilidad a la hora de consultando, relacionando automáticamente según dicta el diseño los distintos elementos pertenecientes al expediente.

La aplicación tendrá como entrada distintos formularios predefinidos que guiarán al usuario a la hora de introducir la información, obligando a introducir al usuario los datos que son necesarios y permitiendo añadir información adicional que puede resultar útil en el estudio de los expedientes.

El sistema debe ser todo lo automático posible, en cuanto al almacenamiento de datos y la organización de los mismos de acuerdo con los requisitos del servicio proporcionado.

3.3. Usuarios y seguridad

La aplicación posee una capa de seguridad que esta predefinida para que se guarden los usuarios y los roles en una base de datos conectada con la aplicación y así acceder a la misma mediante una interfaz de usuario/clave típica de aplicaciones web.

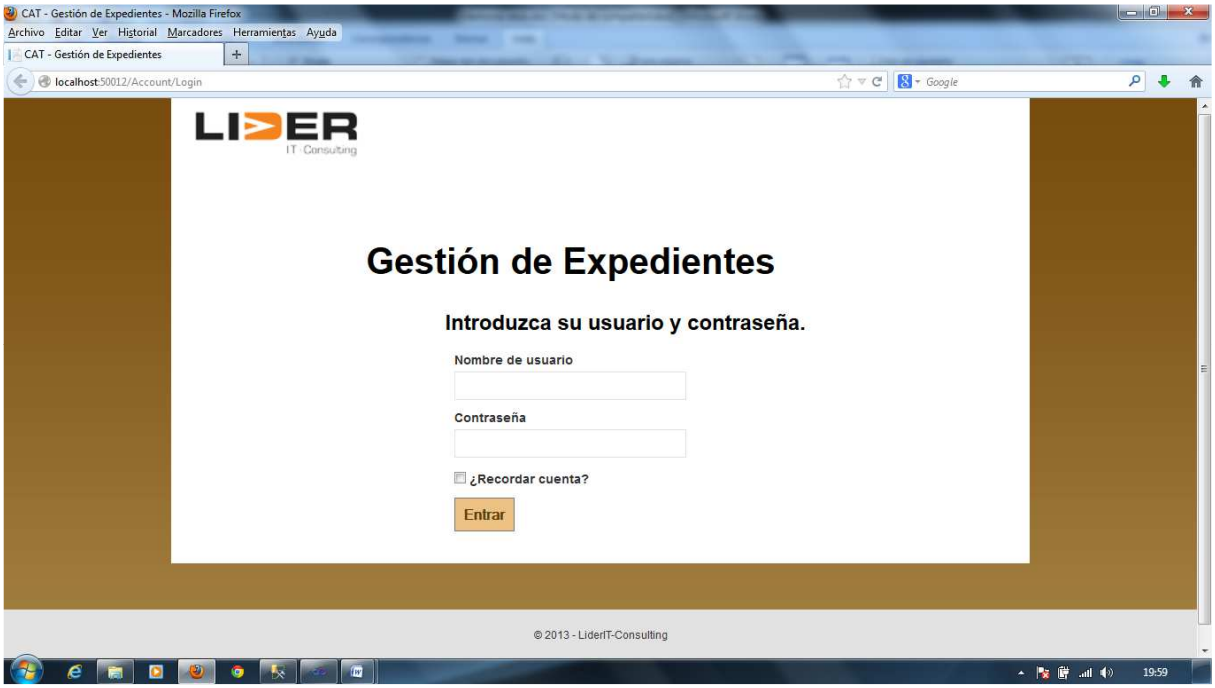


Figura 1. Pantalla de identificación

Además, se permite cambiar el control de acceso a la aplicación para que sea mediante el directorio activo del usuario de Windows que está utilizando el computador. Para ello se necesita acceder con el rol de administrador.

El sistema está previsto para contemplar dos tipos distintos de roles: usuario y administrado. Aunque un usuario administrador puede añadir más roles en el futuro mediante las funciones de administración.

3.4. Interfaz del sistema

La aplicación es una aplicación web y por lo tanto tiene una interfaz web con los elementos comunes de las mismas. Posee menús que permiten el acceso a las distintas funciones de la aplicación y formularios web en aquellas funciones que requieren entrada de datos.

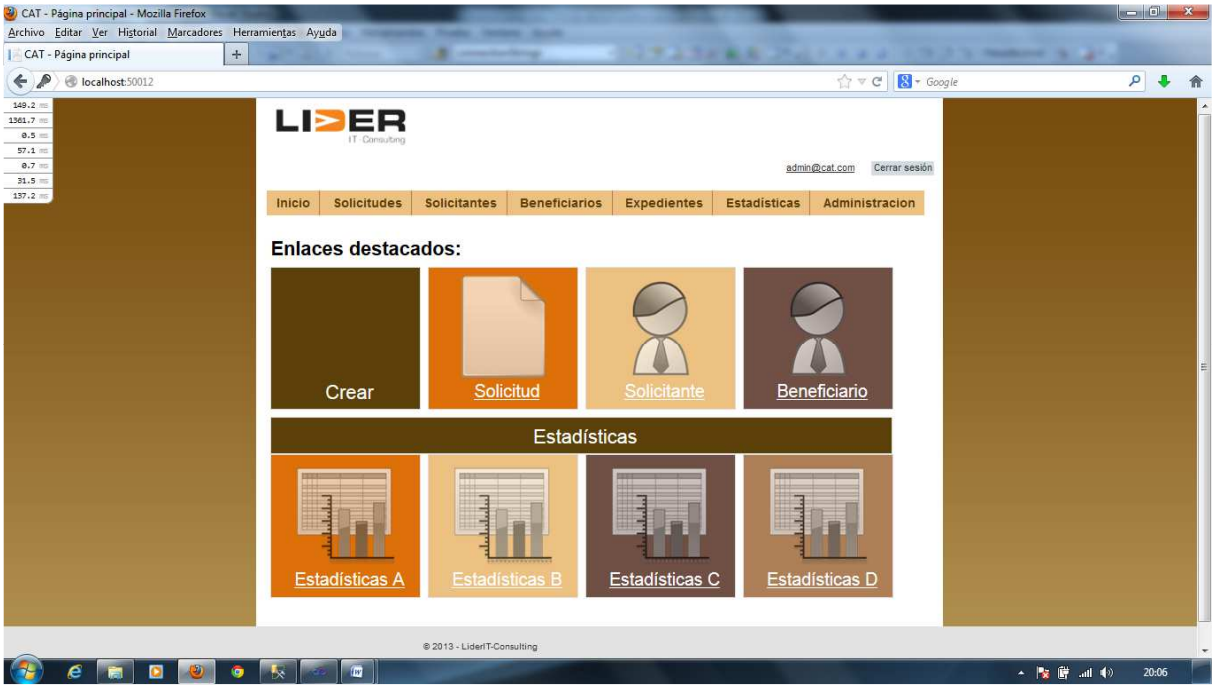


Figura 2. Pantalla de inicio

3.5. Datos de entrada y de salida del sistema

Como se ha dicho previamente el sistema permite la entrada de datos a través de formularios predeterminados, que corresponden con solicitudes utilizadas en formato de papel por el CAT. Dichas solicitudes están obsoletas y se ha añadido al formulario más información que resulta de interés en la actualidad.

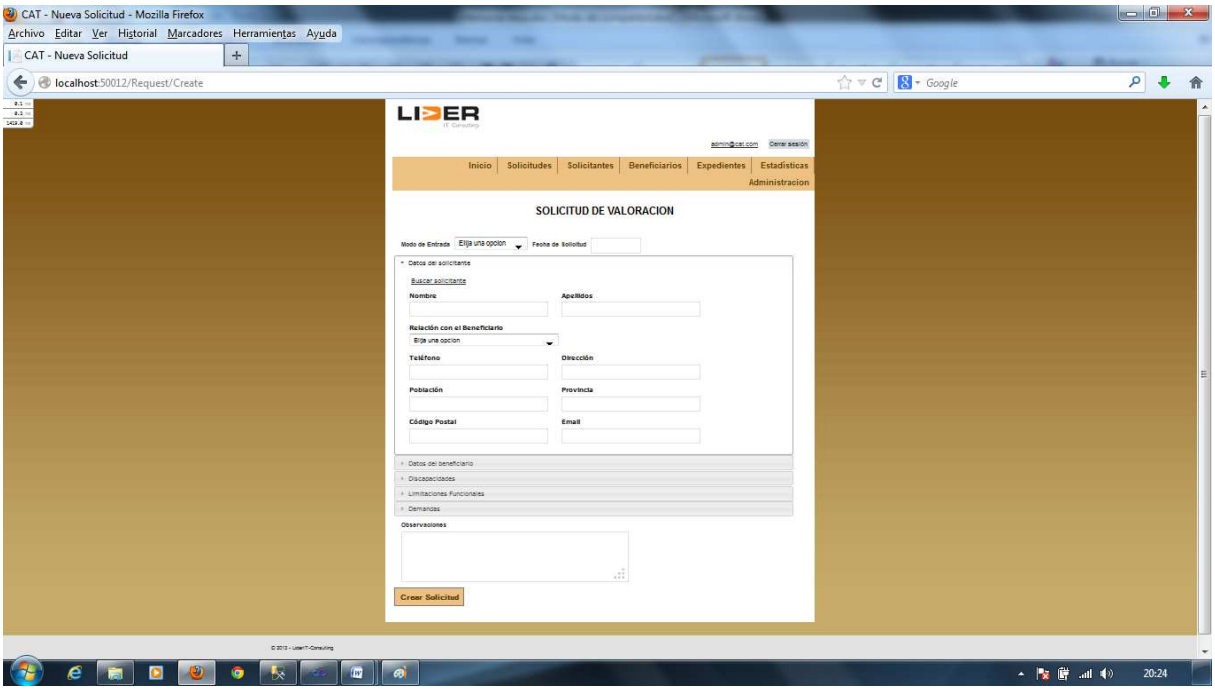


Figura 3. Solicitud-Datos del solicitante

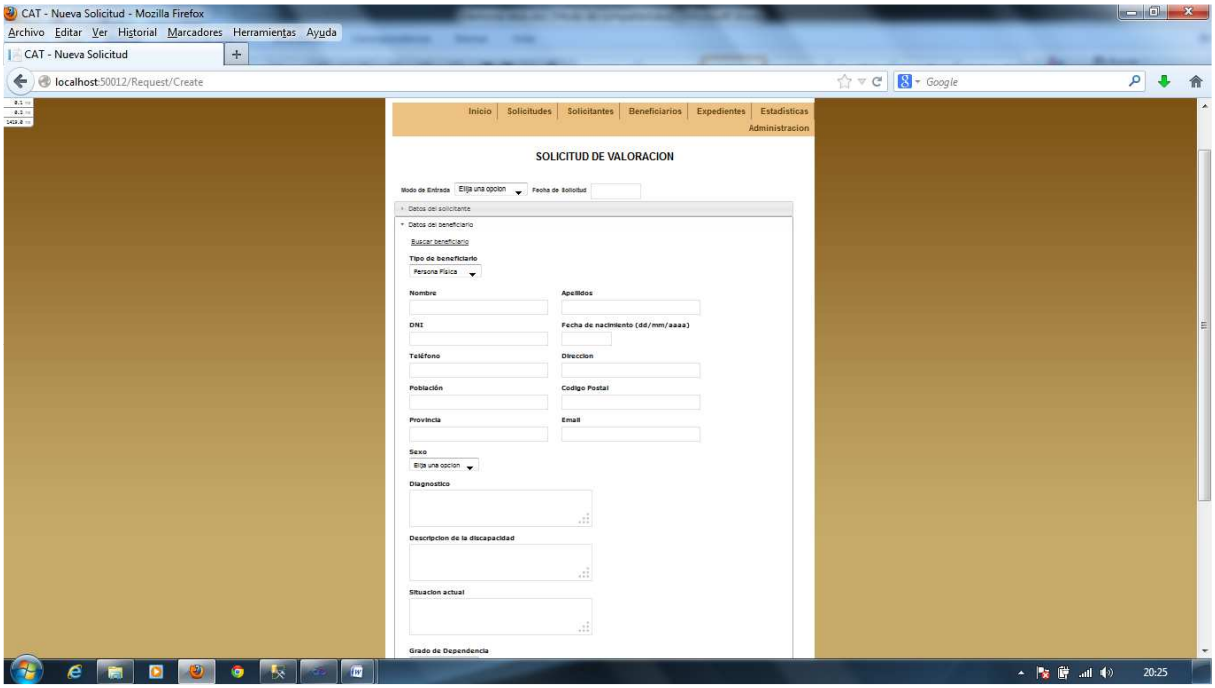


Figura 4. Solicitud-Datos de beneficiario

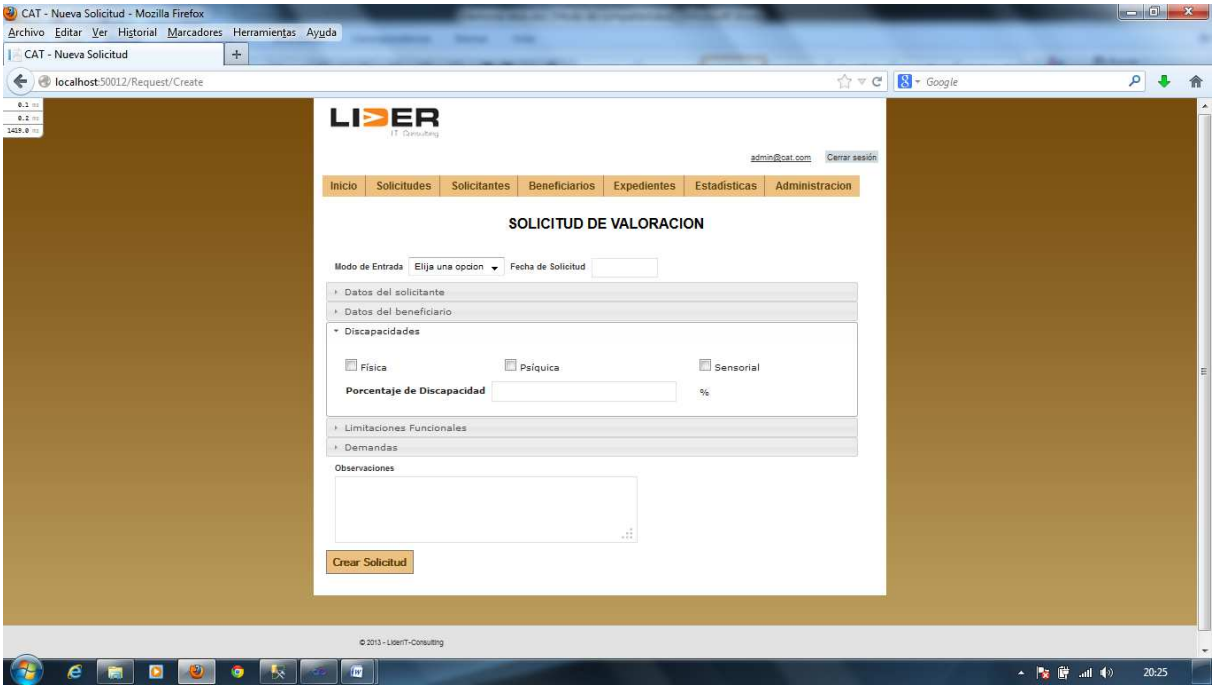


Figura 5. Solicitud-Discapacidades

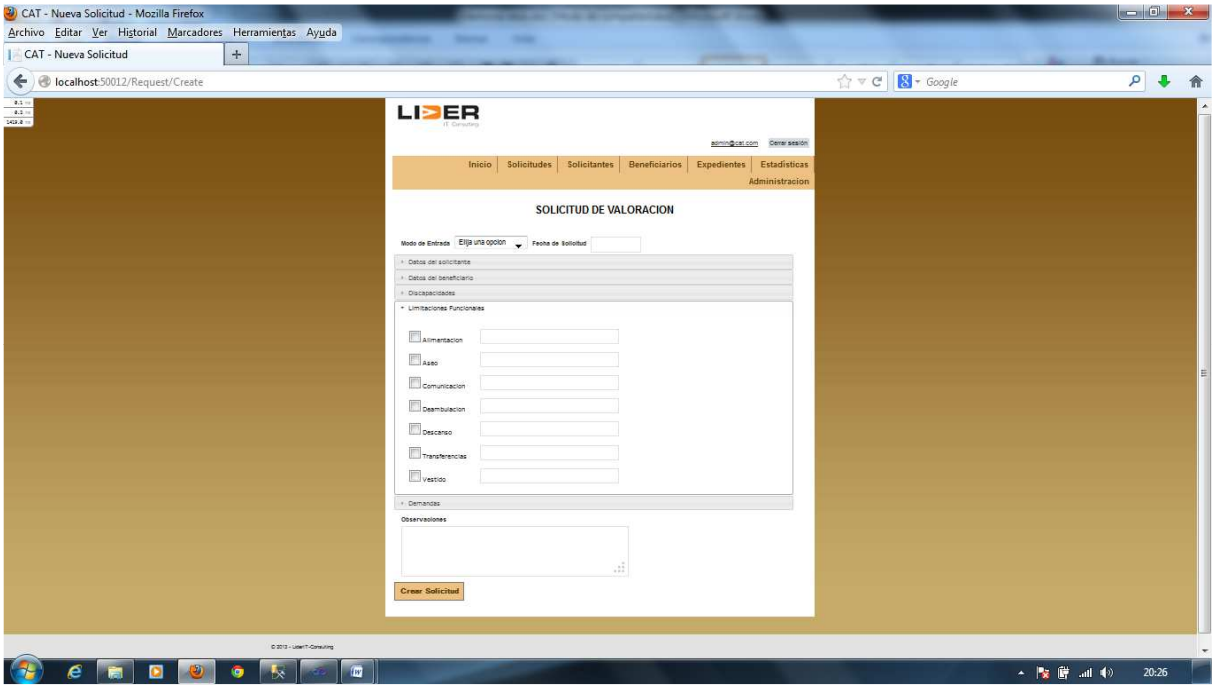


Figura 6. Solicitudes-Limitaciones funcionales

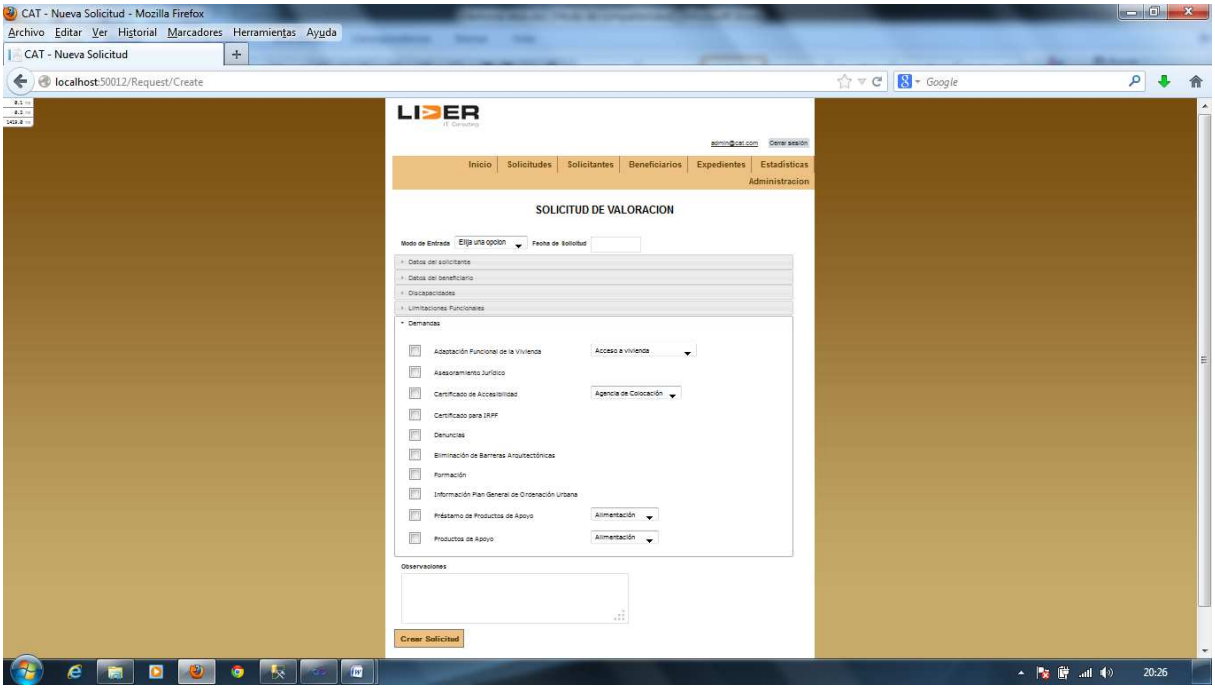


Figura 7. Solicitudes-Demandas

Además se permite también introducir datos de las intervenciones a través de otro formulario específico, que en la actualidad no existe en papel.

También se puede introducir criterios de búsqueda para consultar la información almacenada en el sistema.

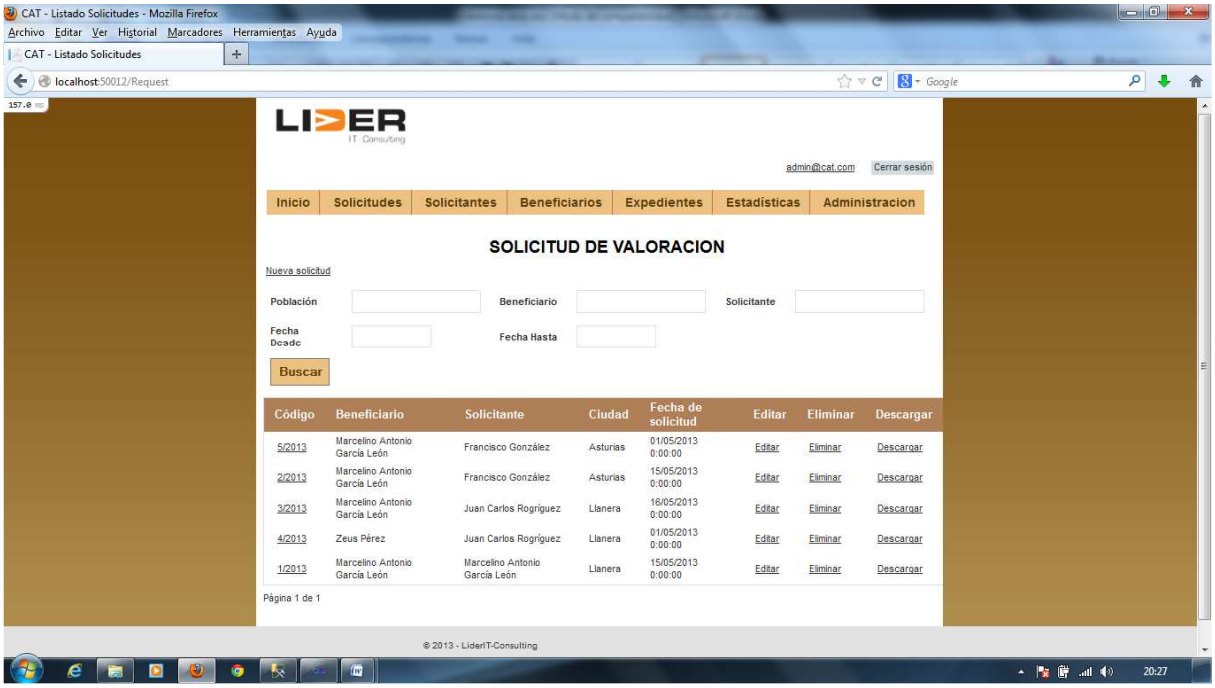


Figura 8. Listado de solicitudes

Como salida principal la aplicación mostrará en pantalla cualquier información relativa a los expedientes que se desee consultar, aunque también permitirá obtener una plantilla complementada con los datos consultados en caso de que los usuarios del sistema necesiten una copia en papel de dicha información.

3.6. Opciones de procesamiento

A la hora de almacenar toda la información en el sistema, se realizará una conexión una base de datos que será donde se almacenen los datos introducidos por los usuarios. Hay una conexión con una base de datos predeterminada por defecto, pero un usuario programador podrá cambiar dicha conexión a la base de datos permitiendo utilizar otra base de datos, que tiene que tener el mismo diseño aunque puede tener distintos datos almacenados. Esto se realiza mediante una cadena de conexión en el archivo de configuración de la aplicación según se explica en el manual técnico.

3.7. Funcionamiento del sistema

El sistema proporciona principalmente dos grupos de funciones al usuario: introducción de datos y consulta de datos.

La introducción de datos originalmente se realiza a través de una nueva solicitud, que permite la creación del expediente en caso de que no exista, con todos los datos necesarios. Pero también se puede añadir una intervención a un expediente existente.

La otra opción principal es la consulta de datos. A través de distintos filtros de búsqueda se puede acceder a la información deseada por el usuario.

El sistema también tiene otras funcionalidades adicionales:

- Gestión documental, que permite visualizar y almacenar los posibles documentos que estén asociados a las intervenciones.
- Generación de informes, que permite generar documentos a partir de plantillas y rellenarlas con la información deseada.
- Estadística.
- Administración de la aplicación, que permite administrar distintas opciones de los formularios y de la base de datos en caso de posibles cambios en el futuro.

3.8. Explicación de los términos relacionados con la impresión Gestión de Expedientes del CAT

En esta sección se explicara brevemente conceptos que tienen un significado muy general, pero particularizados para la aplicación de Gestión de Expedientes según el funcionamiento actual del Centro de Accesibilidad y Ayudas Técnicas.

Solicitud: la solicitud es la forma en la que se demanda un servicio del CAT, y que debe tener unos datos mínimos para que sea válida, como son datos relativos al solicitante, datos relativos al beneficiario y detalles de la solicitud (tipo de demanda, observaciones, etc.).

Solicitante: el solicitante es la persona física o jurídica que realiza la demanda del servicio. Puede ser el mismo beneficiario de la demanda o una persona distinta ya sea un familiar o un tercero en el ejercicio de sus funciones (terapeuta ocupacional, enfermero/a, médico, etc.)

Beneficiario: es, como su propio nombre indica, la persona que se beneficiará de la demanda solicitada.

Expediente: el expediente es el elemento que relaciona a un beneficiario con las solicitudes que le conciernen y las intervenciones realizadas a consecuencia de esas solicitudes. Cabe destacar que el expediente es único para cada beneficiario.

Intervención: es donde se plasman las actuaciones realizadas por el experto/s asignado con motivo de una solicitud. Además puede tener documentación asociada (fotografías, informes, certificados, etc.)

3.9. Fases del sistema

En líneas generales el sistema se compone de las siguientes fases:

- Introducción de datos.
- Consulta de datos.
- Gestión documental.
- Generación de informes.
- Estadística.
- Administración de la aplicación.

3.9.1. Introducción de datos

La introducción de datos en el sistema se realizará inicialmente a través de una nueva solicitud. Dicha solicitud puede tener como beneficiario a una persona que nunca haya realizado una solicitud con anterioridad o a uno que tenga una o más solicitudes previas. En caso de no tener solicitudes previas se generará un nuevo beneficiario en la base de datos con su respectivo expediente, relacionando la solicitud a dicho expediente. También se permitirá la creación de una nueva intervención relacionada con dicha solicitud. En caso de que sea un beneficiario con expediente existente en el sistema simplemente se relacionará la solicitud con el expediente, permitiendo de igual modo asignar una intervención.
También se guardarán los datos de los solicitantes, relacionando la solicitud al solicitante en caso de que exista o creando uno nuevo y guardando la información en la base de datos en caso de no existir.

Respecto a las intervenciones, si no se crean en el momento de la intervención, la aplicación permite crearla a posteriori, relacionándola con un expediente existente.

Además también se permite modificar la información a través de las pantallas de consulta, en caso de que cambie información relacionada con el beneficiario, solicitud, solicitante o intervención.

Asimismo se validará los datos que son necesarios para cada una de las entidades nombradas.

3.9.2. Consulta de datos

Para la consulta de datos se permite al usuario introducir los filtros que desea a la hora de buscar solicitantes, beneficiarios, solicitudes, expedientes e intervenciones, y cada uno de estos permite varias opciones de búsqueda como son por fecha, por nombre, código de expediente, etc.

Para ello se necesita normalizar la información introducida por el usuario y la información existente en la base de datos para así poder compararlas en el mismo formato y obtener el resultado deseado.

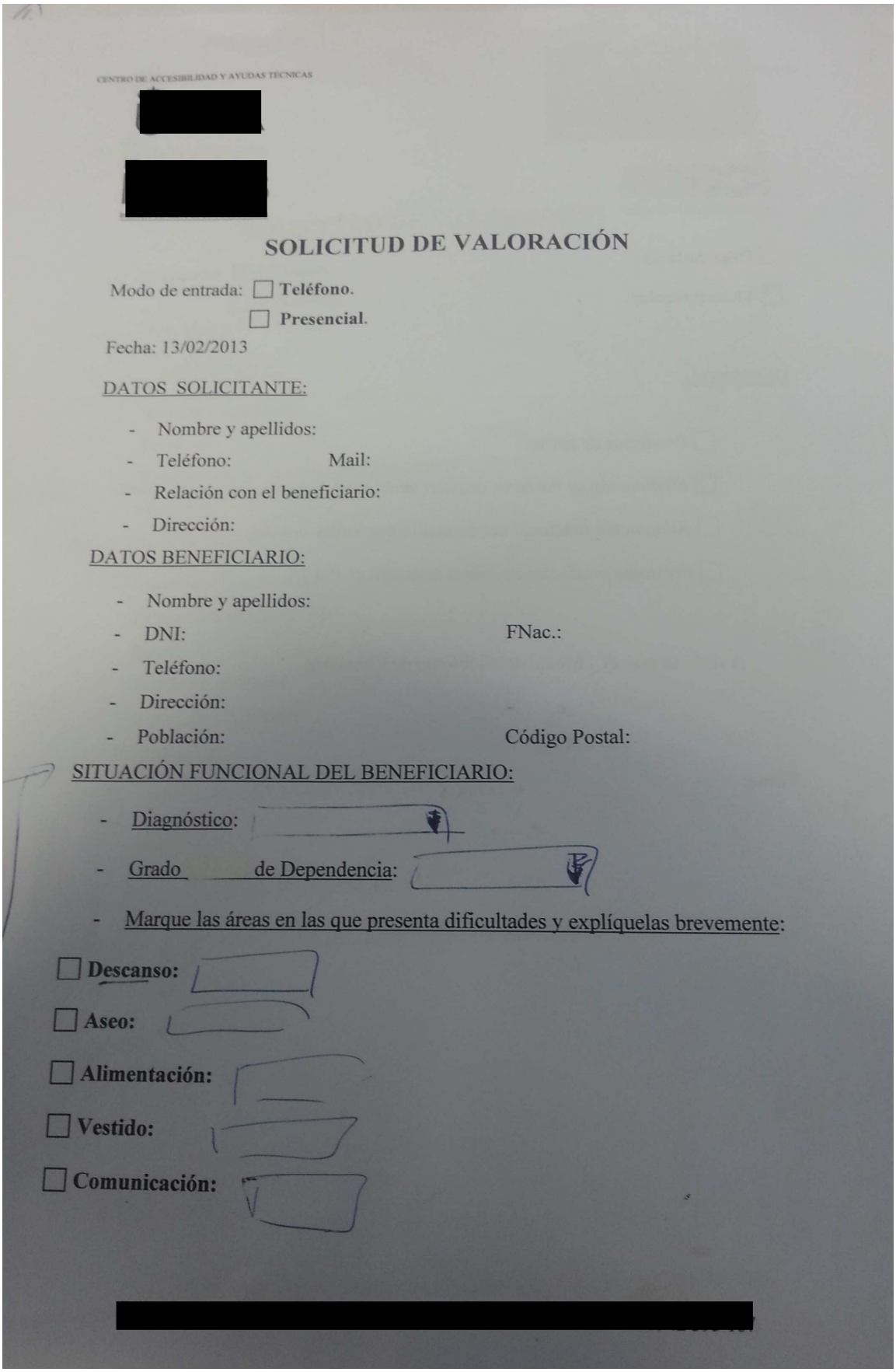
3.9.3. Gestión documental

El sistema también tiene una opción que permite una gestión de los documentos asociados a las intervenciones. Para ello se agregará el archivo directamente al crear la nueva intervención o en una posterior modificación de la intervención. Se permitirá al usuario administrador configurar el directorio donde guardar todos los documentos asociados. Asimismo existirá una tabla en la base de datos para relacionar todos los documentos con las intervenciones.

Además desde la pantalla de consulta de expedientes o intervenciones se podrá visualizar los documentos relacionados.

3.9.4. Generación de informes

Los usuarios del sistema requieren en muchos casos tener la información sobre las solicitudes en papel por cuestiones administrativas. Para ello se provee también un mecanismo de generación de informes que permite auto rellenar una plantilla con los datos de una solicitud o una intervención, permitiendo generar un documento que se podrá guardar en formato digital o imprimirlo para obtener su versión en papel.



CENTRO DE ACCESIBILIDAD Y AYUDAS TÉCNICAS

[redacted]

[redacted]

SOLICITUD DE VALORACIÓN

Modo de entrada: ☐ **Teléfono.**

☐ **Presencial.**

Fecha: 13/02/2013

DATOS SOLICITANTE:

- Nombre y apellidos:
- Teléfono: Mail:
- Relación con el beneficiario:
- Dirección:

DATOS BENEFICIARIO:

- Nombre y apellidos:
- DNI: FNac.:
- Teléfono:
- Dirección:
- Población: Código Postal:

SITUACIÓN FUNCIONAL DEL BENEFICIARIO:

- Diagnóstico:
- Grado de Dependencia:
- Marque las áreas en las que presenta dificultades y explíquelas brevemente:

☐ **Descanso:**

☐ **Aseo:**

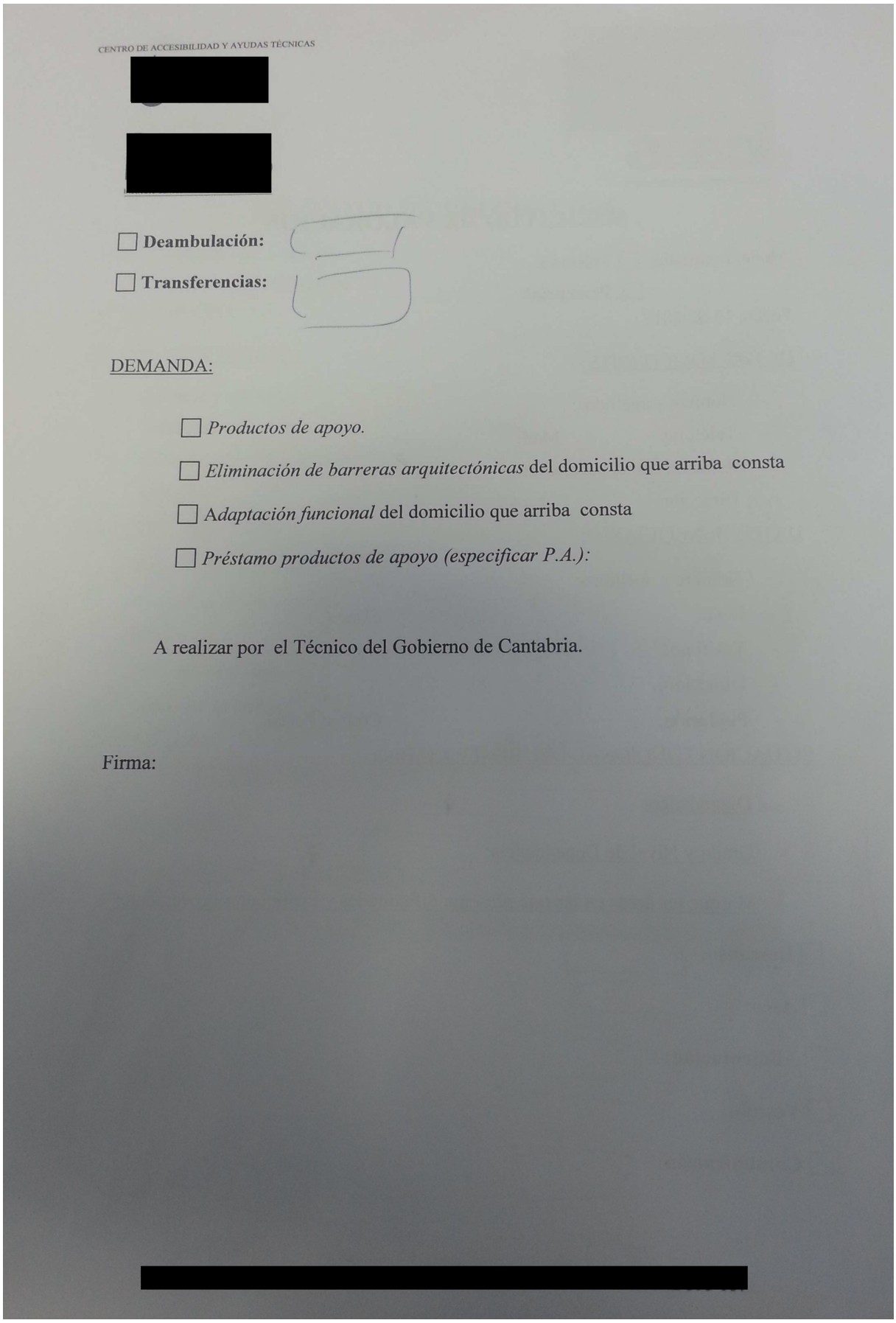
☐ **Alimentación:**

☐ **Vestido:**

☐ **Comunicación:**

[redacted]

Figura 9. Documento solicitud anverso



CENTRO DE ACCESIBILIDAD Y AYUDAS TÉCNICAS

[redacted]

[redacted]

☐ **Deambulación:** 5

☐ **Transferencias:** 5

DEMANDA:

☐ *Productos de apoyo.*

☐ *Eliminación de barreras arquitectónicas* del domicilio que arriba consta

☐ A*daptación funcional* del domicilio que arriba consta

☐ *Préstamo productos de apoyo (especificar P.A.):*

A realizar por el Técnico del Gobierno de Cantabria.

Firma:

[redacted]

Figura 10. Documento solicitud reverso

3.9.5. Estadística

Existe también un mecanismo que proporciona estadísticas en la aplicación. Inicialmente son datos estadísticos básicos sobre las solicitudes, expedientes, beneficiarios e intervenciones existentes, que mostrarán en pantalla los datos requeridos.

Existe la posibilidad de una extensión de la variedad de estadísticas a mostrar según las opiniones de los expertos de los Servicios Sociales después de la fase inicial de implantación de la aplicación.

3.9.6. Administración de la aplicación

Para los usuarios administradores existe la posibilidad de administrar los formularios de entrada de datos de la aplicación. Esto consiste en modificar las opciones de los formularios que tienen datos definidos, es decir, que no son de texto libre. Para ello se habilitará en la aplicación funciones de administración que permiten modificar, eliminar o crear cada uno de los elementos que deben estar definidos en estos formularios.

Mediante este mecanismo se permite mayor adaptación de la aplicación pensando en la posibilidad de que el Centro de Accesibilidad de Ayudas Técnicas pueda dar soporte a nuevos servicios y con mayor número de opciones de los que existen en la actualidad.

4. Descripción del producto

En este apartado se describe el entorno de implantación, junto a los requisitos hardware y software del sistema, así como las tecnologías utilizadas.

4.1. Entorno de implantación

El sistema se implantara, en una planta de la acería, en concreto el PC donde se instalará el sistema será un Pentium 4 a 2.66GH con 512MB de memoria.

4.2. Requisitos hardware

El hardware utilizado para la consecución práctica de este proyecto es:

- **PC**: Para este proyecto se cuenta con un PC de características comunes, procesador Pentium Dual-Core E5300 a una frecuencia de 2.60 GHz y con 4GB de memoria RAM.

El hardware utilizado para la implantación de este proyecto es:

- **PC**: Para este proyecto se cuenta con un servidor con procesador Intel Xeon CPU E5506 a una frecuencia de 2.14 GHz y con 4GB de memoria RAM.

4.3. Requisitos software

El software necesario para la consecución práctica de este proyecto es:

- **Sistema operativo Windows**: impuesto por otros programas que utilizaremos, que corren sobre él.

- **Microsoft Visual Studio .Net 2010 con Framework .Net MVC 4.0**: Entorno de desarrollo.

El software necesario para la implantación de este proyecto es:

- **Sistema operativo Windows Server 2003**.

- **Internet Explorer 5:** en versiones anteriores puede que algunas funciones den errores, principalmente al mostrar ciertos elementos.

4.4. Alternativas software

Existen dos partes software bien diferenciadas en la aplicación, que son el motor de base de datos y la aplicación web, aunque la segunda tiene mucho más peso en este proyecto. Se estudiaron varias posibilidades a la hora de elegir las tecnologías a utilizar, de las cuales las más significativas fueron:

Para la implementación de la aplicación web inicialmente se pensó en utilizar **Sharepoint 2010**, pero se desechó por cuestiones de compatibilidad de la máquina donde se implantaría debido a que el Sistema Operativo no tenía compatibilidad con dicha versión de Sharepoint y habría que desarrollarlo en Sharepoint 2007, lo que implicaba utilizar como motor de la base de dato SQL Server 2005, lo que limitaba el tamaño de la base de datos a 10 Gigabytes de almacenamiento, sin posibilidad de configurarlo como dinámico, y cabía la posibilidad de que esa cantidad de espacio no fuese suficiente.

Otra opción a valorar fue realizar una aplicación de escritorio, pero también de descarto porque este tipo de aplicaciones no es a lo que tiende el futuro de las aplicaciones informáticas, se podría decir que están quedando obsoletas. Además esta la desventaja de tener que realizar una instalación y aplicar cualquier cambio en la misma en cada uno de los ordenadores en los que se va a utilizar dicha aplicación.

Finalmente se opto por desarrollar una aplicación web en Modelo Vista-Controlador **Microsoft Visual Studio .Net 2010**, se opto por .Net básicamente por la facilidad de manejo, rendimiento, la utilización de librerías de enlace dinámico, y la conexión a la base de datos y la posibilidad de crear una aplicación en modelo vista-controlador enlazando el modelo de datos mediante la conexión a la base de datos.

El motor de base de datos a utilizar para el desarrollo es **SQL Server 2008 R2**, valorando principalmente las ventajas en cuanto a cuestiones de compatibilidad con Windows y Microsoft Visual Studio.

A la hora de implementar la aplicación se ha utilizado el framework **.NET**, utilizando **C#** como lenguaje para la implementación del controlador, **HTML** para la implementación de la vista y el modelo se obtiene a partir de una base de datos creada con **SQL.**

5. Documentos que acompañan al proyecto

Los documentos desarrollados son:

- Documento I: Memoria
- Documento II: Planificación y presupuesto
- Documento III: Análisis
- Documento IV: Diseño
- Documento V: Manual técnico
- Documento VI: Manual de usuario
- Anexo I: Código de la Aplicación
- Anexo II: Código de la Base de Datos

6. Información adicional

6.1. Metodología utilizada

Para desarrollar el análisis y el diseño del proyecto se ha seguido una metodología orientada a objetos, utilizando la notación UML para las representaciones gráficas.

6.1.1. Análisis orientado a objetos

El objetivo del análisis es especificar qué debe realizar la aplicación, pero no cómo se debe realizar.

El análisis orientado a objetos es un método de análisis que examina los requisitos desde la perspectiva de las clases y objetos que se encuentran en la definición del problema.

Las fases que se pueden seguir en el análisis son:

- Especificación de los requisitos
- Casos de uso
- Escenarios y sub-escenarios
- Modelado preliminar
- Prototipos

6.1.1.1. Especificación de requisitos

En esta fase se debe averiguar qué debe realizar el sistema. Para ello se ha realizado una reunión con los representantes del Centro de Accesibilidad y Ayudas Técnicas y se ha observado in situ la manera de trabajar de las personas que serán los futuros usuarios de la aplicación.

Los requisitos encontrados se expresan, y describen, de forma textual.

6.1.1.2. Casos de uso

Describen una conducta observable entre el usuario y el sistema. Permiten definir los límites del sistema y las relaciones entre el sistema y el entorno.

Cada caso de uso es una manera específica de utilizar el sistema.

Se representan mediante los diagramas de casos de uso y una descripción textual de cada uno de ellos.

- ✓ **Diagrama de casos de uso**

En un diagrama de casos de uso se pueden encontrar los siguientes elementos:

- Actores
- Sistema
- Casos de uso
- Relaciones
    - De comunicación
    - De extensión
    - De uso

Un actor representa un papel interpretado por una persona o cosa que interactúa con el sistema.

Una relación de comunicación se establece entre un actor y un caso de uso.

Una relación de extensión se establece entre casos de uso y significa que el caso de uso fuente extiende el comportamiento del caso de uso destino.

Una relación de uso se establece entre casos de uso y significa que el caso de uso fuente comprende también el comportamiento del caso de uso destino.

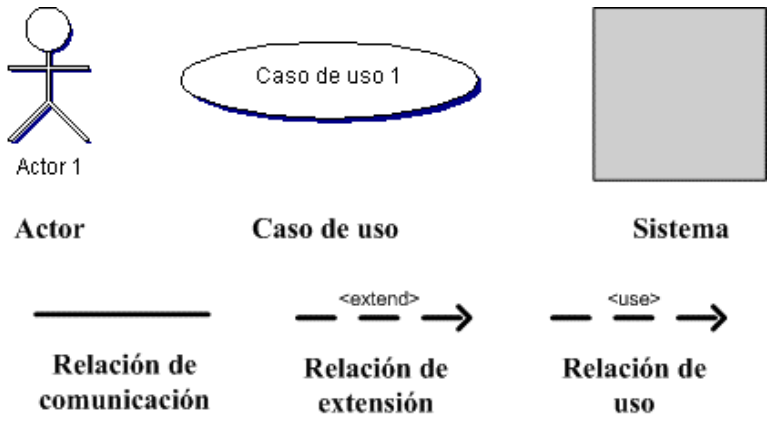


Figura 11. Elementos de un diagrama de casos de uso

6.1.1.3. Escenarios y sub-escenarios

Los escenarios constituyen una potente herramienta para el análisis orientado a objetos. Un caso de uso puede descomponerse en N escenarios, que a su vez pueden dividirse en sub-escenarios. Cada escenario identifica una ruta de acceso al caso de uso.

La descripción de los escenarios y sub-escenarios se realiza de forma textual.

6.1.1.4. Modelado preliminar

Se realiza un diagrama de clases preliminar, que posteriormente será refinado en el diseño.

Un diagrama de clase expresa de manera general la estructura estática de un sistema, en términos de clases y relaciones entre ellas.

Figura 12. Elementos de un diagrama de clases

Las relaciones entre clases también pueden tener multiplicidad, que muestra cuantos objetos de la clase considerada pueden ser relacionados con un objeto de la otra clase:

- 1 → uno y sólo uno
- * → uno o varios

La interpretación de la multiplicidad se puede ver en el siguiente ejemplo:

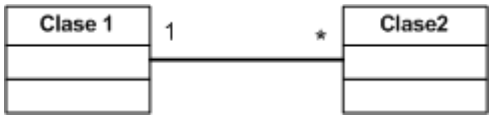


Figura 13. Ejemplo de multiplicidad

Un objeto de “Clase 1” se relaciona con cero o muchos de “Clase 2”, un objeto de “Clase 2” se relaciona sólo con 1 de “Clase 1”.

Las **asociaciones** representan relaciones estructurales entre clases. En ellas se puede expresar un sentido de navegación, indicado por la flecha, que indica en que sentido se debe entender la asociación. Si no se indica la flecha, la asociación es navegable en ambos sentidos.

La **herencia** representa una relación de clasificación entre un elemento más general y otro más específico. El elemento más general se representa junto a la punta de flecha.

La **agregación** representa una relación “parte de”. La clase que se encuentra en el extremo del rombo agrega objetos de la clase del otro extremo.

6.1.1.5. Prototipos

Consiste en la elaboración de una maqueta del sistema a desarrollar. En los prototipos se incluye una especificación de los interfaces de usuario, para comprobar que se satisfacen todos los requisitos.

6.1.2. Diseño orientado a objetos

El objetivo del diseño es determinar cómo se ha de realizar el sistema.

El diseño orientado a objetos es un método de diseño que abarca el proceso de descomposición orientado a objetos y una notación para describir modelos lógicos y físicos, dinámicos y estáticos, del sistema bajo diseño.

Las fases que se pueden seguir en el diseño son:

- Diseño del modelo de datos
- Modelado de clases y mecanismos de colaboración
- Modelado del comportamiento de las clases
- Construcción del modelo físico

6.1.2.1. Modelado de clases y mecanismos de colaboración

En esta fase se refina el modelo de clases creado en el análisis, añadiendo clases, relaciones,... y se realizan los diagramas que muestran las interacciones entre objetos.

Los objetos interactúan para realizar colectivamente los servicios ofrecidos por el sistema. Los diagramas de interacción muestras cómo se comunican los objetos.

Existen dos tipos de diagramas de interacción:

- Diagramas de secuencias: resaltan la ordenación temporal de las interacciones.
- Diagramas de colaboración: resaltan la organización estructural de los objetos que envían y reciben mensajes. Normalmente estos diagramas se obtienen a partir de los diagramas de secuencia.

En ocasiones no es necesario detallar los diagramas de colaboración.

- ✓ **Diagramas de secuencia**

Muestran los mensajes entre objetos. Cada objeto viene dado por una barra vertical en la que se representa su vida. El tiempo transcurre de arriba abajo

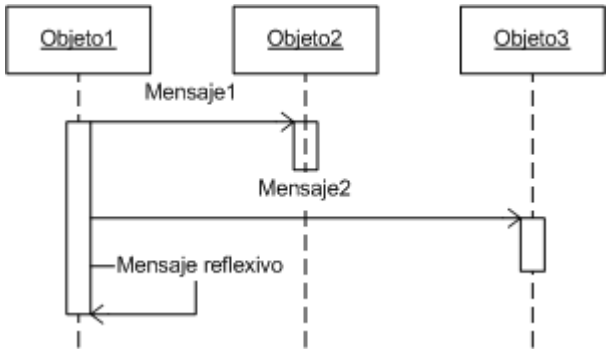


Figura 14. Diagrama de secuencia (I)

Las bandas rectangulares representan los periodos de actividad de los objetos.

Los mensajes pueden ir destinados hacia otro objeto (Mensaje1, Mensaje2) o hacia el propio objeto (Mensaje reflexivo). El retorno de cada mensaje se encuentra implícito en el mismo.

En un diagrama de secuencia también se pueden representar estructuras de control:

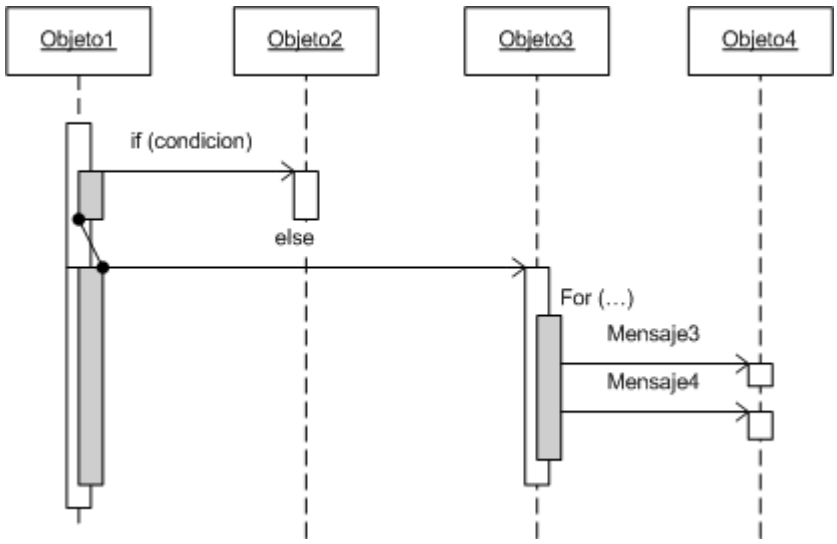


Figura 15. Diagrama de secuencia (II)

Si se cumple una condición “Objeto1” interacciona con “Objeto2”, en caso contrario interactúa con “Objeto3”. A su vez, “Objeto3” realizar un proceso iterativo en el que colabora con “Objeto4”.

6.1.2.2. Modelado del comportamiento de las clases

Para realizar este modelado se emplean los diagramas de estados. Estos diagramas visualizan autómatas de estados finitos, desde el punto de vista de los estados y las transiciones.

Los diagramas de estados se realizan para aquellos objetos que tienen un comportamiento significativo.

Figura 16. Elementos de un diagrama de estados

Los estados se caracterizan por la noción de duración y de estabilidad. Un objeto siempre está en algún estado en un determinado momento. Existen dos estados cuya representación es diferente al resto: estado inicial y estado final.

Cuando las condiciones dinámicas evolucionan, los objetos cambian de estado, siguiendo las transiciones. El paso de un estado a otro se efectúa cuando se desencadena una transición a causa de su evento asociado. Un evento se corresponde con la ocurrencia de una situación dada en el ámbito del problema.

6.1.2.3. Construcción del modelo físico

Para detallar la construcción del modelo físico se utilizan los diagramas de componentes. Estos diagramas describen los elementos físicos y sus relaciones en el entorno de la realización.

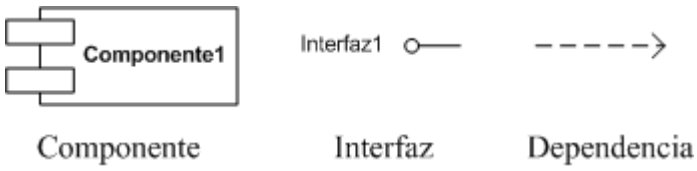


Figura 17. Elementos de un diagrama de componentes

Un componente representa todos los tipos de elementos físicos que entran en la fabricación de una aplicación. Pueden ser simples archivos, paquetes o bibliotecas. Un componente puede incluir, a su vez, a otros componentes.

Un interfaz representa un punto de comunicación entre dos componentes. Un componente puede tener un interfaz a través del cual otros componentes pueden utilizar sus servicios.

Las relaciones de dependencia se utilizan para indicar que un componente utiliza los servicios ofrecidos por otro componente.

6.2. Descripción de herramientas utilizadas

A continuación se describirán brevemente las herramientas utilizadas para la realización del proyecto.

6.2.1. Microsoft Visual Studio .Net 2003

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, y Visual Basic .NET, al igual que entornos de desarrollo web como ASP.NET. aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

Visual Studio 2010 es la versión más reciente de esta herramienta, acompañada por .NET Framework 4.0.

6.2.2. SQL Server 2008 R2

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional.

SQL Server 2008 R2 añade respecto a sus versiones anteriores ciertas características como son la gestión de datos maestros denominados Servicios de Datos Maestros y la gestión central de entidades y jerarquías de datos maestros. Además incorpora una Gestión Multi-Servidor, que es un consola centralizada que gestiona múltiples instancias de SQL Server 2008 y servicios como son bases de datos relacionales, servicios de informes, servicios de análisis y servicios de integración.

SQL Server 2008 R2 también incluye nuevos servicios compatibles con otras herramientas de Microsoft como son el PowerPivot para Excel y Sharepoint, Master Data Services y Reporting Services para Sharepoint, función de nivel de Datos para Visual Studio que habilita el empaquetado de bases de datos jerarquizadas como parte de una aplicación, y una utilidad de SQL Server llamada UC (Utility Control Point) que se utilizada para gestionar múltiples instancias de SQL Servers.

6.3. Lenguajes de programación usados

6.3.1. Lenguaje C#

El C# es un lenguaje de programación, diseñado en 2002, por Anders Hejlsberg.

C# es un lenguaje de programación de propósito múltiple, fuertemente tipado, declarativo, funcional, genérico y orientado a objetos que encaja con una amplia variedad de necesidades surgidas en el ámbito del desarrollo.

C# fue desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, y aprobado como un estándar por ECMA e ISO. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Además de los principios que tienen los lenguajes orientados a objetos, C# facilita el desarrollo de software gracias a ciertas construcciones del lenguaje, como son:

- Firmas de métodos encapsuladas que permiten las notificaciones de eventos en modo seguro.
- Propiedad, que sirven como ancestros de variables de métodos privados.
- Atributos, que proporcionan metadatos declarativos sobre tipos en tiempo de ejecución.
- Comentarios documentales entrelineas en XML.
- Peticiones de lenguaje integradas (LINQ), que proporcionan opciones de petición a través de una gran variedad de recursos de almacenamiento de datos.

6.3.2. HTML

La primera descripción de HTML fue un documento publicado por primera vez en Internet por Tim Berners-Lee en 1991, en el que se especificaba el diseño inicial de HTML.

Las siglas de HTML corresponden con HyperText Markup Language, y es por lo tanto un lenguaje de marcado, que en la actualidad es el principal lenguaje para la elaboración de páginas web y toda la información que se pueda mostrar en un navegador web.

HTML se escribe de manera en la que los elementos HTML son introducidos dentro de etiquetas, que definen el tipo de elemento y que se introducen entre paréntesis angulares. Dentro de estas etiquetas los diseñadores web pueden añadir texto, etiquetas, comentarios y otro contenido de texto para definir o caracterizar los elementos.

Los navegadores web se encargan de leer el código HTML y traducirlo o interpretarlo de modo que lo transforman en páginas web visibles y/o audibles. Para realizar esta tarea, el

navegador no muestra las etiquetas, pero las utilizar para interpretar el contenido que debe tener la página.

HTML permite introducir imágenes y otros objetos dentro de la página y ser usados como elementos interactivos. Además proporciona etiquetado estructural, que permite definir cabeceras, títulos, párrafos, etc. A su vez, HTML permite introducir scripts escritos en lenguajes como JavaScript que permiten modificar el comportamiento de las páginas web, añadiendo en muchos casos nuevas opciones que no se podrían implementar con HTML.

Los navegadores pueden interpretar también Hojas de Estilo o CSS, que definen la apariencia de los elementos que forman el código HTML y permiten realizar una presentación correcta del documento HTML.

6.3.3. SQL

SQL es un lenguaje que fue desarrollado por Donald D. Chamberlin and Raymond F. Boyce para IBM a principio de los años 1970.

Las siglas de SQL corresponden a Structured Query Language, y es un lenguaje de propósito especial diseñado para gestionar el manejo de datos en los sistemas de gestión de bases de datos relacionales. Está basado en el cálculo de tuplas relacionales y en el algebra relacional, y consta de un lenguaje de definición de datos y un lenguaje de manipulación de datos, que proporciona inserción de datos, peticiones de datos, actualización y borrado, esquemas de creación y modificación y control de acceso a dichos datos.

SQL fue aprobado como estándar por ANSI e ISO. Desde entonces ha sido optimizado en varias ocasiones, añadiendo nuevas características.

6.3.4. XML

XML deriva de un lenguaje inventado por IBM en los años setenta, que surgió por la necesidad que tenía la empresa de almacenar grandes cantidades de información.

Las siglas corresponden a Extensible Markup Language is es un lenguaje de marcación que define normas para la codificación de documentos que permiten crear un formato que sea legible tanto por los humanos como por los computadores.

Las ventajas de este lenguaje son la simplicidad, generalidad y usabilidad que proporciona dentro del mundo de internet, siendo un formato de datos textual muy usado por otros lenguajes. Además, aunque XML fue inicialmente diseñado para formatear documentos, en la actualidad es bastante extendido su uso para representar también estructuras de datos.

6.4. Tecnologías usadas

Para mejorar las funcionalidades de la aplicación, además de los lenguajes utilizados, se han utilizado otras tecnologías muy extendidas en la actualidad y compatibles con los lenguajes y herramientas anteriormente explicados.

6.4.1. JavaScript

JavaScript es un lenguaje de programación de script basado en prototipos, dinamico y débilmente tipado, que fue originalmente implementado como parte de los navegadores, permitiendo que a través de scripts del lado de cliente los usuarios puedan interactuar con la aplicación controlar el navegador, realizar comunicaciones asíncronas y cambiar el contenido a mostrar del documento.

La sintaxis de JavaScript es muy similar a C, aunque también adopta aspectos de Java, aunque no están relacionados ya que tienen semánticas y propósitos diferentes.

Actualmente el código JavaScript está muy extendido y cualquier navegador interpreta código JavaScript existente en páginas web. Para interactuar con las páginas, se provee al lenguaje JavaScript del DOM (Modelo de Objetos del Documento).

6.4.2. JSON

JSON es el acrónimo de JavaScript Objetc Notation, y es un estándar especificado Douglas Crockford y diseñado para el intercambio de datos y legible por humanos que deriva de JavaScript, aunque se utiliza con otros leguajes, y diseñado para representar estructuras de datos y objetos.

El formato JSON es utilizado para transmitir estructuras de datos, principalmente entre servidores y aplicaciones web, y que se utiliza como alternativa a XML, debido a la sencillez al escribir un analizador sintáctico para JSON, aunque también es posible y frecuente usar ambas tecnologías juntas en la misma aplicación.

6.4.3. AJAX

Ajax son las siglas de Asynchronous JavaScript and XML, aunque el uso de XML no es un requisito indispensable (de hecho JSON es utiliza muy frecuentemente como alternativa) y las peticiones no son siempre asíncronas. Ajax no es una tecnología en concreto, sino que es un grupo de tecnologías o un conjunto de técnicas de desarrollo web, usadas en el lado del cliente para crear aplicaciones web dinamicas y asíncronas. Usando Ajax las aplicaciones pueden enviar y recuperar datos del servidor de manera asíncrona, sin interferir con la presentación y el comportamiento de la página actual.

El DOM es accesible con JavaScript, lo que permite modificar lo que se muestra en la página dinámicamente, permitiendo al usuario interactuar con la información mostrada. Además mediante JavaScript y el objeto XMLHttpRequest se puede acceder a métodos para intercambio de archivos asíncronamente entre navegador y servidor evitando cargas completas de páginas, simplemente se carga la parte deseada de la misma.

6.4.4. jQuery

jQuery es una librería de JavaScript diseñada para simplificar el scripting de lado del cliente con HTML. Fue creada en 2006 por John Resig y actualmente está muy extendido su uso dentro de los sitios web, de hecho es la librería de JavaScript más popular.

jQuery es un recurso software libre y abierto cuya sintaxis está diseñada para facilitar la navegación a través del documento cargado, seleccionar elementos del DOM, manejar eventos, crear animaciones y desarrollar aplicaciones con Ajax.

7. Posibles ampliaciones

Entre las posibles ampliaciones del proyecto podemos citar las siguientes:

- Ampliar la aplicación, por módulos, para que además de gestionar los expedientes de las demandas pueda gestionar los elementos pertenecientes a ella. Por ejemplo, en caso de demandas de préstamo productos de apoyo, realizar un módulo que permita gestionar los productos existentes, la cantidad, a qué beneficiario se le ha prestado y cuando, etc.

- La aplicación está diseñada para ser utilizada por los trabajadores del CAT, pero podría extenderse la interfaz, y añadir otro rol a la capa de usuarios, permitiendo realizar solicitudes vía web, directamente a la aplicación.

8. Conclusiones

En este proyecto he tenido la fortuna de realizar todos los ciclos de vida de un producto software en la vida real, desde la reunión con el cliente hasta la implantación. Por lo tanto ha sido una experiencia muy útil de la cuál he podido sacar conclusiones que van a ser de ayuda si las tengo en cuenta en futuros proyectos:

Primeramente y una de las cosas más difíciles e importante es clarificar las necesidades del cliente. A no ser que el cliente pertenezca al sector de las tecnologías de la información, es posible que tenga unos conocimientos informáticos bastante limitado, lo que impone una diferencia bastante amplia entre los puntos de vista de ambas partes y el entendimiento en cuanto a términos y expresiones. En este caso me ha sido muy útil ver in situ la manera de trabajar del usuario final, para poder obtener una idea inicial de las necesidades del cliente.
Esto permite guiar al cliente con el objetivo de plasmar todos sus requisitos y detallarlos lo más minuciosamente posible para realizar un diseño adecuado de la aplicación.

Además conviene cerrar por completo la toma de requisitos y obtener la conformidad de ambas partes, ya que si no es probable que haya cambios continuamente y una vez comenzado el desarrollo de la aplicación puede obstaculizar y ralentizar mucho la implementación del proyecto si hay cambios frecuentes.

En cuanto al análisis y al diseño pude comprobar que la experiencia es un factor muy importante. Una persona que no está acostumbrada a realizar estas tareas se atasca con facilidad y le cuesta avanzar y en cambio cuando se trabaja o se recibe la ayuda de una persona con experiencia y que ha realizado estas fases en una gran cantidad de proyectos se puede comprobar que hay mucha más fluidez y es una tarea mucho más mecánica en la que se aplica muchas veces la analogía con proyectos anteriores de manera automática.

En lo que se refiere al desarrollo, se puede sacar en claro que para crear una aplicación web con unas prestaciones óptimas es necesario conocer varios lenguajes y tecnologías, lo que es un problema para alguien que no ha tenido experiencia en este ámbito. El aprendizaje de todas estas técnicas y tecnologías supone un aumento considerable del tiempo de desarrollo, pero también es verdad que una vez se aprende, las siguientes situaciones que requieran su uso, el tiempo se reduce muchísimo, solo hay que buscar como se resuelve el problema en concreto y no como funciona dicha tecnología.

9. Bibliografía

9.1. Documentos y libros

- Documentación propia del Centro de Accesibilidad y Ayudas Técnicas
- Beginning ASP.NET 2.0 with C# (Wrox Beginning Guides)

9.2. Sitios web:

- Web oficial de Microsoft Development Network:
  http://msdn.microsoft.com

- Web oficial de .Net:
  http://www.microsoft.com/net

- Web oficial de Asp.Net MVC:
  http://www.asp.net/mvc

- Foro sobre programación:
  http://stackoverflow.com/

- Sitio Web para desarrolladores:
  http://www.w3schools.com/

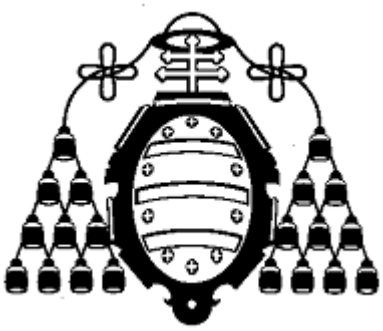
- Enciclopedia online:
  http://www.wikipedia.com

- Web de jQuery
  http://jquery.com/

- Web del Gobierno de España sobre Metrica v.3
  http://administracionelectronica.gob.es/?_nfpb=true&_pageLabel=P800292251293651550991&langPae=es&detalleLista=PAE_000000432

- Web de UML
  http://www.uml.org/

- Guía sobre modelado ágil
  http://www.agilemodeling.com/



UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

HERRAMIENTA DE GESTIÓN DE EXPEDIENTES PARA EL CAT

DOCUMENTO Nº II

PLANIFICACIÓN

ZEUS PÉREZ GARCÍA
JUNIO 2012

ÁREA DE TELEMÁTICA
TUTOR: XICU XABIEL GARCÍA PAÑEDA

ÍNDICE DE LA PLANIFICACIÓN Y PRESUPUESTO

ÍNDICE DE LA PLANIFICACIÓN

ÍNDICE DE FÍGURAS DE LA PLANIFICACIÓN

PLANIFICACIÓN Y PRESUPUESTO

10. Planificación

10.1. Descomposición de tareas

Las tareas en las cuales se descompone el proyecto son:

1. Especificación de requisitos
2. Análisis
3. Diseño
4. Adaptación al entorno de desarrollo
5. Implementación del sistema
6. Depuración y pruebas
7. Elaboración de la documentación

10.2. Descripción de las tareas

10.2.1. Especificación de requisitos

En esta fase se realizan reuniones con el cliente, visualización in situ del trabajo que realiza el usuario final y una consultoría para poder especificar los requerimientos que debe tener le sistema final, realizar una puesta en común de los mismos y llegar a un acuerdo sobre las características que debe tener el producto final.

10.2.2. Análisis

En esta tarea se detalla la especificación de requisitos anterior, así como su representación en casos de uso y escenarios. Además se detallan los sistemas con los que se va a comunicar la aplicación y el guión del interfaz de la misma.

10.2.3. Diseño

Se realizan las fases del diseño orientado a objetos, siguiendo la notación UML, y el diseño del plan de pruebas de la aplicación. Además se diseñara el modelo de datos (con la base de datos correspondiente).

10.2.4. Adaptación al entorno de desarrollo

En esta tarea se efectúa la comprensión de las técnicas empleadas en el desarrollo de aplicaciones que trabajan con imágenes, así como la adaptación a las mismas.

10.2.5. Implementación del sistema

En esta tarea se construye el sistema con las especificaciones presentes en el análisis y el diseño.

10.2.6. Depuración y pruebas

Se realiza la verificación del correcto funcionamiento de la aplicación y la detección de posibles fallos de implementación.

10.2.7. Elaboración de la documentación

Se genera la documentación requerida en el Reglamento de Trabajos Fin de Máster de la Escuela Politécnica de Ingeniería de Gijón.

10.3. Planificación de tareas

ID	Nombre de la tarea	Febrero 2013				Marzo 2013				Abril 2013				Mayo 2013				Junio 2013			
		S 1	S 2	S 3	S 4	S 1	S 2	S 3	S 4	S 1	S 2	S 3	S 4	S 1	S 2	S 3	S 4	S 1	S 2	S 3	S 4
1	Especificación de requisitos																				
2	Análisis																				
3	Diseño																				
4	Adaptación al entorno de desarrollo																				
5	Implementación del sistema																				
6	Depuración y pruebas																				
7	Elaboración de la documentación																				

Tabla 01. Calendario de desarrollo del proyecto.

11. Presupuesto

11.1. Presupuesto de recursos humanos

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN					
PRESUPUESTO					
Nº	NUMERO DE UNIDADES (horas)	DESCRIPCIÓN	PRECIO DE LAS UNIDADES (€/hora)	IMPORTES PARCIALES Euros	IMPORTES TOTALES Euros
1	30	Especificación de requisitos	20	600	600
2	25	Adaptación al entorno de desarrollo.	20	500	1.100
3	90	Análisis	40	3.600	4.700
4	120	Diseño	40	4.800	9.500
5	160	Implementación del sistema	30	4.800	14.300
6	30	Depuración y pruebas	30	900	15.200
7	80	Elaboración de la documentación	20	800	**16.000**

Tabla 02. Presupuesto de recursos humanos

11.2. Presupuesto hardware

Para realizar el presupuesto de hardware, se estima una proporción del coste del software para asignarle a cada uno de los proyectos en los que se utiliza, ya que no es necesario realizar una partida solo para este proyecto al poder utilizarse en anteriores y posteriores proyectos.

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN					
PRESUPUESTO					
Nº	NUMERO DE UNIDADES	DESCRIPCIÓN	PRECIO PROPORCIONAL DE LAS UNIDADES (€/unidad)	IMPORTES	
				PARCIALES	TOTALES
				Euros	Euros
1	1	PC de características comunes, procesador Pentium 4 a 3GHz, con	100	10	**100**

Tabla 03. Presupuesto de recursos hardware

11.3. Presupuesto software

Para realizar el presupuesto de software, se estima una proporción del coste del software para asignarle a cada uno de los proyectos en los que se utiliza, ya que no es necesario realizar una partida solo para este proyecto al poder utilizarse en anteriores y posteriores proyectos.

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN					
PRESUPUESTO					
Nº	NUMERO DE UNIDADES	DESIGNACION DE LAS OBRAS	PRECIO PROPORCIONAL DE LAS UNIDADES (€/unidad)	IMPORTES	
				PARCIALES	TOTALES
				Euros	Euros
1	1	Microsoft Visual Studio .Net 2010	20	20	20
2	1	SQL Server 2008 R2	15	15	**35**

Tabla 04. Presupuesto de recursos software

11.4. Presupuesto final

Para realizar el cálculo total del producto se han sumando distintos costes:

- Presupuesto de recursos humanos
- Presupuesto de recursos hardware
- Presupuesto de recursos software
- Un beneficio industrial del 6% según política de empresa
- Unos costes generales del 15% que incluyen gastos proporcionales de luz, agua, seguro, conexión a Internet, etc.
- El I.V.A. actual del 21%

Por lo tanto el presupuesto de ejecución se resume en las siguientes tablas:

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN				
PRESUPUESTO				
Nº	NUMERO DE UNIDADES	DESIGNACION DE LAS OBRAS	IMPORTES PARCIALES Euros	TOTALES Euros
1	1	Presupuesto recursos humanos	16.000	16.000
2	1	Presupuesto recursos hardware	35	16.035
3	1	Presupuesto recursos software	100	**16.135**

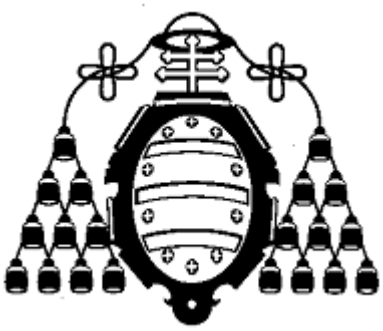
Tabla 05. Presupuesto final

Presupuesto de ejecución material (€)	=	16.135,00
Beneficio industrial (6 %) (€)	=	968,10
Costes generales (15 %) (€)	=	2.420,25
Suma de gastos y beneficios (€)	=	19.523,35
I.V.A. (21 %) (€)	=	4.099,90
Presupuesto de ejecución por contrata (€)	=	**23.623,25**

Tabla 06. Presupuesto de ejecución por contrata

Asciende el presupuesto de ejecución por contrata a la expresada cantidad de veintitrés mil seiscientos veintitrés con veinticinco céntimos de euro (23.623,25 €).

Gijón, 18 de Febrero de 2013



UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

HERRAMIENTA DE GESTIÓN DE EXPEDIENTES PARA EL CAT

DOCUMENTO Nº III

ANÁLISIS

ZEUS PÉREZ GARCÍA
JUNIO 2012

ÁREA DE TELEMÁTICA
TUTOR: XICU XABIEL GARCÍA PAÑEDA

ÍNDICE DEL ANÁLISIS

ÍNDICE DEL ANÁLISIS

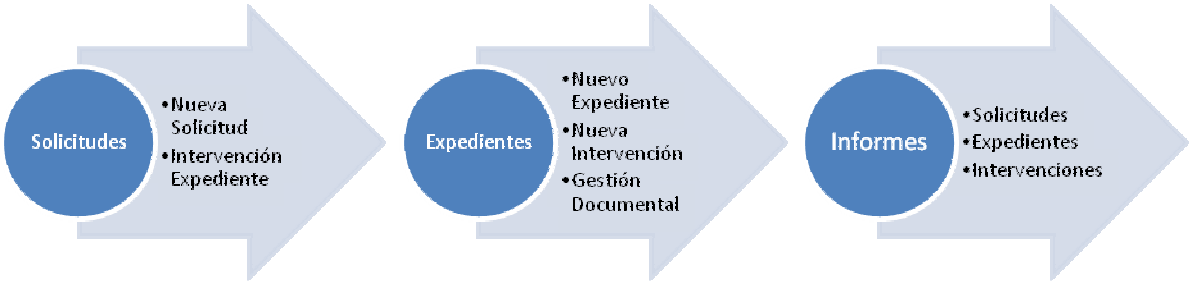
ÍNDICE DE FÍGURAS DEL ANÁLISIS

ANÁLISIS

1. Análisis de requisitos

1.1. Introducción

El presente documento recoge los requerimientos para la aplicación de Gestión de Expedientes que tramita el Instituto Cántabro de Servicios Sociales (ICASS).



El proceso de un expediente se inicia siempre con el registro de una Solicitud de un "Solicitante" para un "Beneficiario". Esta solicitud genera un único expediente que permanece vivo en el sistema con las diferentes intervenciones que registran los agentes del ICASS.

1.2. Módulo I: Solicitudes

1.2.1. Resumen

Este módulo tiene por objeto registrar y almacenar en la base de datos las diferentes peticiones que puede recibir el Instituto Cántabro de Servicios Sociales a través de diferentes medios: teléfono, correo electrónico, fax…
Las solicitudes pueden generar a dos tipos de elementos:

- Expedientes.
- Intervenciones.

1.2.2. Requisitos

1. Registrar peticiones de solicitantes que generan expedientes para beneficiarios de servicios.
2. Registrar peticiones de solicitantes que generan intervenciones en los expedientes existentes de beneficiarios de servicios.

1.3. Módulo II: Expedientes

1.3.1. Resumen

Este módulo tiene por objeto registrar y almacenar en la base de datos los diferentes expedientes generados a partir de una solicitud.

1.3.2. Requisitos

1. Registrar expedientes para beneficiarios de servicios.
2. Registrar intervenciones en los expedientes existentes de beneficiarios de servicios.

1.4. Módulo III: Informes

1.4.1. Resumen

Este módulo tiene por objeto explotar la información almacenada en la Gestión de Expedientes.

1.4.2. Requisitos

1. Generar informes sobre Solicitudes.
2. Generar informes sobre Expedientes.
3. Generar informes sobre Intervenciones.

2. APLICACIÓN WEB

El nuevo Sistema de Gestión de Expedientes será una aplicación web desarrollada bajo .NET que tendrá aspectos claramente diferenciados:

- Registro de Solicitudes, Expedientes e Intervenciones
- Generación de informes y estadísticas de los datos almacenados en la aplicación.

2.1. Entidades Principales

Solicitudes: Listado de las solicitudes presentadas.
Solicitantes: Listado de personas que presentan las solicitudes.
Beneficiarios: Listado de personas beneficiarias de las solicitudes.
Expedientes: Listado de los expedientes, cada uno asociado a un único beneficiario.
Intervenciones: Listado de las intervenciones realizadas.

2.2. Entidades Secundarias

Demandas: Listado con la definición con el tipo de demanda de la solicitud.
Entidades: Listado con la definición con los distintos tipos de entidades.
Centros: Listado con la definición con el tipo de centro de cada entidad.
Estados de Intervención: Listado con la definición con los estados de una intervención.

Grados de Dependencia: Listado con la definición con los distintos grados/niveles de dependencia.
Limitaciones Funcionales: Listado con la definición de las posibles limitaciones funcionales de un beneficiario.
Puestos: Listado con la definición de los distintos puestos que pueden tener los técnicos.
Sexo: Listado con la definición del sexo de los beneficiarios.
Técnicos: Listado con los técnicos existentes en el sistema.
Discapacidades: Listado con la definición de los tipos de discapacidad del beneficiario.
Entradas: Listado con la definición de las distintas vías por las que se puede hacer una solicitud.
Tipo de Solicitantes: Listado con la definición con el tipo de los solicitantes.
Tipo de Beneficiarios: Listado con la definición con el tipo de los beneficiarios.

2.3. Roles de Usuarios

a. Usuario Básico
b. Usuario Administrador

3. REQUERIMIENTOS FUNCIONALES

3.1. Módulo I: Solicitudes

3.1.1. Resumen

La gestión de solicitudes tiene por objeto registrar las distintas peticiones que puede recibir y tramitar el Instituto Cántabro de Servicios Sociales.
Las solicitudes pueden desencadenar dos actividades:

- Expedientes.
- Intervenciones.

3.1.2. Requisitos Funcionales

Las **solicitudes** deben cumplir los requerimientos descritos en los próximos puntos.

RF 1: Las solicitudes se componen de varios elementos:

- Modo de entrada y Fecha de Solicitud.
- Solicitante.
- Beneficiario.
- Situación Funcional del Beneficiario.
- Demanda.

RF 2: Los beneficiarios pueden ser de dos tipos: “Persona Física” y “Entidad”.

RF 3: Un beneficiario de tipo “Entidad” puede tener diferentes centros que pueden realizar solicitudes. Asimismo un beneficiario de tipo “Entidad” puede tener varias direcciones.

RF 4: Cada beneficiario tendrá un único expediente asociado con una solicitud.

RF 5: Cada solicitud generará un documento de Word con la información rellenada por el operario en la aplicación web.

RF 6: El modo de entrada de una solicitud podrá tomar un conjunto de valores que serán gestionados por un usuario Administrador. Algunos de sus valores serán: Teléfono, Presencial, Fax, Corre Electrónico…

RF 7: El formulario de Solicitud para una Personas Jurídica no tiene el apartado “Situación Funcional del Beneficiario”.

RF 8: Los datos del apartado “Solicitante” serán los siguientes:

Persona Física

- Nombre y apellidos
- Teléfono
- Mail
- Relación con el beneficiario
- Dirección

RF 9: Los datos del apartado “Beneficiario” serán los siguientes:

Persona Física

- Nombre y apellidos
- DNI
- Fecha Nacimiento
- Teléfono
- Dirección
- Población
- Código Postal
- Email:

Persona Jurídica

- Denominación Social
- Denominación Comercial
- CIF
- Teléfono
- Dirección
- Población
- Código Postal

RF 10: Los datos del apartado “Situación Funcional del Beneficiario” (Este apartado será solamente para el beneficiario de tipo “Persona Física”) serán los siguientes:

- Diagnóstico
- Grado y Nivel de Dependencia
- Limitaciones funcionales:
    - Descanso
    - Aseo
    - Alimentación

- Vestido
- Comunicación
- Deambulación
- Transferencia

RF 11: Los datos del apartado “Demanda” serán los siguientes:

1. Productos de Apoyo, desplegable en las siguientes opciones:
    a. Descanso
    b. Aseo
    c. Alimentación
    d. Vestido
    e. Comunicación
    f. Deambulación
    g. Transferencias
2. Préstamo de Productos de Apoyo, desplegable en los mismos campos que el anterior, y relacionado con el Módulo “Gestión del Préstamo de Productos de Apoyo”, aún sin determinar el modo.
3. Adaptación Funcional de la Vivienda, desplegable en:
    a. General
    b. Servicio higiénico-sanitario
    c. Cocina
    d. Dormitorio
    e. Acceso a vivienda
    f. Otras
4. Eliminación de Barreras Arquitectónicas
5. Certificado de Accesibilidad, desplegable en:
    a. Agencia de Colocación
    b. Centro de Formación
    c. Autoescuela
    d. Otras
6. Certificado para IRPF
7. Denuncias
8. Información Plan General de Ordenación Urbana
9. Formación
10. Asesoramiento Jurídico

RF 12: Cada solicitud tendrá un código que la identifica y la diferencia de cualquier otra solicitud. Su formato será: [Num/Año]. “Num” será un valor entero correlativo que comienza en 1. “Año” será el año en el que se genera la solicitud.

3.2. Módulo II: Expedientes

3.2.1. Resumen

La gestión de expedientes tiene por objeto registrar los distintos expedientes que tramita el Instituto Cántabro de Servicios Sociales.
Los expedientes se componen de:

- Detalle Expediente.
- Intervenciones.
- Documentos.

3.2.2. Requisitos Funcionales

Los **expedientes** deben cumplir los requerimientos descritos en los próximos puntos.

RF13: Cada expediente puede tener asociadas varias solicitudes y varias intervenciones.

RF 13: Cada expediente heredará de la solicitud inicial los campos: Año, Tipo de Entrada, Solicitante y Beneficiario.

RF 14: Cada expediente tendrá un código que lo identifica y lo diferencia de cualquier otro expediente. Su formato será: [Num/Año]. “Num” será un valor entero correlativo que comienza en 1. “Año” será el año en el que se genera el expediente.

RF 15: La ficha de detalle del expediente solicita al operador información sobre: “Técnico”, “Puesto”, “Proyecto”, “Asunto”, “Entidad”, “Centros”.

RF 16: Cada expediente tiene un “Estado” definido que toma los valores: “Pendiente” y “Terminado”.

RF 17: Los campos “Proyecto” y “Asunto” se unifican en un único campo denominado “Demanda”.

RF 18: Los expedientes se componen de un conjunto de intervenciones y un conjunto de documentos anexos al mismo.

RF 19: Cada intervención recoge los trabajos realizados por los técnicos del Instituto Cántabro de Servicios Sociales sobre cada expediente.

RF 20: Cada intervención recoge los campos: “Acción a Realizar”, “Observaciones”, “Técnico”, “Demanda”, “Actuación”, “Fecha de Intervención”, “Fecha de Apertura”, “Fecha de Cierre”, “Fecha de la Visita” y “Estado” (Abierta o Cerrada).

RF 21: Cada intervención puede tener asociada un conjunto de documentos anexos.

4. Casos de uso

Diagrama de Casos de Uso

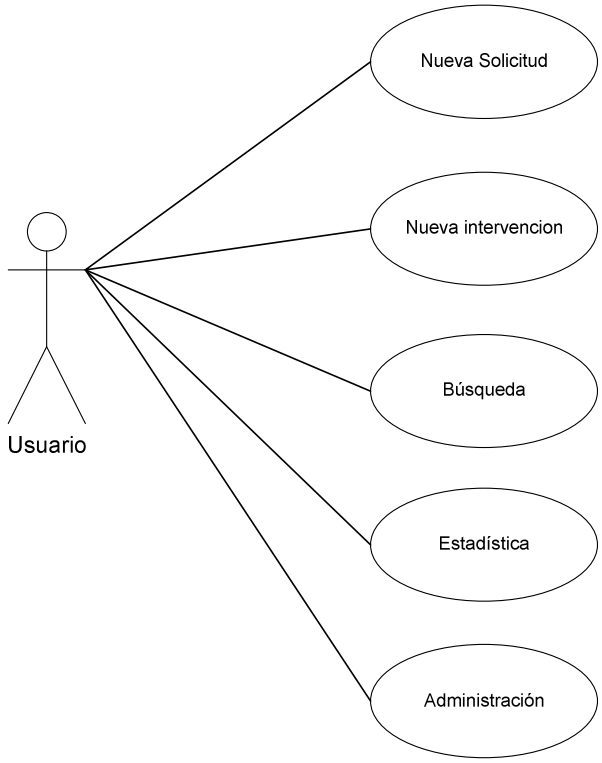


Figura 1. Casos de uso

2.1 Casos de Uso Nueva Solicitud

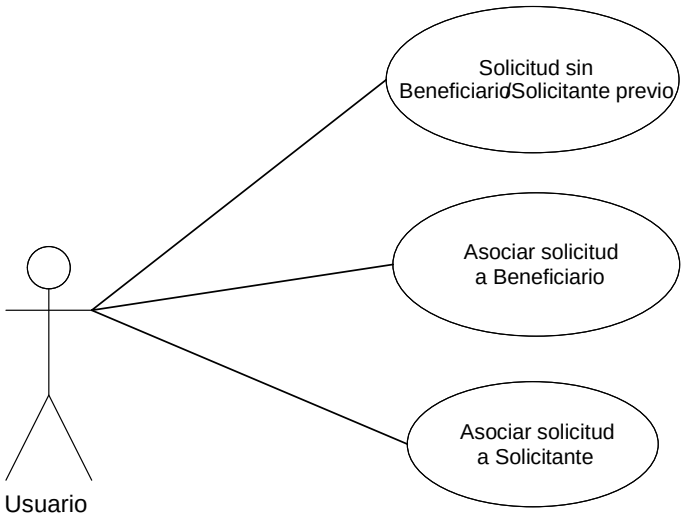


Figura 1. Caso de uso nueva solicitud

1. Cuando el usuario desea introducir una nueva solicitud accede a la pantalla de introducción de datos de la solicitud.
2. En caso de que ni el solicitante ni el beneficiario hayan realizado solicitudes previas se tendrán que rellenar todos los datos necesarios sobre solicitante beneficiario y solicitud, y se creara una nueva solicitud, un nuevo beneficiario, un nuevo solicitante y un nuevo expediente.
3. En caso de que el beneficiario haya realizado solicitudes previas, se seleccionará en una lista dicho beneficiario, autocompletandose sus datos, y solo se rellenarán los datos del solicitante y la solicitud. Dicha solicitud se asociará al expediente del beneficiario y se creará un nuevo solicitante.
4. En caso de que el solicitante haya realizado solicitudes previas, se seleccionará en una lista dicho solicitante, autocompletandose sus datos, y solo se rellenarán los datos del beneficiario y la solicitud. Dicha solicitud se asociará al solicitante y se creará una nueva solicitud, un nuevo expediente y un nuevo beneficiario.

2.1.1 Caso de uso 1.1: Solicitud sin Beneficiario/Solicitante previo

Caso de uso:	**1.1: Solicitud sin Beneficiario/Solicitante previo**
Descripción:	Se rellenará el formulario completando los datos de la solicitud. Después de validarlos se agregará la solicitud, el solicitante, el beneficiario y se creará un nuevo expediente.
Notas:	Se muestran los detalles de la solicitud añadida

2.1.2 Caso de uso 1.2: Asociar solicitud a Beneficiario

Caso de uso:	**1.2: Asociar solicitud a Beneficiario**
Descripción:	Se seleccionará el beneficiario en la lista y se rellenará el resto formulario completando los datos de la solicitud. Después de validarlos se agregará la solicitud, asociándola al expediente del beneficiario seleccionado y se insertará un nuevo solicitante.
Notas:	Se muestran los detalles de la solicitud añadida

2.1.3 Caso de uso 1.3: Asociar solicitud a Solicitante

Caso de uso:	**1.3: Asociar solicitud a Solicitante**
Descripción:	Se seleccionará el solicitante en la lista y se rellenará el resto formulario completando los datos de la solicitud. Después de validarlos se agregará la solicitud, relacionándola con el solicitante seleccionado y se creará un nuevo beneficiario son su expediente correspondiente.
Notas:	Se muestran los detalles de la solicitud añadida

2.2 Casos de Uso Nueva Intervención

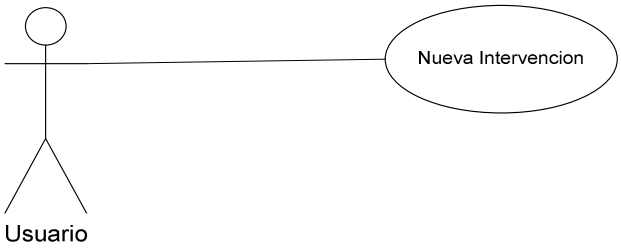


Figura 2. Caso de uso nueva intervención

1. Cuando el usuario desea introducir una nueva solicitud accede a la pantalla de introducción de datos de la intervención.
2. A continuación se seleccionará el expediente al que asociar la intervención.
3. Si existe solicitud previa se seleccionará la solicitud relacionándola con la intervención
4. En caso de que no exista solicitud para esa intervención se creará la intervención sin solicitud asociada.

2.2.1 Caso de uso 2.1: Nueva intervención

Caso de uso:	**2.1: Intervención sin solicitud asociada**
Descripción:	Se seleccionará el expediente/beneficiario y se rellenan los datos relativos a la intervención. Después de validarlos se agregará la intervención.
Notas:	Se muestran los detalles de la intervención añadida

2.3 Casos de Uso Búsqueda

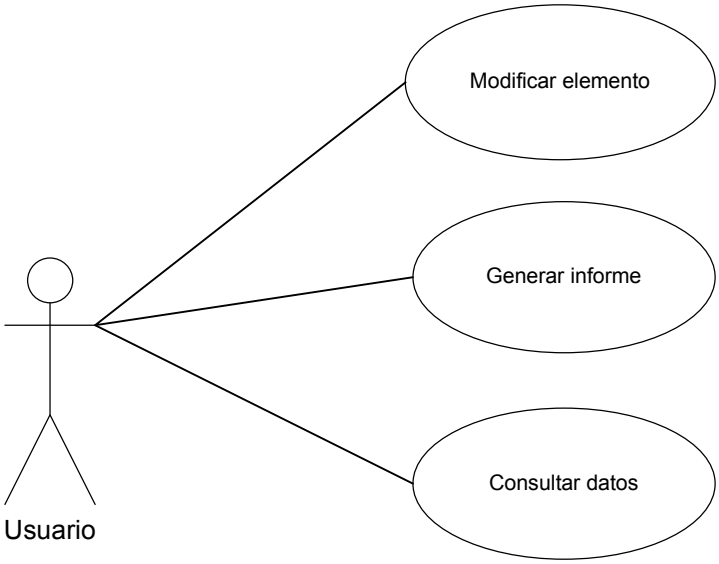


Figura 3. Caso de uso búsqueda

1. Cuando el usuario desea localizar un expediente, beneficiario, solicitante, solicitud o intervención accede a la pantalla de búsqueda y selecciona el tipo de elemento que desea buscar.
2. Posteriormente introducirá información en el filtro de búsqueda y seleccionara qué elemento de la lista resultante de ese filtro quiere consultar.
3. A continuación se muestran los detalles del elemento seleccionado.
4. A través de la pantalla de consulta se puede modificar la información marcada como modificable en el elemento.
5. Asimismo a través de la pantalla de consulta se puede generar un informe.

2.3.1 Caso de uso 3.1: Modificar elemento

Caso de uso:	**3.1: Modificar elemento**
Descripción:	Se accederá a la consulta del elemento seleccionado. A partir de ahí se sobrescribirán los campos que se desean modificar y se aceptarán los cambios.
Notas:	Se muestran los nuevos detalles del elemento modificado

2.3.2 Caso de uso 3.2: Generar informe

Caso de uso:	**3.2 : Generar informe**
Descripción:	Se accederá a la consulta del elemento seleccionado. A partir de se seleccionará la acción de generar informe y se guardará un archivo en el sistema con el informe generado.
Notas:	Se muestran los detalles de la consulta realizada

2.3.3 Caso de uso 3.3: Consultar datos

Caso de uso:	**3.3: Consultar datos**
Descripción:	Se accederá a la consulta del elemento seleccionado. Se mostrará por pantalla los detalles de dicho elemento.
Notas:	Se muestran los detalles de la consulta realizada

2.4 Casos de Uso Estadística

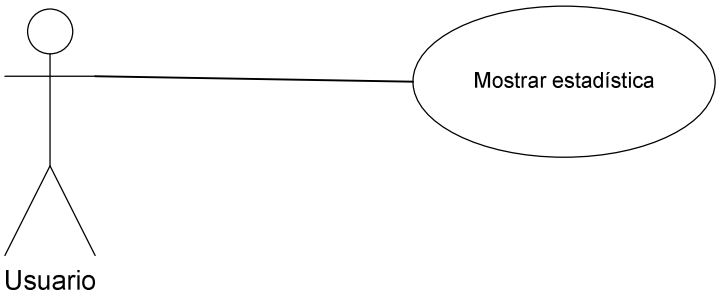


Figura 4. Caso de uso estadística

1. Cuando el usuario desea consultar algún tipo de estadística sobre los datos existentes en el sistema accederá a la pantalla de estadística.
2. Posteriormente introducirá información en el filtro.

2.4.1 Caso de uso 4.1: Mostrar estadística

Caso de uso:	**4.1: Mostrar estadística**
Descripción:	Se seleccionará la estadística a consultar. A continuación se muestra por pantalla el resultado de la estadística.
Notas:	Se muestran los detalles de la estadística consultada.

2.5 Casos de Uso Administración

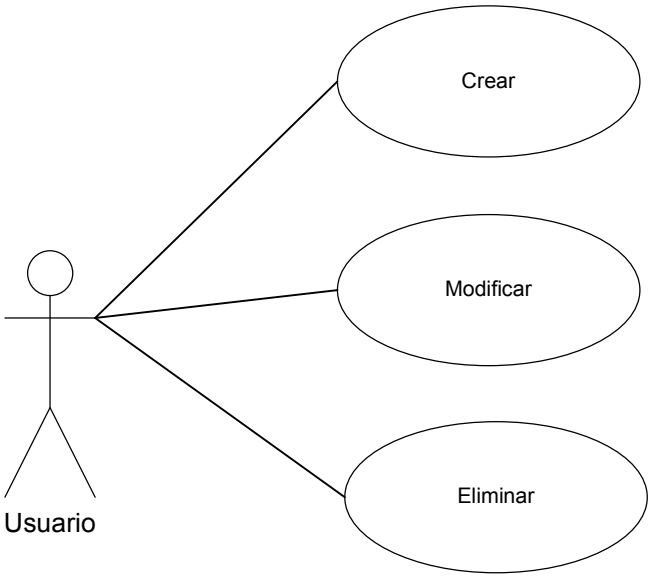


Figura 5. Caso de uso administración

1. Cuando el usuario desea administrar los datos de los formularios accederá a la pantalla de administración.
2. A continuación se seleccionará la categoría de elementos a administrar.
3. Posteriormente se listarán los elementos permitiendo crear, modificar o eliminar algún elemento.

2.5.1 Caso de uso 5.1: Crear

Caso de uso:	**5.1: Crear**
Descripción:	Se seleccionará la opción de crear elemento y se introducirá el nombre del mismo.
Notas:	Se muestran los elementos existentes incluyendo el elemento añadido.

2.5.2 Caso de uso 5.2: Modificar

Caso de uso:	**5.2: Modificar**
Descripción:	Se seleccionará la opción de modificar elemento y se introducirá el nuevo nombre del mismo.
Notas:	Se muestran los elementos existentes incluyendo el elemento modificado.

2.5.3 Caso de uso 5.3: Eliminar

Caso de uso:	**5.3: Eliminar**
Descripción:	Se seleccionará la opción de eliminar elemento.
Notas:	Se muestran los elementos existentes excluyendo el elemento eliminado.

3. Escenarios

3.1 Caso de uso 1: Nueva Solicitud

Caso de uso:	**1.1: Solicitud sin Beneficiario/Solicitante previo**
Actor:	Usuario
Detalle de operaciones:	La aplicación inserta en la base de datos la información relativa al beneficiario, solicitud, solicitante y expediente.
Precondiciones:	Se deben introducir los datos necesarios y válidos
Poscondiciones:	Se guardará en la base de datos la información introducida y los datos de relación entre los elementos.

Caso de uso:	**1.2: Asociar solicitud a Beneficiario**
Actor:	Usuario
Detalle de operaciones:	La aplicación inserta en la base de datos la información relativa al beneficiario, solicitud, solicitante y expediente.
Precondiciones:	Se deben introducir los datos necesarios y válidos y el beneficiario debe existir en el sistema
Poscondiciones:	Se guardará en la base de datos la información introducida y los datos de relación entre los elementos.

Caso de uso:	**1.2: Asociar solicitud a Solicitante**
Actor:	Usuario
Detalle de operaciones:	La aplicación inserta en la base de datos la información relativa al beneficiario, solicitud, solicitante y expediente.
Precondiciones:	Se deben introducir los datos necesarios y válidos y el solicitante debe existir en el sistema
Poscondiciones:	Se guardará en la base de datos la información introducida y los datos de relación entre los elementos.

3.2 Caso de uso 2: Nueva Intervención

Caso de uso:	**2.1: Nueva intervención**
Actor:	Usuario
Detalle de operaciones:	La aplicación inserta en la base de datos la información relativa la intervención y el expediente.
Precondiciones:	Se deben introducir los datos necesarios y válidos y debe existir el beneficiario y expediente en el sistema.
Poscondiciones:	Se guardará en la base de datos la información introducida y los datos de relación entre los elementos.

3.3 Caso de uso 3: Búsqueda

Caso de uso:	**3.1: Modificar elemento**
Actor:	Usuario
Detalle de operaciones:	La aplicación modifica en la base de datos la información relativa al elemento seleccionado.
Precondiciones:	Se deben introducir los nuevos datos válidos y debe existir el elemento en el sistema.
Poscondiciones:	Se sobrescribirá en el sistema la nueva información introducida.

Caso de uso:	**3.2: Generar informe**
Actor:	Usuario
Detalle de operaciones:	La aplicación genera un archivo en formato Word y lo guarda en el directorio determinado en la configuración.
Precondiciones:	Debe existir el elemento del que generar informe en el sistema.
Poscondiciones:	-

Caso de uso:	**3.3: Consultar datos**
Actor:	Usuario
Detalle de operaciones:	La aplicación accede a la base de datos y muestra a través de la interfaz los datos requeridos.
Precondiciones:	Se debe seleccionar un elemento existente en el sistema.
Poscondiciones:	Se muestra en pantalla los detalles del elemento consultado.

3.4 Caso de uso 4: Estadística

Caso de uso:	**4.1: Mostrar estadística**
Actor:	Usuario
Detalle de operaciones:	La aplicación accede a la base de datos, realiza los cálculos necesarios y muestra a través de la interfaz los resultados.
Precondiciones:	Se debe seleccionar el filtro deseado.
Poscondiciones:	Se muestra en pantalla los detalles de la estadística consultada.

3.5 Caso de uso 5: Administración

Caso de uso:	**5.1: Crear**
Actor:	Usuario
Detalle de operaciones:	La aplicación inserta en la base de datos la nueva información introducida.
Precondiciones:	Se deben introducir los datos válidos
Poscondiciones:	Se guardará en la base de datos la información introducida.

Caso de uso:	**5.2: Modificar**
Actor:	Usuario
Detalle de operaciones:	La aplicación sobrescribe en la base de datos la nueva información introducida.
Precondiciones:	Se deben introducir los datos válidos y debe existir el elemento en el sistema.
Poscondiciones:	Se guardará sobrescribirá en la base de datos la nueva información.

Caso de uso:	**5.3: Eliminar**
Actor:	Usuario
Detalle de operaciones:	La aplicación modifica en la base de datos la información relativa al elemento seleccionado.
Precondiciones:	Debe existir el elemento en el sistema.
Poscondiciones:	Se modificará en la base de datos la información sobre el elemento.

4. Diagrama de clases

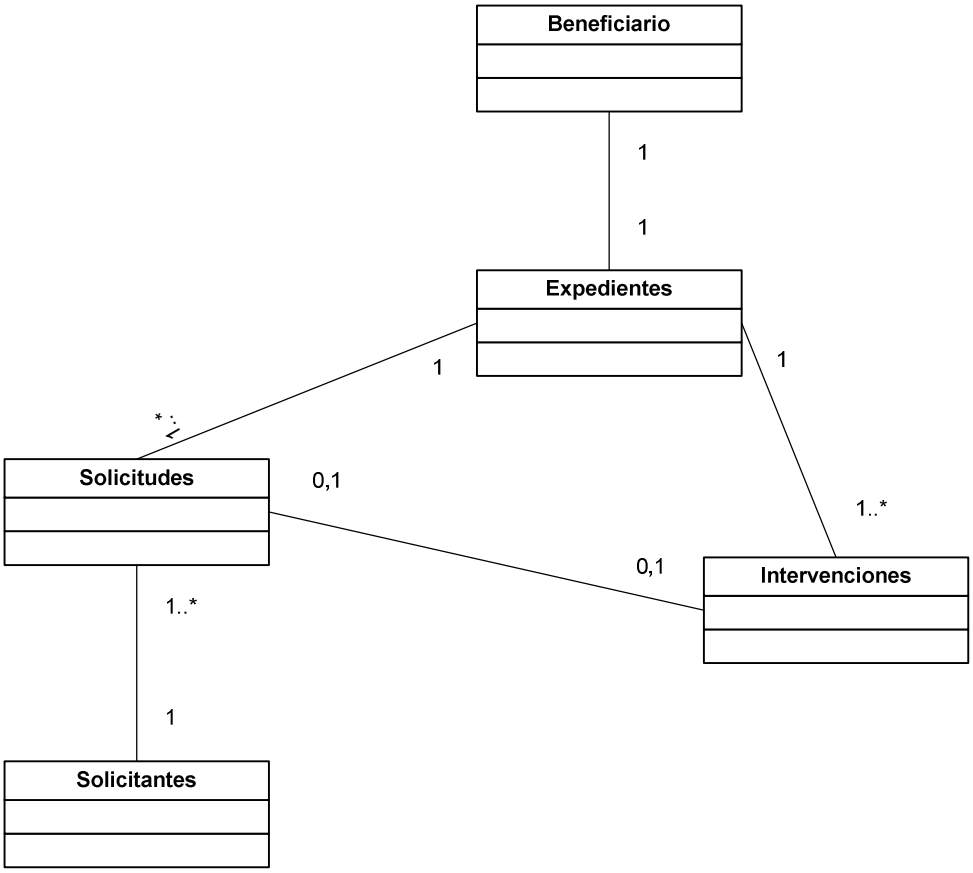


Figura 02. Diagrama de clases

4.1. Descripción de las clases

- **Solicitudes**: Clase que representa el elemento solicitud.
- **Solicitantes**: Clase que representa a las personas que presentan las solicitudes.
- **Beneficiarios**: Clase que representa a las personas beneficiarias de las solicitudes.
- **Expedientes**: Clase que representa el expediente que relaciona al beneficiario con el resto de elementos.
- **Intervenciones**: Clase que representa el elemento intervencion.

5. Interfaces de usuario

La aplicación posee un interfaz propio de una aplicación web, teniendo un *placeholder* como elemento fijo en toda la navegación y cambiando la página de contenido según el usuario navega por la aplicación.

Además la aplicación posee una barra de navegación que permite ir de manera rápida a la vista de una funcionalidad concreta.

En cuanto a la navegación se realizará a través de botones y vínculos que permitirá guiar al usuario para seguir los pasos necesarios para acceder a las opciones de la aplicación de manera adecuada.

5.1. Mapa de pantallas

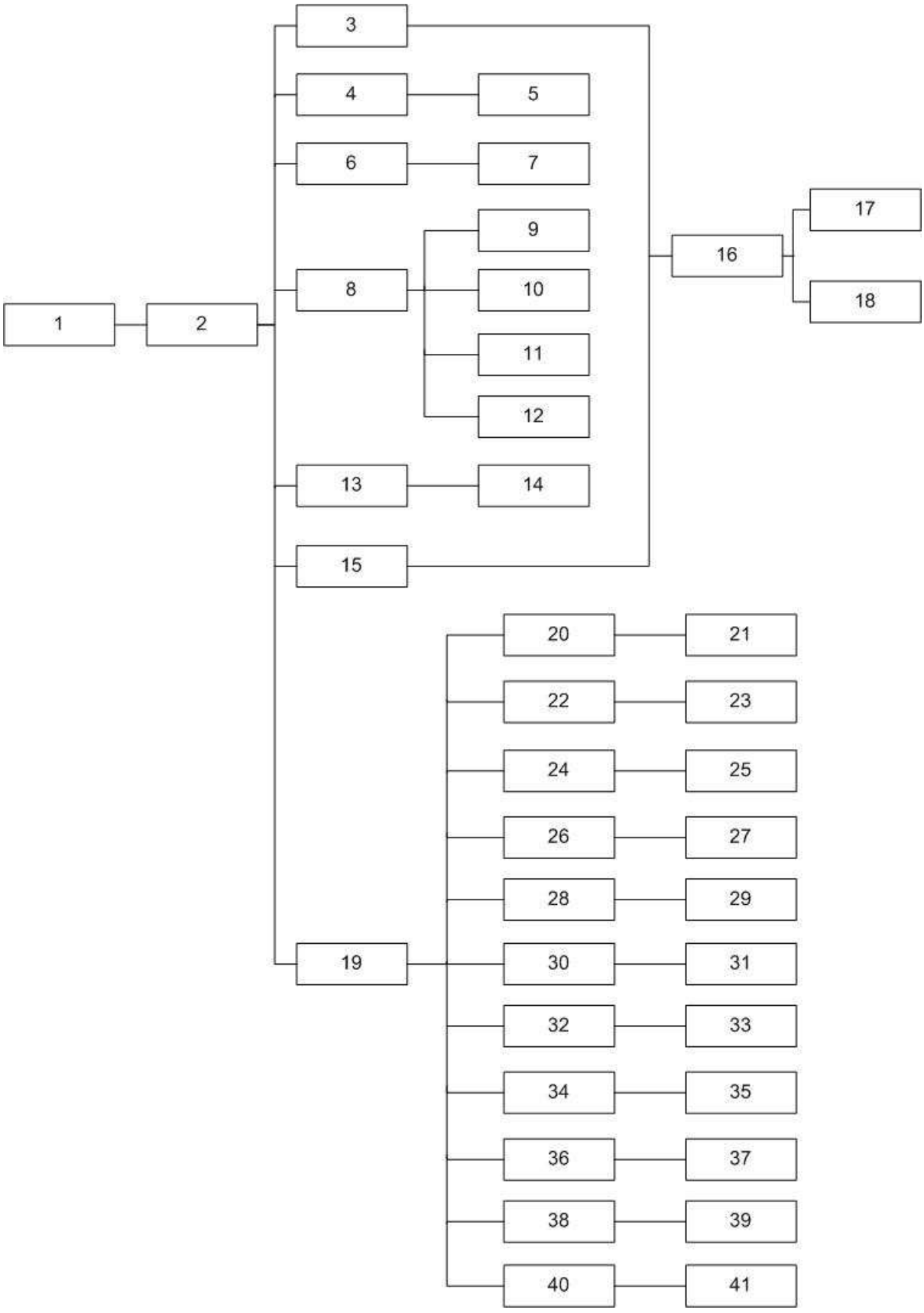


Figura 6. Mapa de pantallas

5.2. Pantallas

1. Identificación
2. Home
3. Nueva solicitud
4. Nuevo solicitante
5. Listado solicitantes
6. Nuevo beneficiario
7. Listado beneficiarios
8. Estadística
9. Estadística por beneficiario
10. Estadística por intervenciones
11. Estadística por expedientes
12. Expedientes nuevos en fecha
13. Listado de solicitudes
14. Destalles de solicitud
15. Listado expedientes
16. Detalles expedientes
17. Nueva intervención
18. Detalles intervención
19. Administrar
20. Administrar tipos de demandas
21. Añadir tipo de demanda
22. Administrar grado de dependencia
23. Añadir grado de dependencia
24. Administrar tipo de discapacidad
25. Añadir tipo de discapacidad
26. Administrar tipo de entidad
27. Añadir tipo de entidad
28. Administrar situación funcional
29. Añadir situación funcional
30. Administrar puestos
31. Añadir puesto
32. Administrar tipos de entrada de solicitud
33. Añadir tipo de entrada de solicitud
34. Administrar tipo de solicitante
35. Añadir tipo de solicitante
36. Administrar tipo de beneficiario
37. Añadir tipo de beneficiario
38. Administrar relaciones con el beneficiario
39. Añadir relación con el beneficiario
40. Administrar usuarios
41. Añadir usuario

5.2.1 Identificación

En esta pantalla la aplicación permitirá al usuario introducir sus datos de identificación para poder acceder a las funciones de la aplicación.

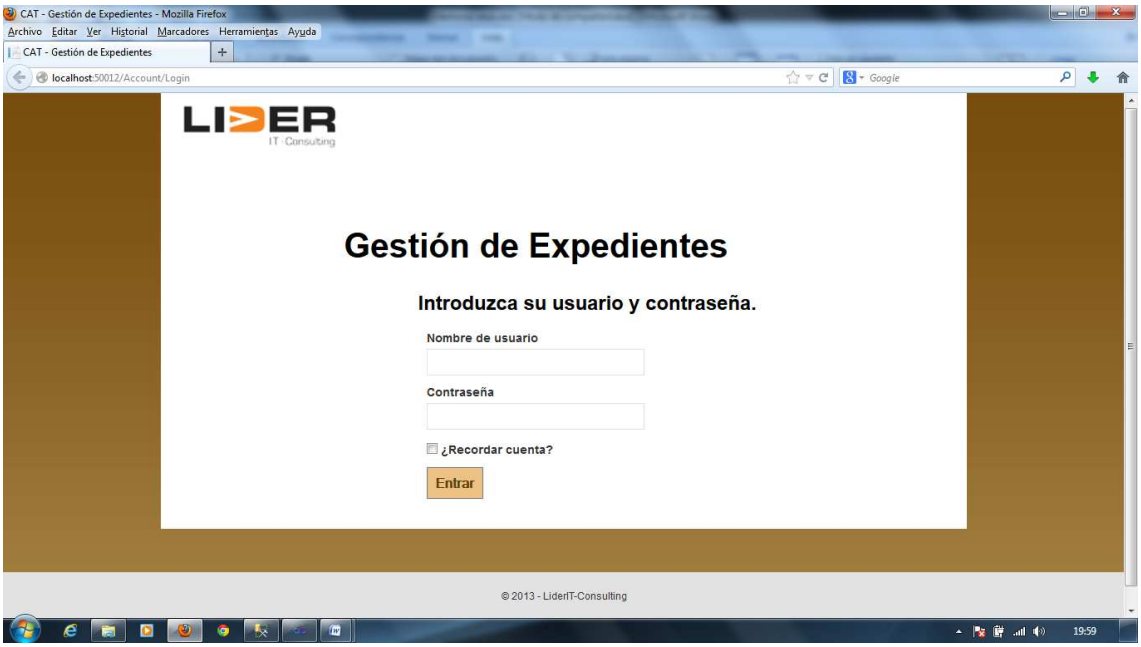


Figura 7. Pantalla identificación

5.2.2 Home

Desde esta pantalla se podrá acceder a todas las funciones que ofrece la aplicación.

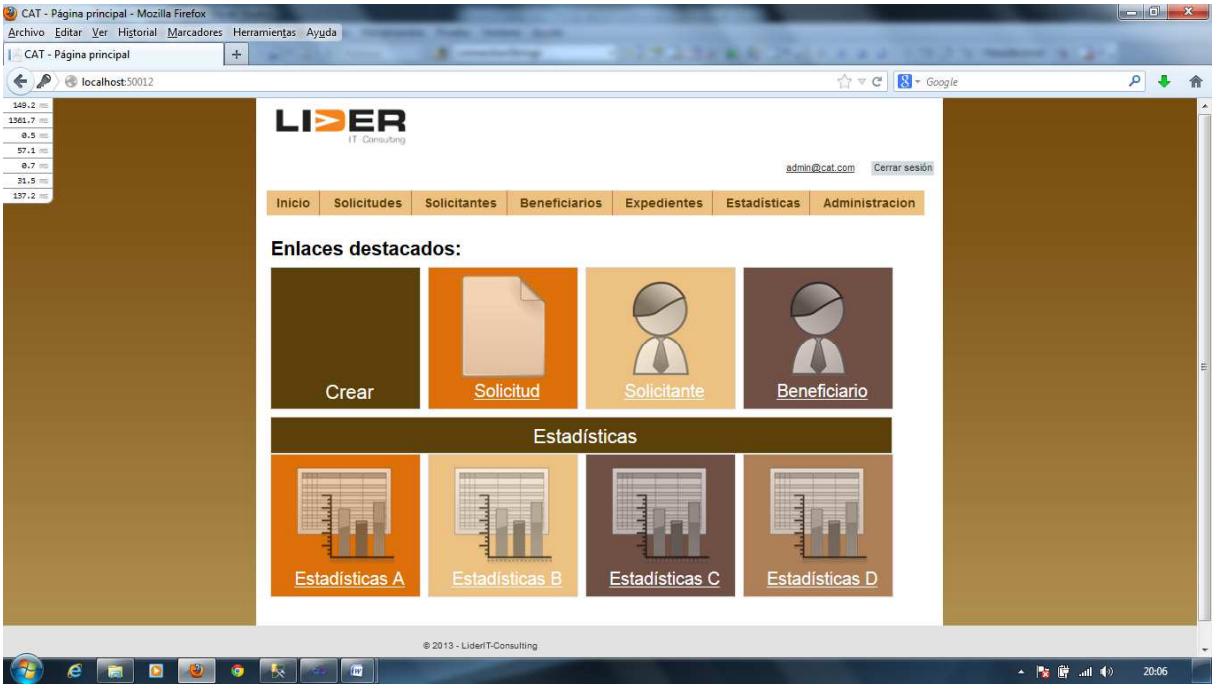


Figura 8. Pantalla home

5.2.3 Nueva solicitud

En esta pantalla se introducirán los datos necesarios para crear una nueva solicitud. Esta vista está formada por varias vistas parciales que se desplegarán según se hace click en ellas. Las vistas parciales son: datos de solicitante, datos del beneficiario, discapacidades, limitaciones funcionales y demandas.

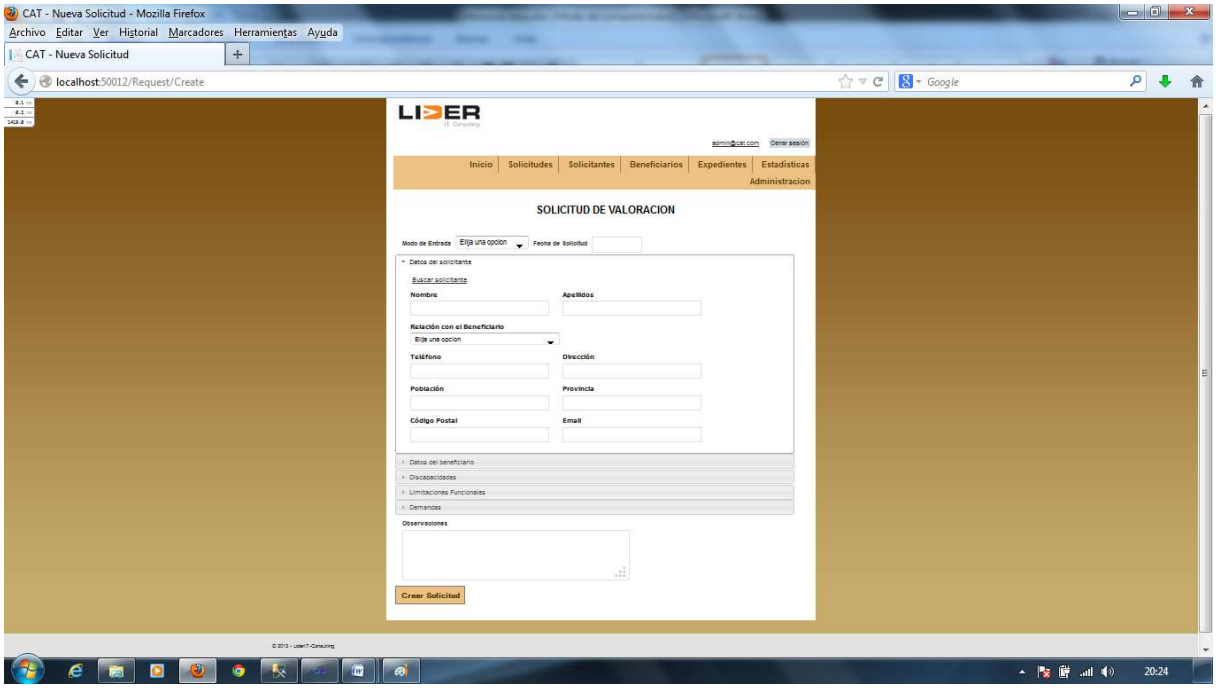


Figura 9. Pantalla nueva solicitud (I)

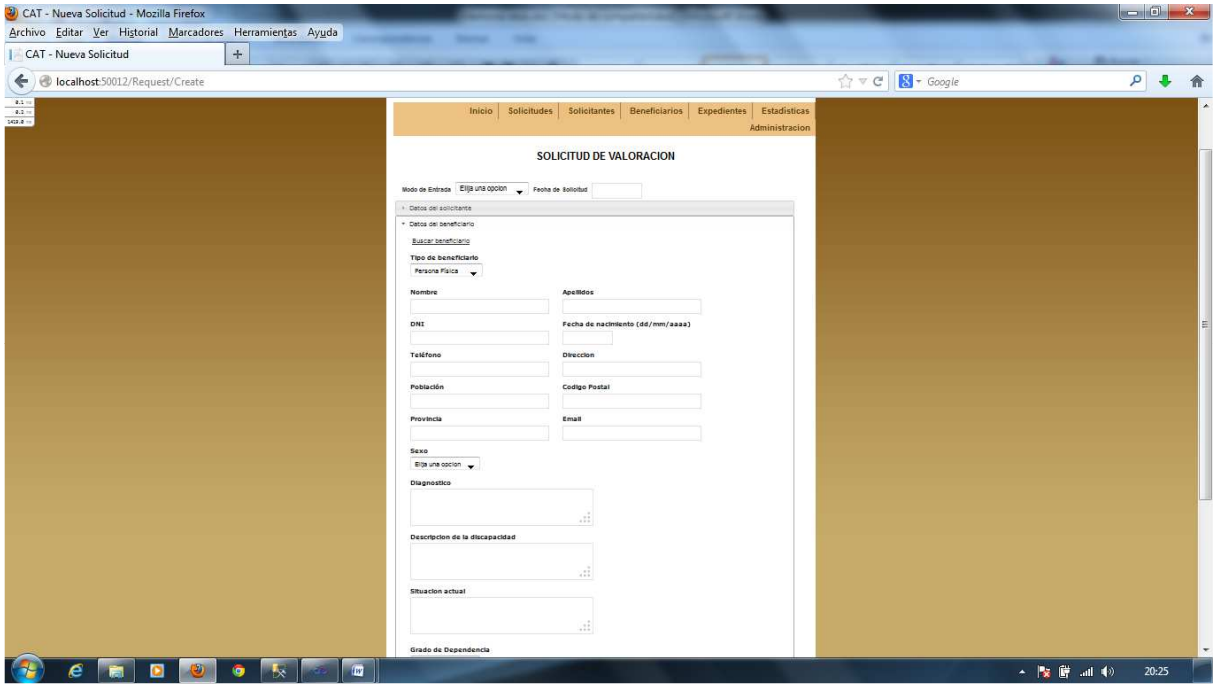


Figura 10. Pantalla nueva solicitud (II)

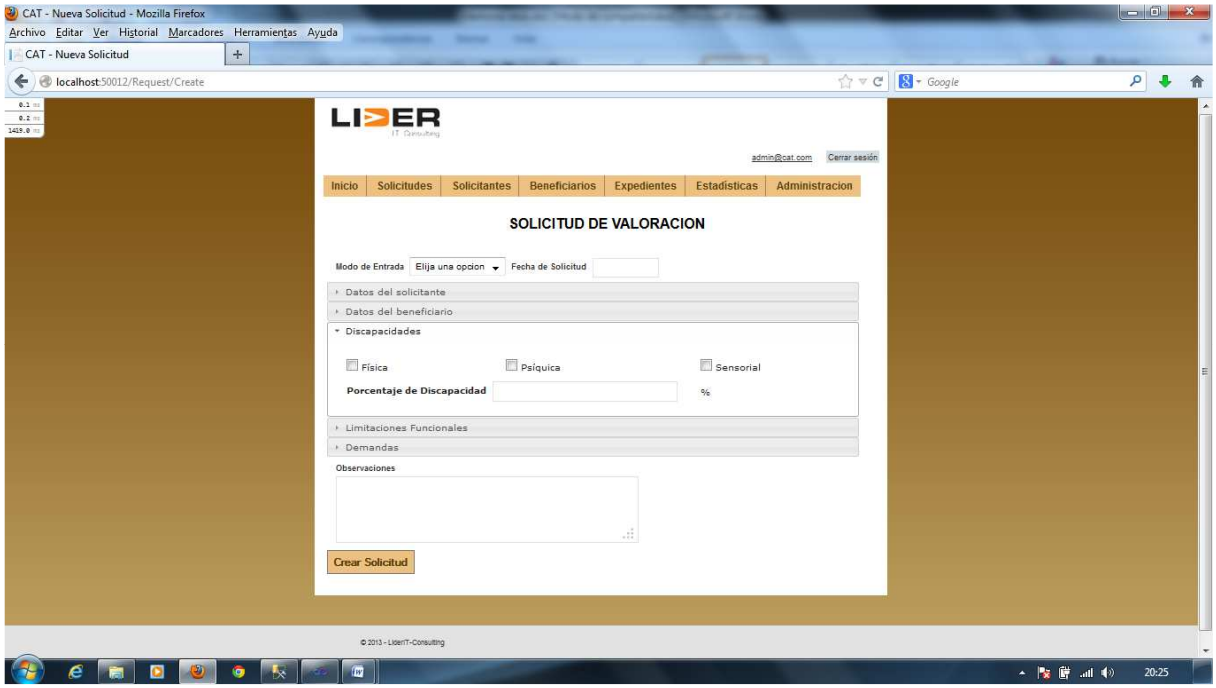


Figura 11. Pantalla nueva solicitud (III)

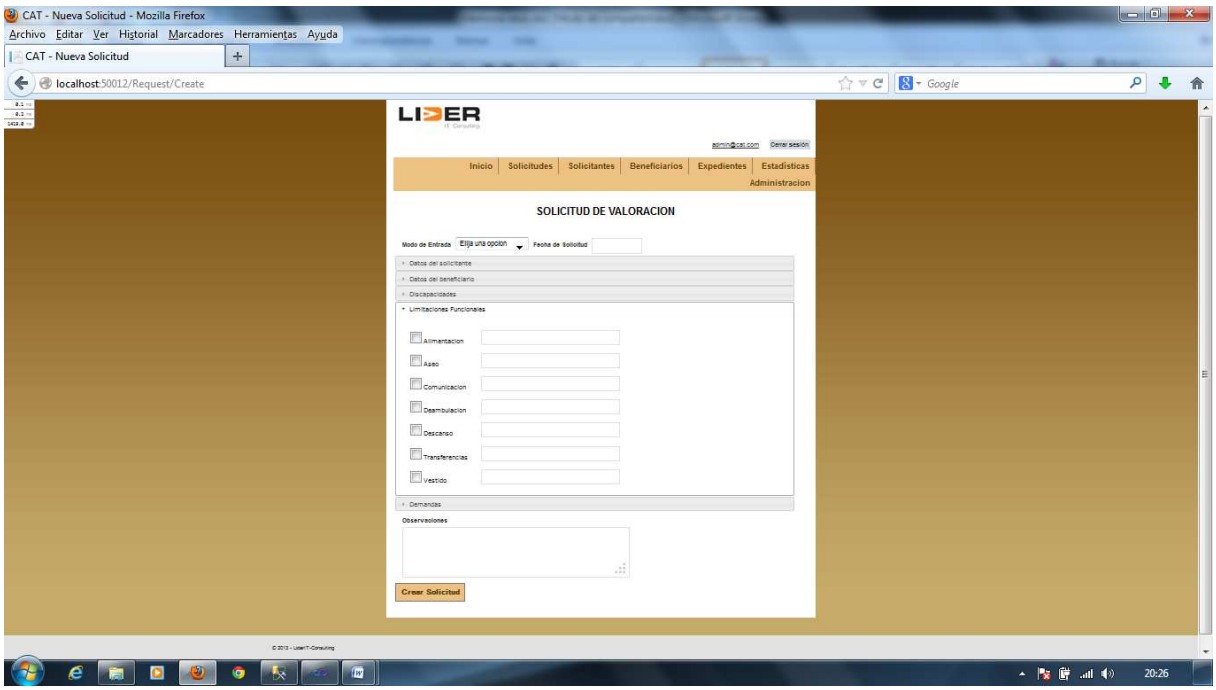


Figura 12. Pantalla nueva solicitud (IV)

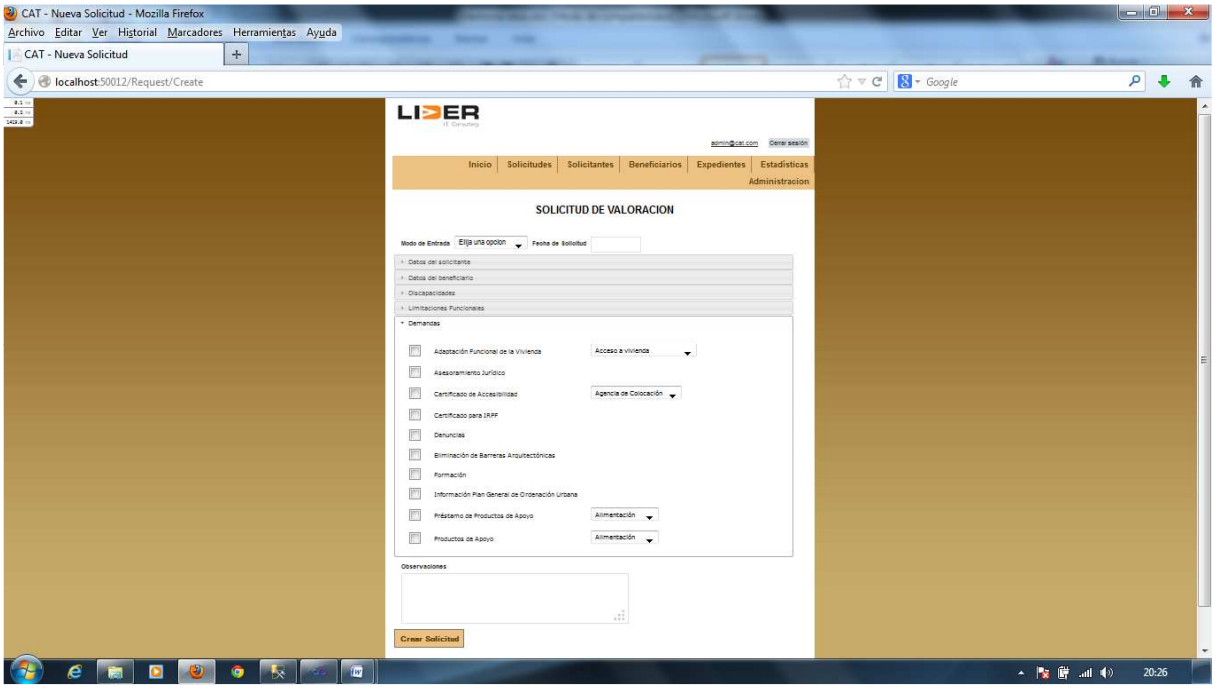


Figura 13. Pantalla nueva solicitud (V)

5.2.4 Nuevo solicitante

A través de esta pantalla se pueden introducir los datos de un solicitante sin que exista una solicitud del mismo.

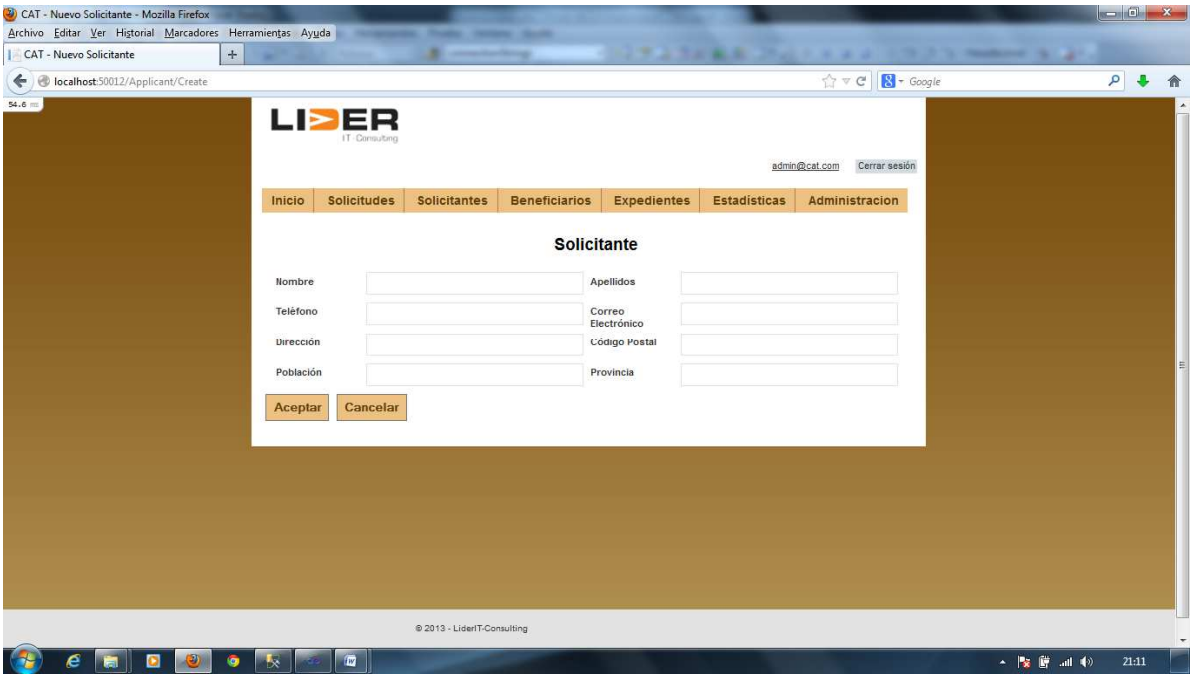


Figura 14. Pantalla nuevo solicitante

5.2.5 Listado de solicitantes

A esta pantalla se puede acceder a través de la barra superior de acceso rápido o se muestra después de crear un solicitante y en ella se puede buscar un solicitante en concreto para modificarlo o eliminarlo.

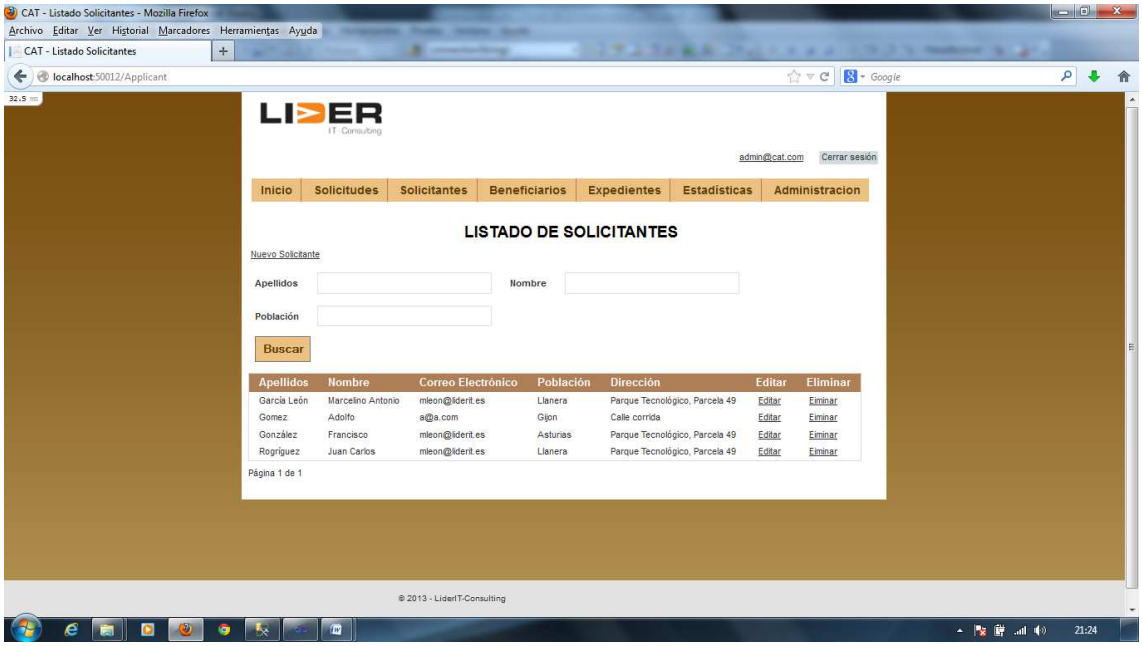


Figura 15. Pantalla listado solicitantes

5.2.6 Nuevo beneficiario

A través de esta pantalla se pueden introducir los datos de un beneficiario sin que exista una solicitud del mismo.

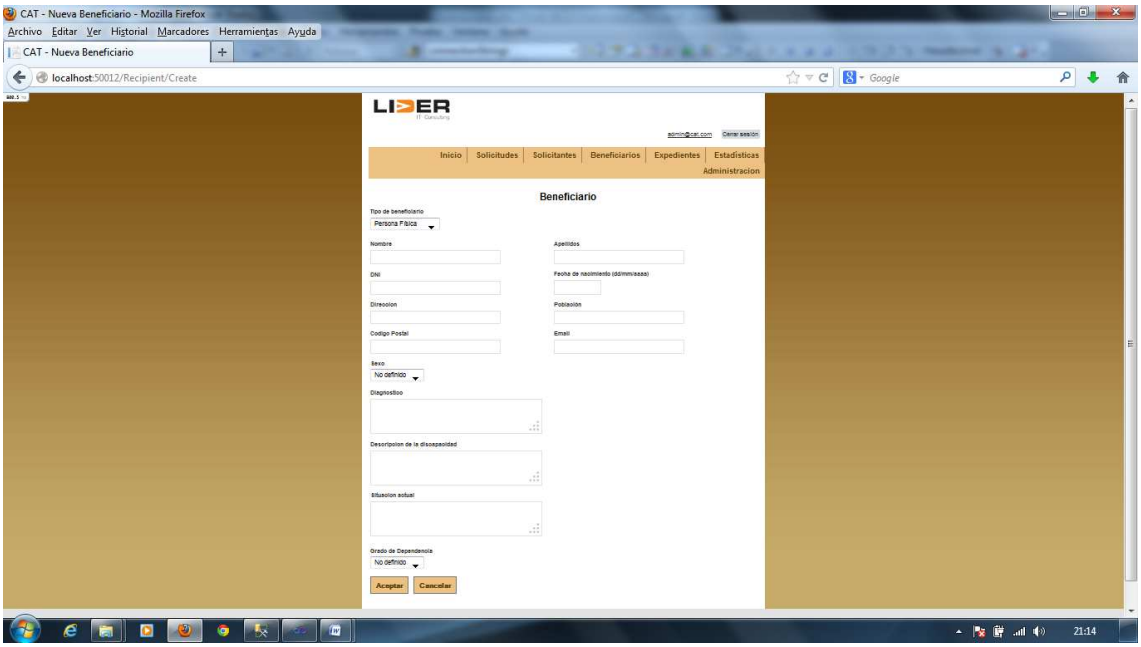


Figura 16. Pantalla nuevo beneficiario

5.2.7 Listado de beneficiario

A esta pantalla se puede acceder a través de la barra superior de acceso rápido o se muestra después de crear un beneficiario y en ella se puede buscar un beneficiario en concreto para modificarlo o eliminarlo.

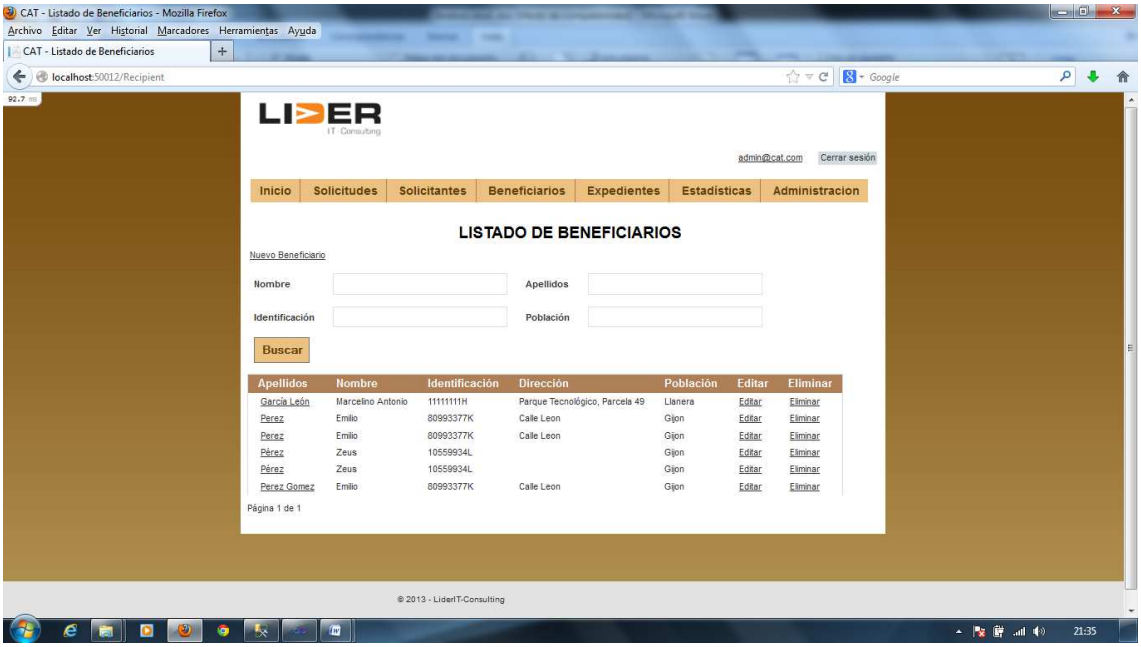


Figura 17. Pantalla listado de beneficiario

5.2.8 Estadísticas

A través de esta pantalla se puede acceder a las estadísticas definidas en el sistema.

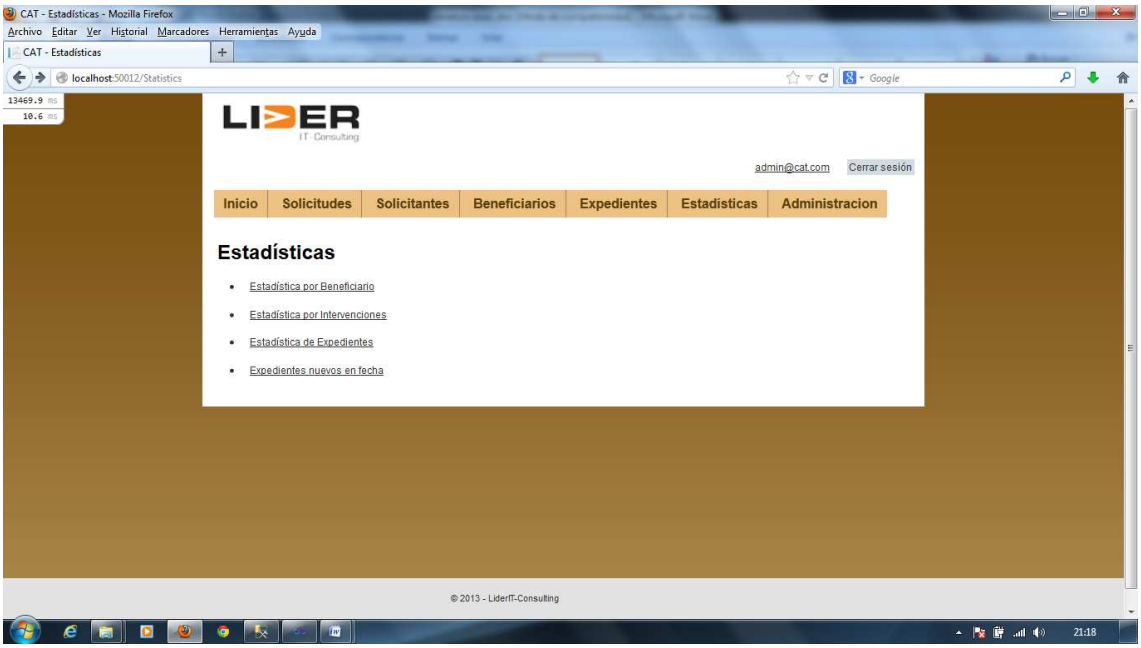


Figura 18. Pantalla estadística

5.2.9 Estadísticas por beneficiario

En esta pantalla se muestran estadísticas sobre los expedientes según las características del beneficiario

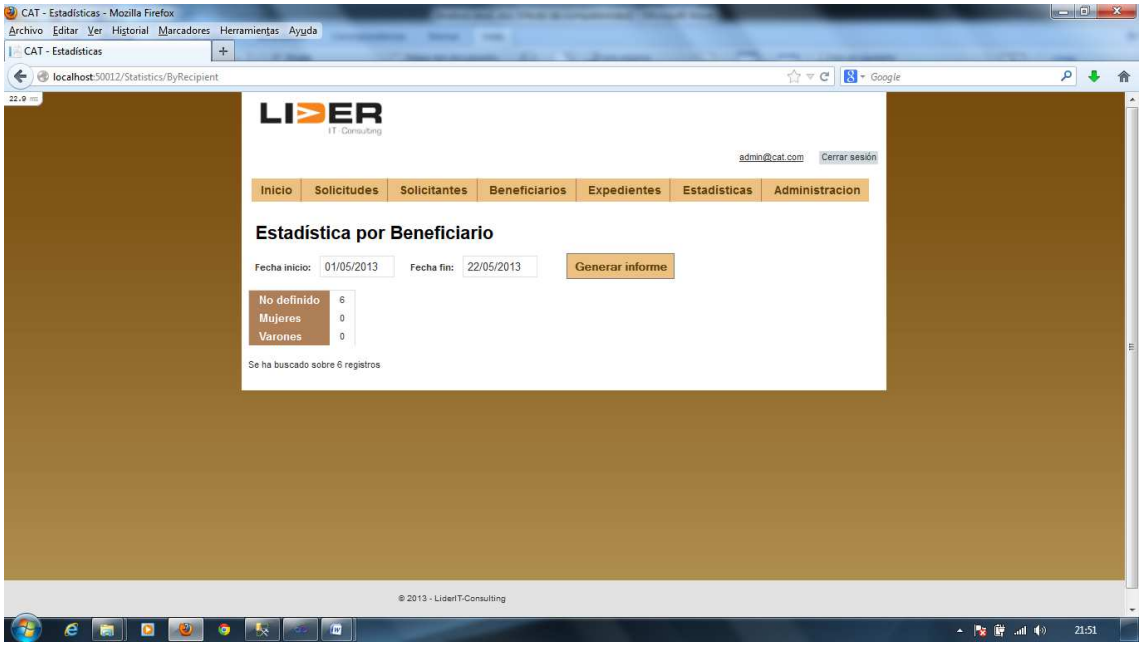


Figura 19. Pantalla estadísticas por beneficiario

5.2.10 Estadísticas por intervenciones

En esta pantalla se muestran estadísticas sobre los expedientes según las características de las intervenciones.

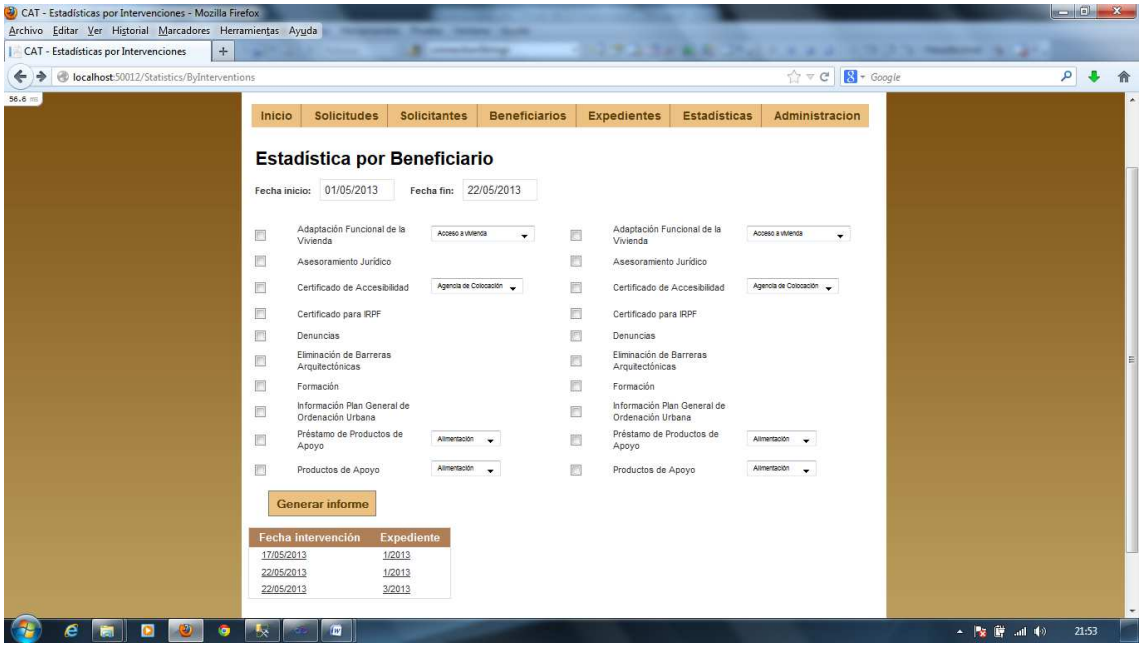


Figura 20. Estadísticas por intervenciones

5.2.11 Estadísticas por expedientes

En esta pantalla se muestran estadísticas sobre los expedientes según las características de los expedientes.

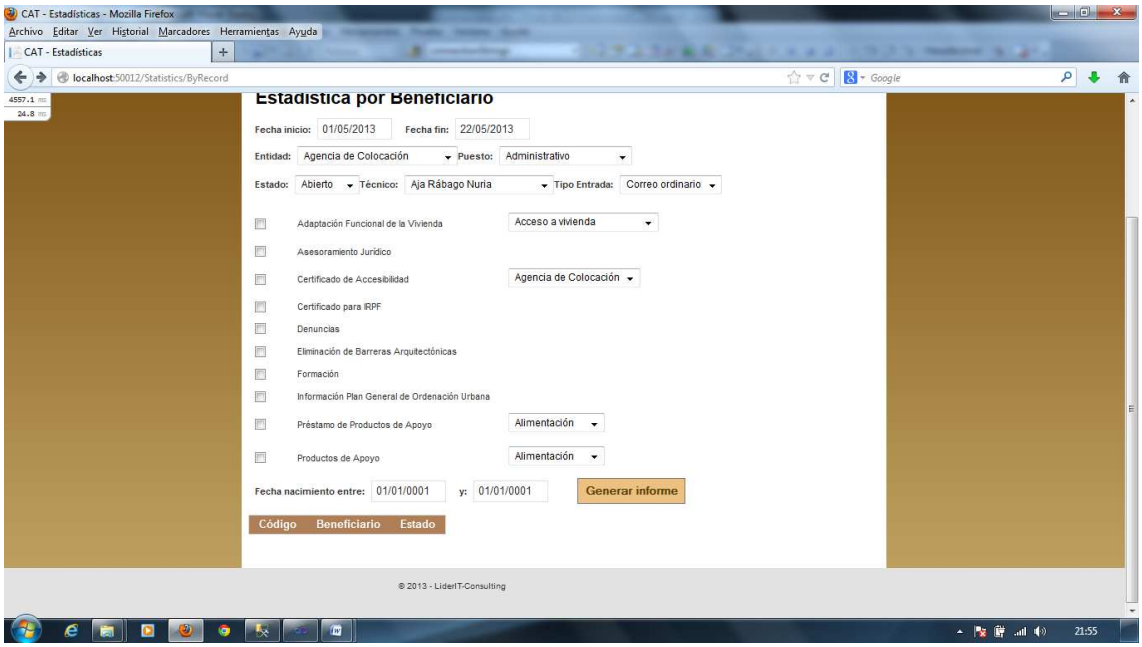


Figura 21. Estadísticas por expediente

5.2.12 Expedientes nuevos en fecha

Muestra los expedientes creados entre las fechas seleccionadas.

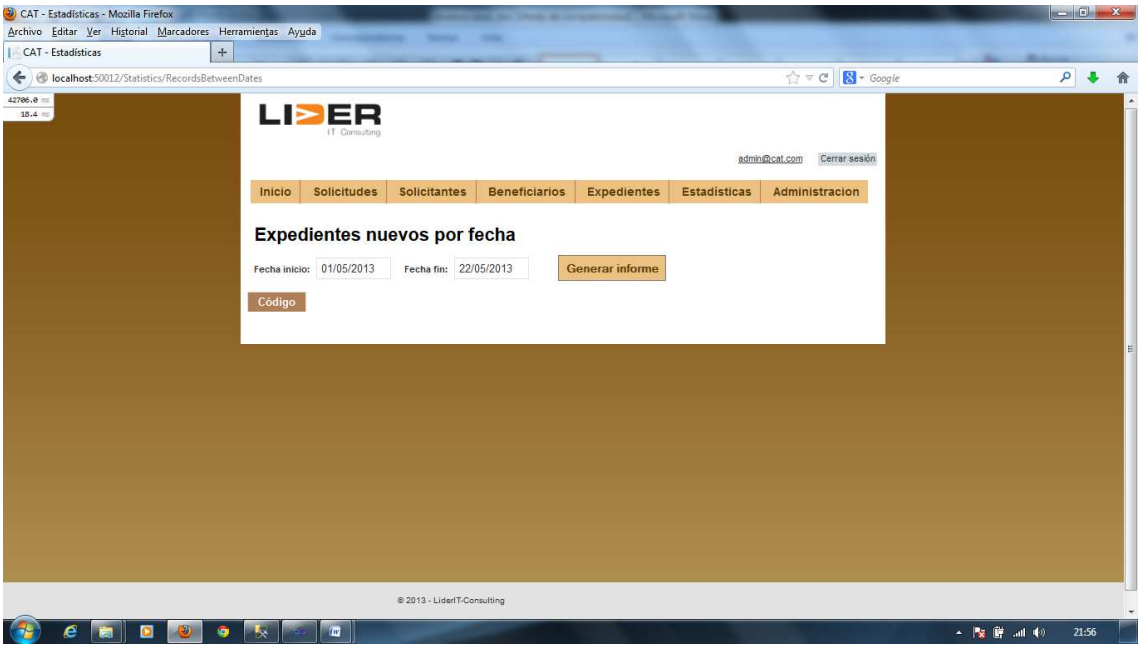


Figura 22. Pantalla expedientes nuevos en fecha

5.2.13 Listado de solicitudes

A esta pantalla se puede acceder a través de la barra superior de acceso rápido y en ella se puede buscar una solicitud en concreto.

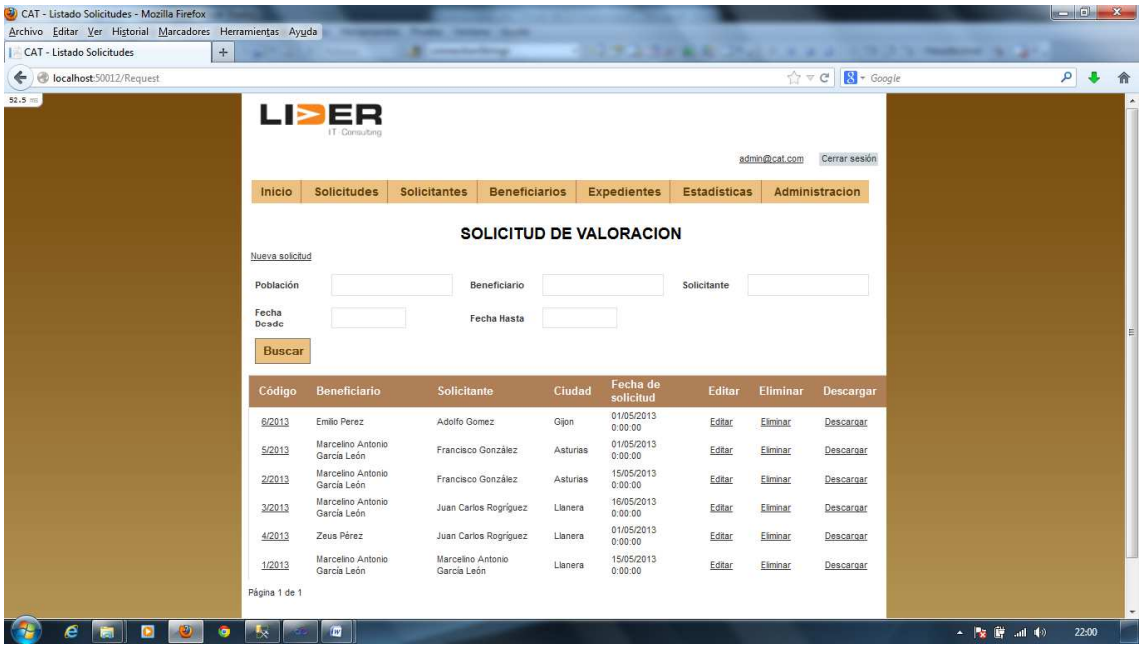


Figura 23. Pantalla listado de solicitudes

5.2.14 Detalles de solicitud

En esta pantalla se podrán consultar los detalles . Esta vista está formada por varias vistas parciales que se desplegarán según se hace click en ellas. Las vistas parciales son: datos de solicitante, datos del beneficiario, discapacidades, limitaciones funcionales y demandas.

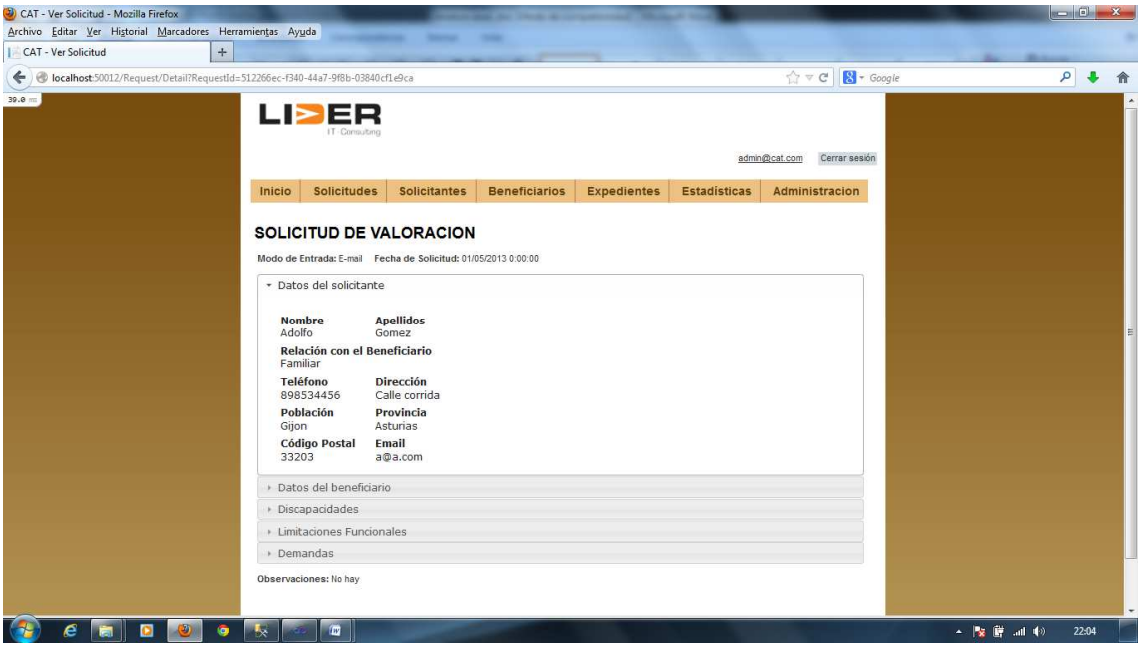


Figura 24. Pantalla detalles de solicitud (I)

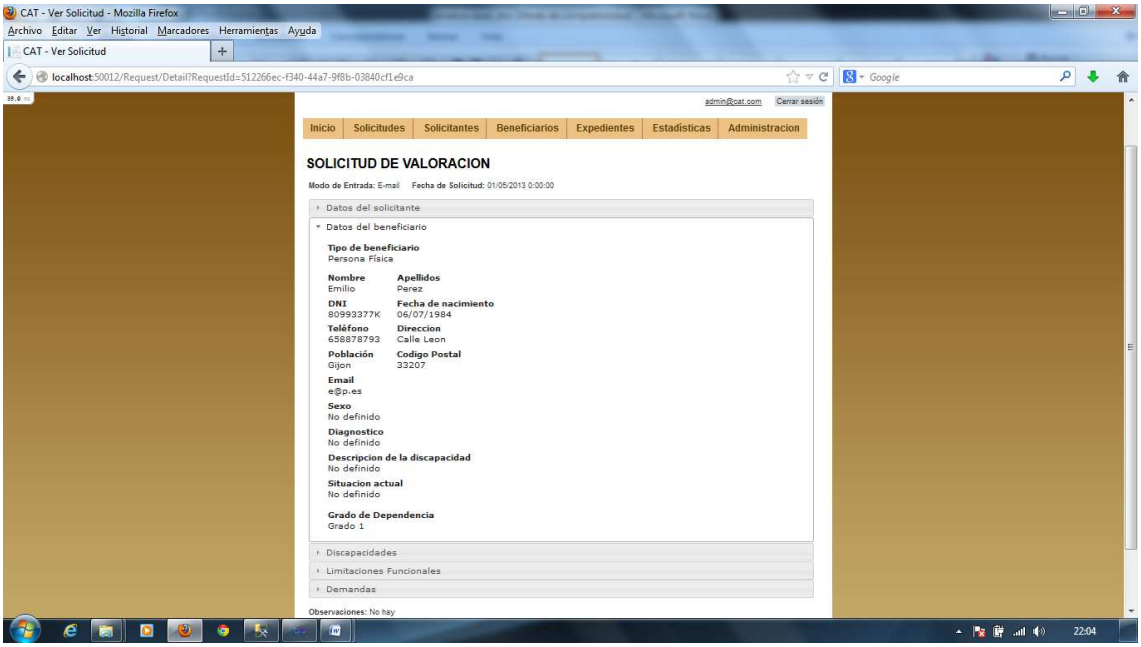


Figura 25. Pantalla detalles solicitud (II)

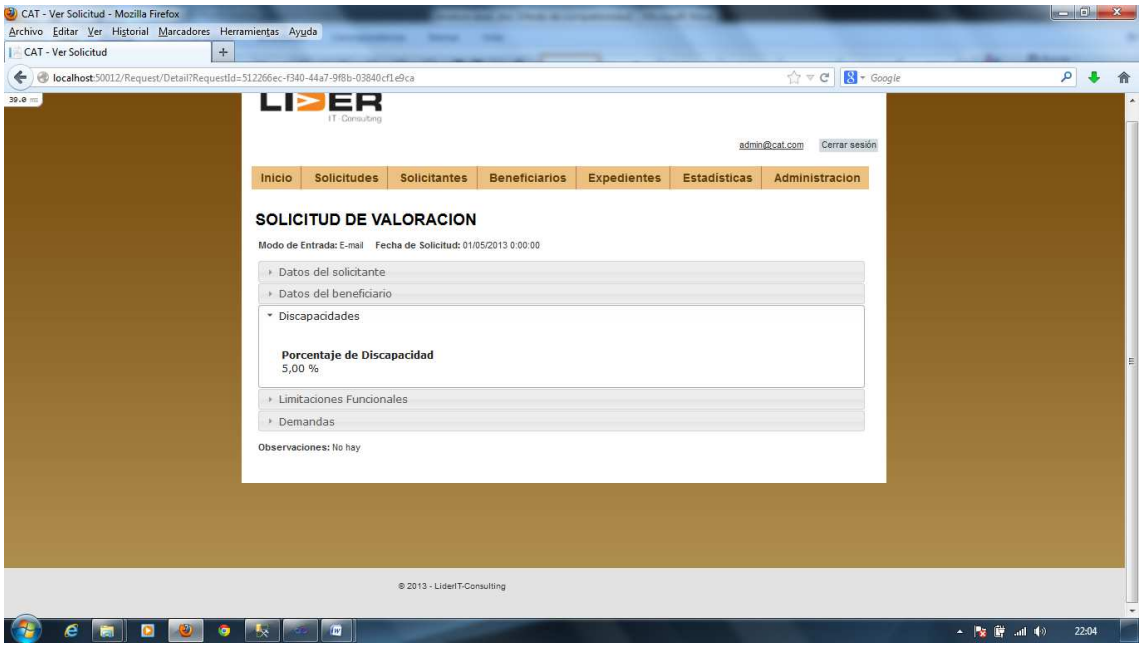


Figura 26. Pantalla detalles solicitud (III)

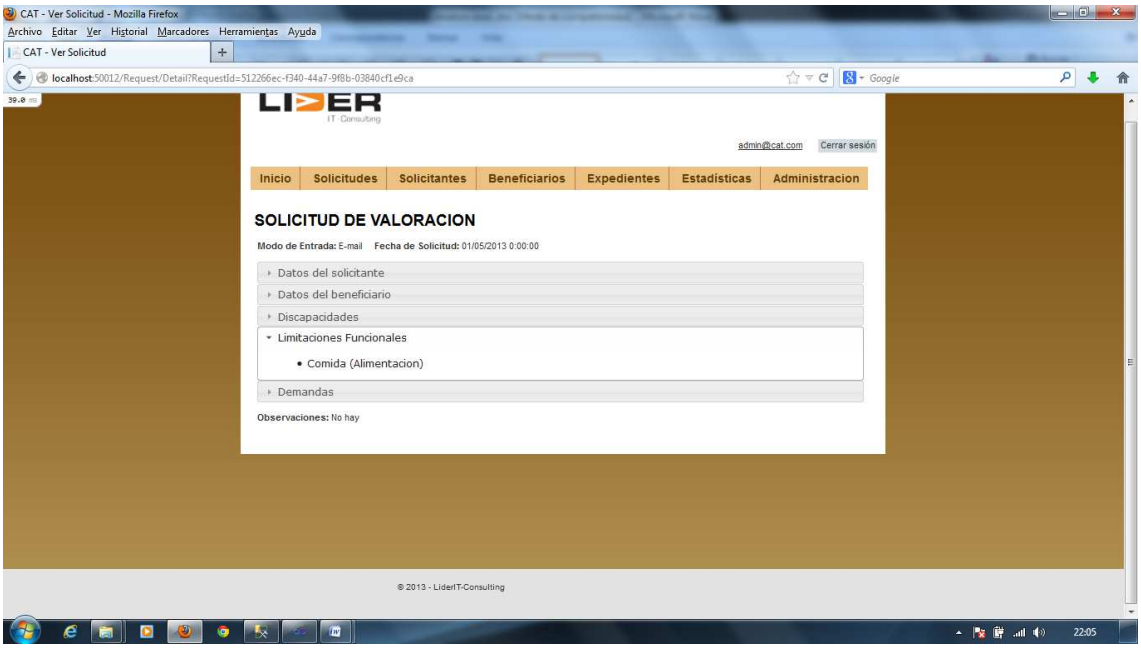


Figura 27. Pantalla detalles solicitud (IV)

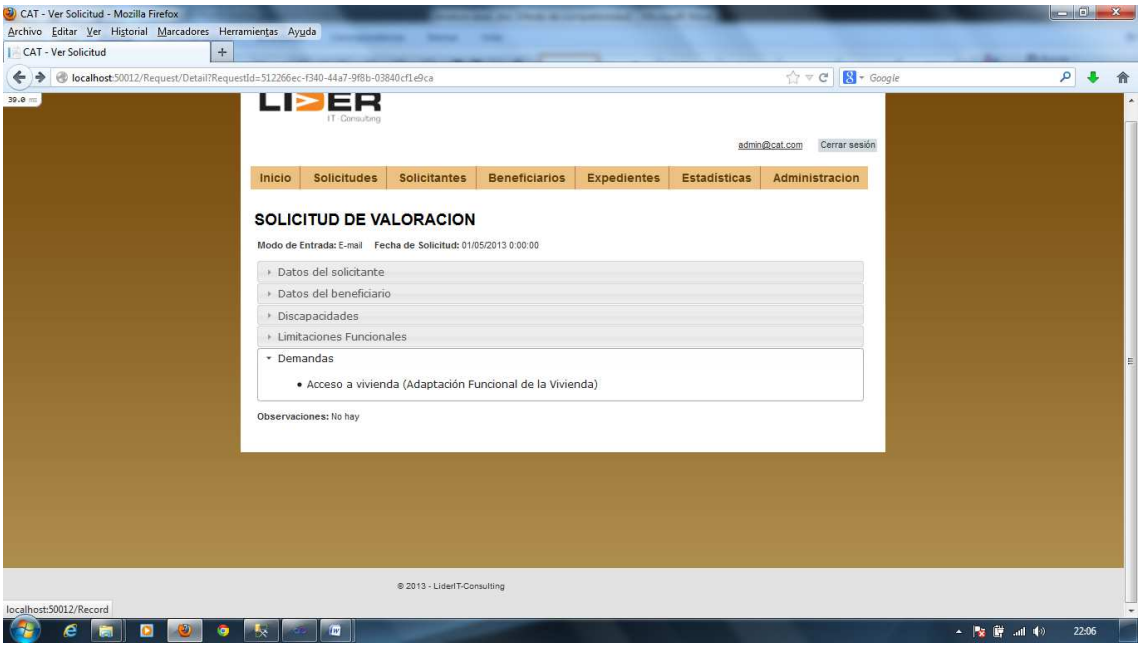


Figura 28. Pantalla detalles de solicitud (V)

5.2.15 Listado de expedientes

A esta pantalla se puede acceder a través de la barra superior de acceso rápido y en ella se puede buscar un expediente en concreto.

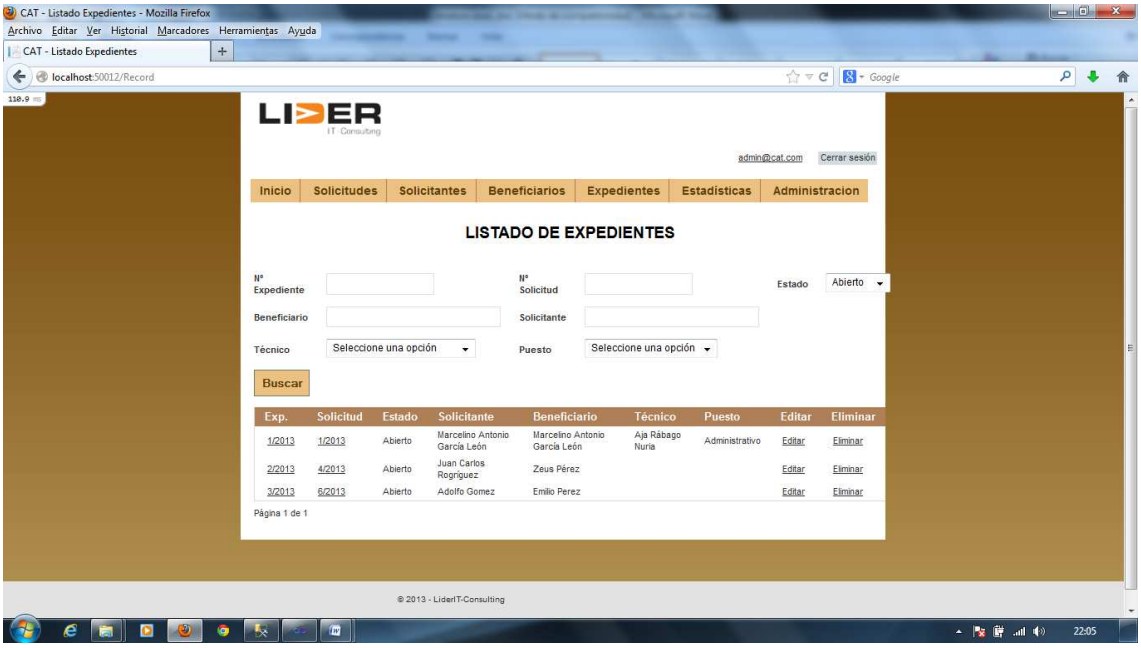


Figura 29. Pantalla listado de expedientes

5.2.16 Detalles de expediente

En esta pantalla se podrán consultar los detalles de un expediente en concreto . Esta vista está formada por varias vistas parciales que se mostrarán según se hace click en ellas. Las vistas parciales son: expediente, intervenciones y gestor documental.
Desde la vista parcial de intervenciones se podrán ver las intervenciones existentes y se podrán añadir intervenciones nuevas y a través de la vista parcial del gestor documental se podrán añadir archivos al expediente.

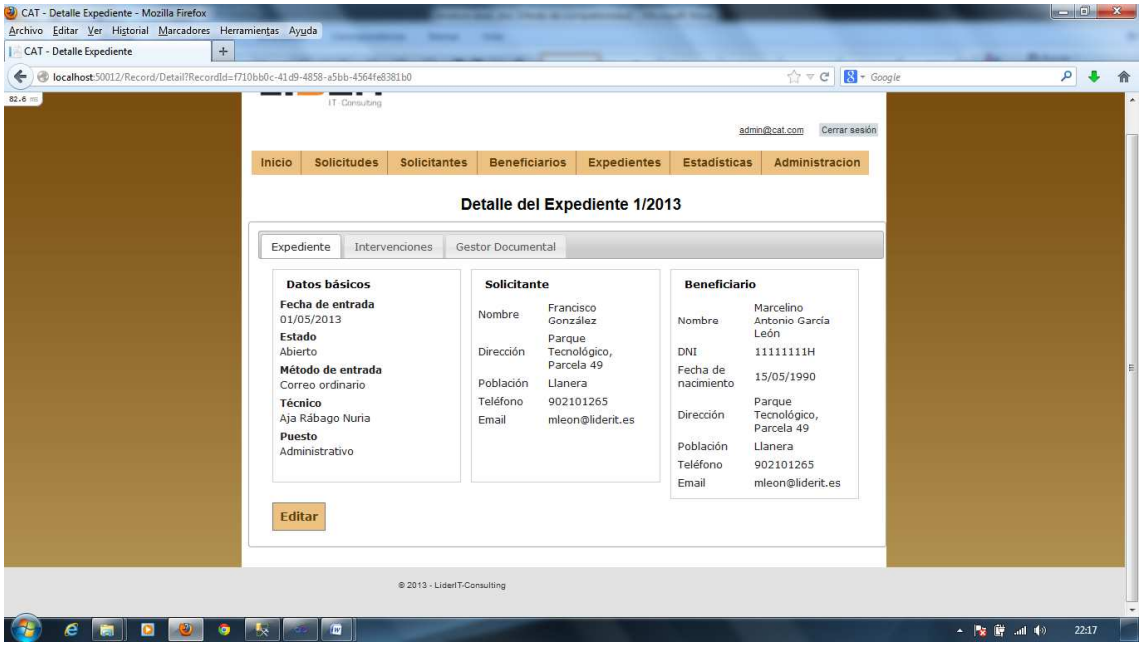


Figura 30. Pantalla detalles de expediente (I)

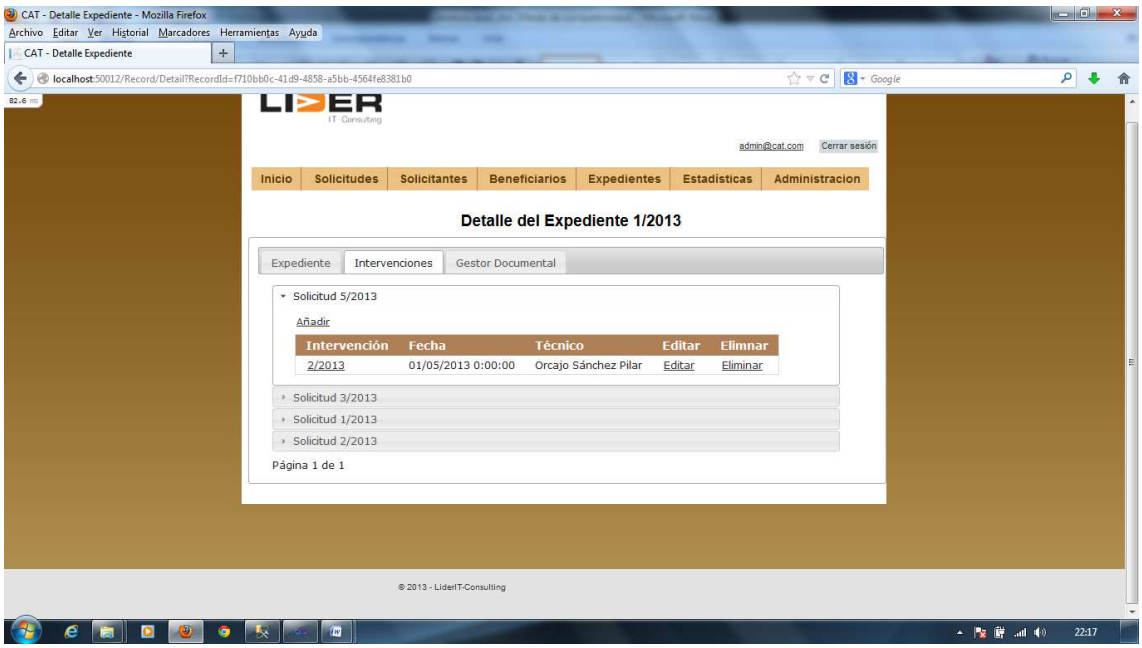


Figura 31. Pantalla detalles expediente (II)

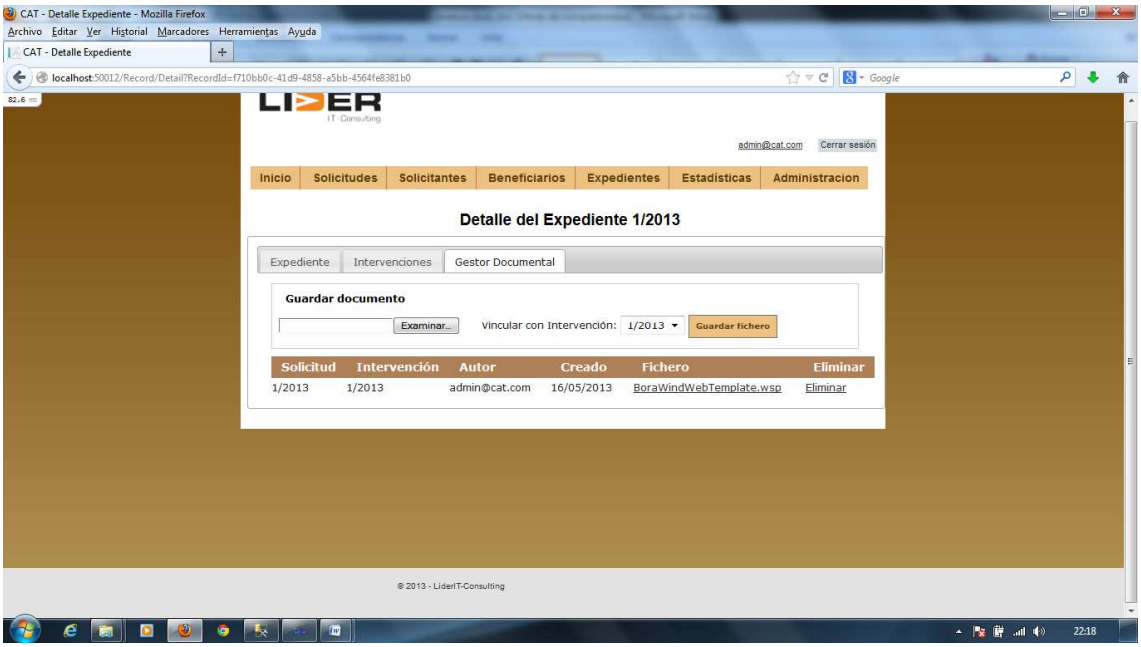


Figura 32. Pantalla detalles de expedientes (III)

5.2.17 Nueva intervención

A esta pantalla se accede a través de los detalles del expediente y permite introducir los datos necesarios para crear una nueva intervención en el expediente.

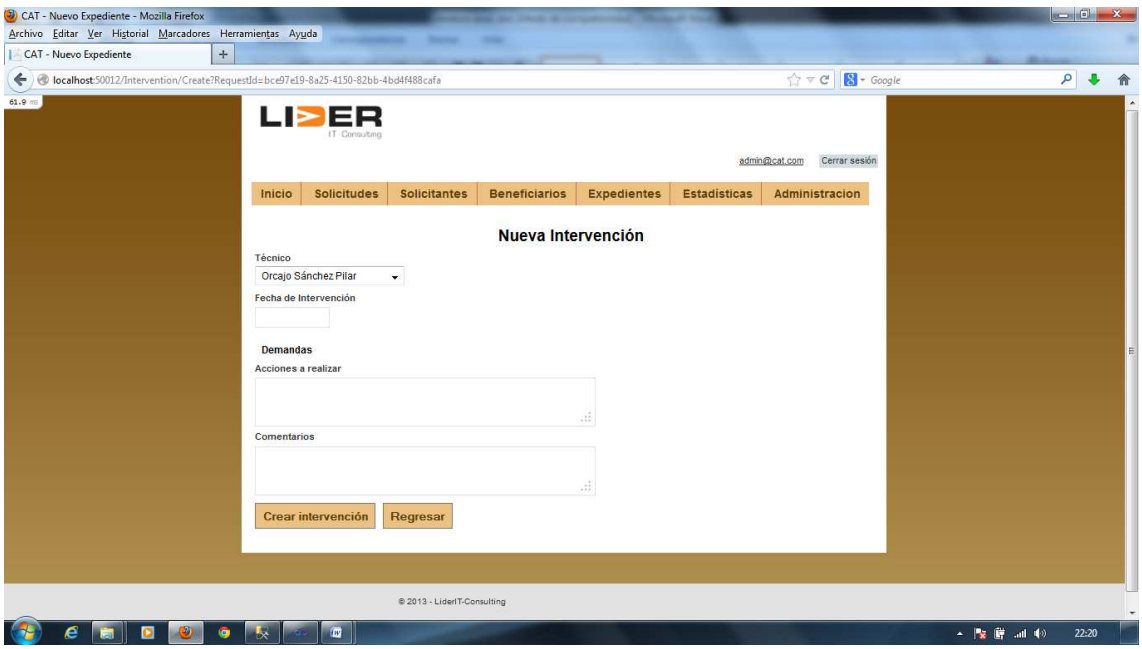


Figura 33. Pantalla nueva intervención

5.2.18 Detalles de la intervención

A esta pantalla se accede a través de los detalles del expediente y permite consultar los detalles de una intervención concreta.

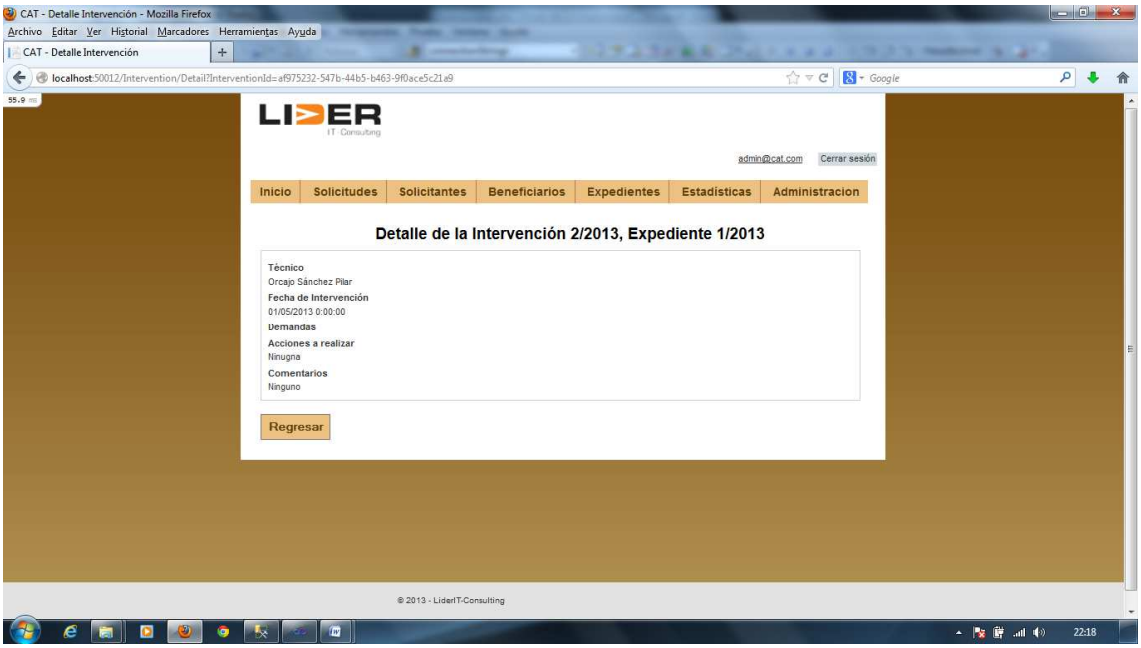


Figura 34. Pantalla detalles de la intervención

5.2.19 Administración

Mediante esta pantalla se pueden administrar las opciones a mostrar en los formularios.

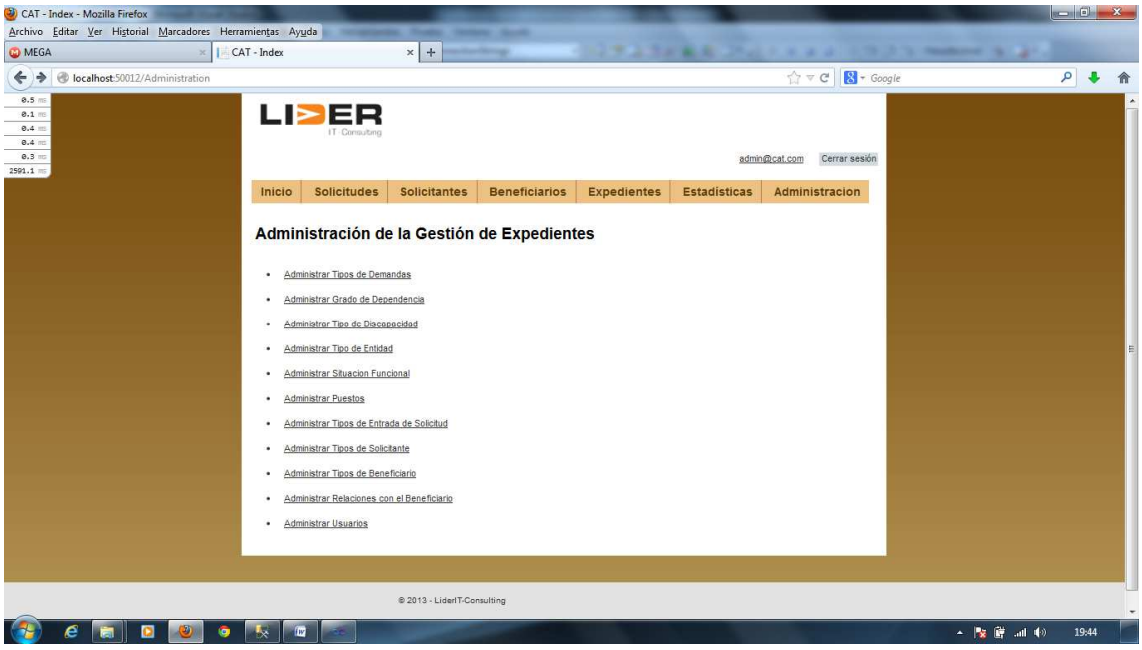


Figura 35. Pantalla administración

5.2.20 Administrar Tipos de Demandas

A través de esta pantalla se pueden crear/modificar/eliminar tipos de demandas.

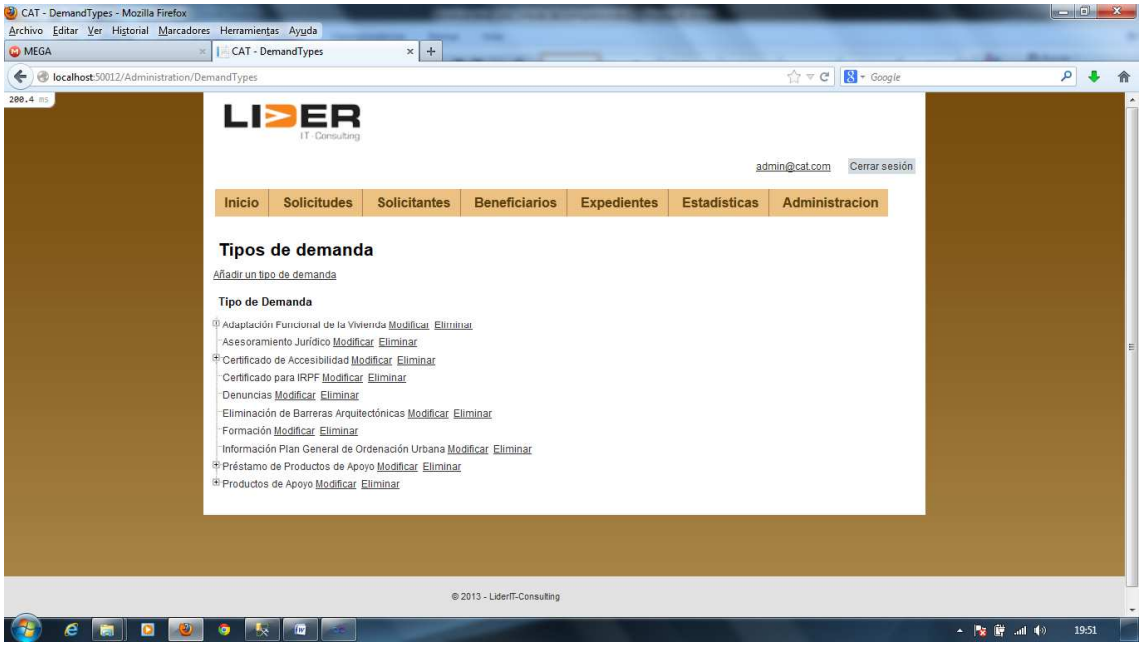


Figura 36. Pantalla administrar tipos de demanda

5.2.21 Añadir Tipos de Demandas

A través de esta pantalla se añade el texto del nuevo tipo de demanda a agregar.

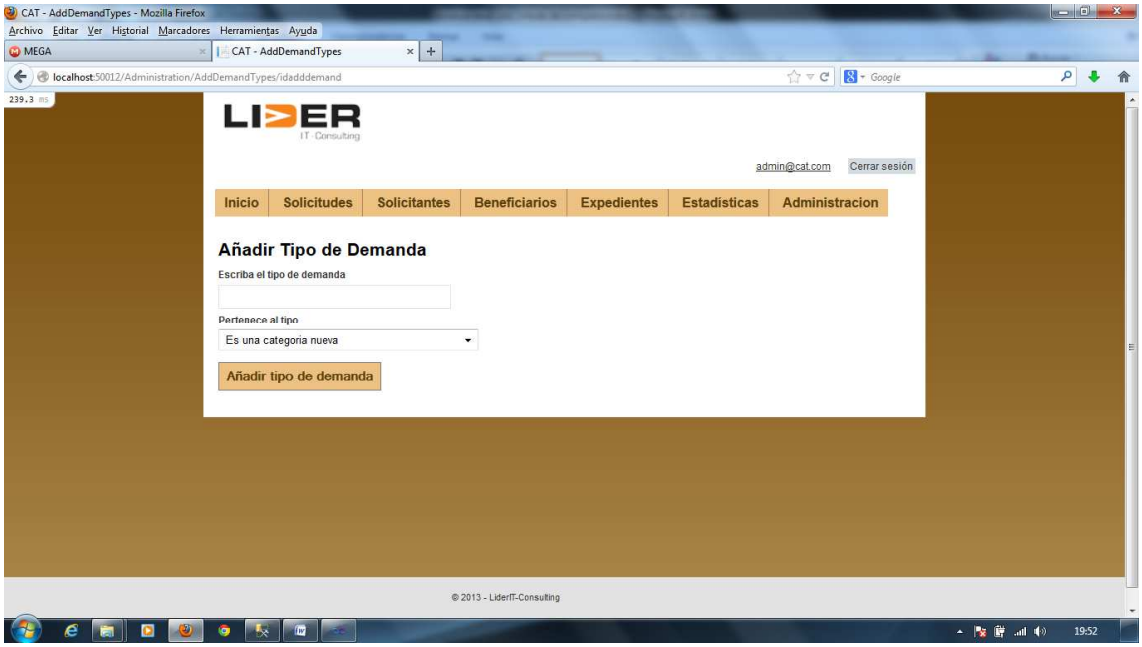


Figura 37. Pantalla añadir tipos de demandas

5.2.22 Administrar Grado de Dependencia

A través de esta pantalla se pueden crear/modificar/eliminar grados de dependencia.

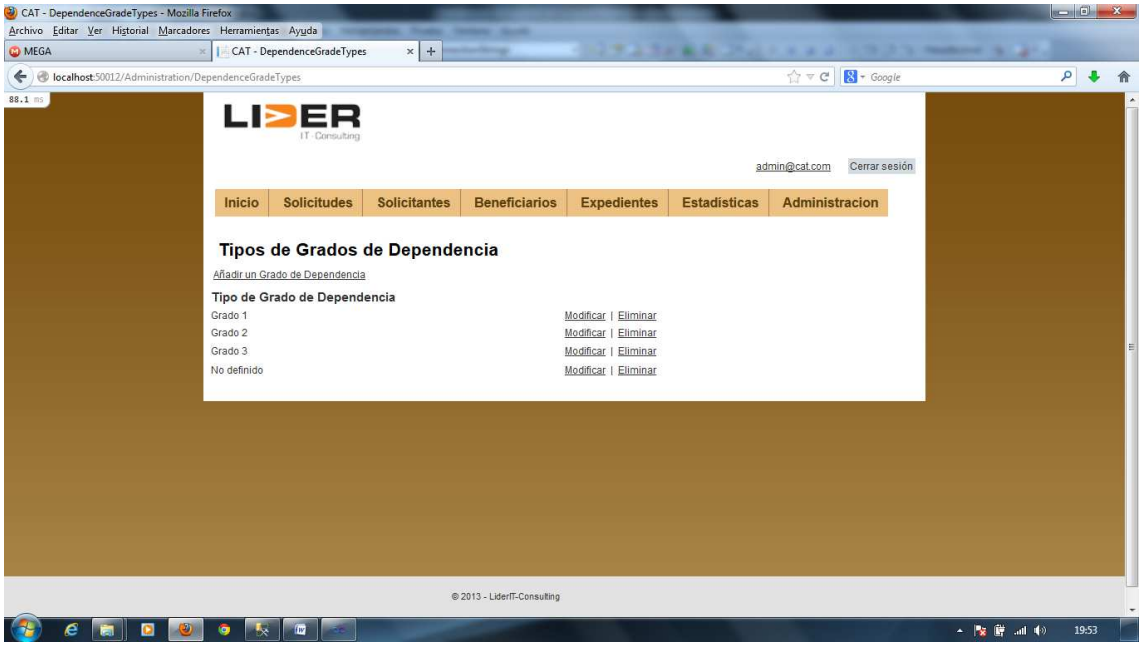


Figura 38. Pantalla administrar grado de dependencia

5.2.23 Añadir Grado de Dependencia

A través de esta pantalla se añade el texto del nuevo grado de dependencia a agregar.

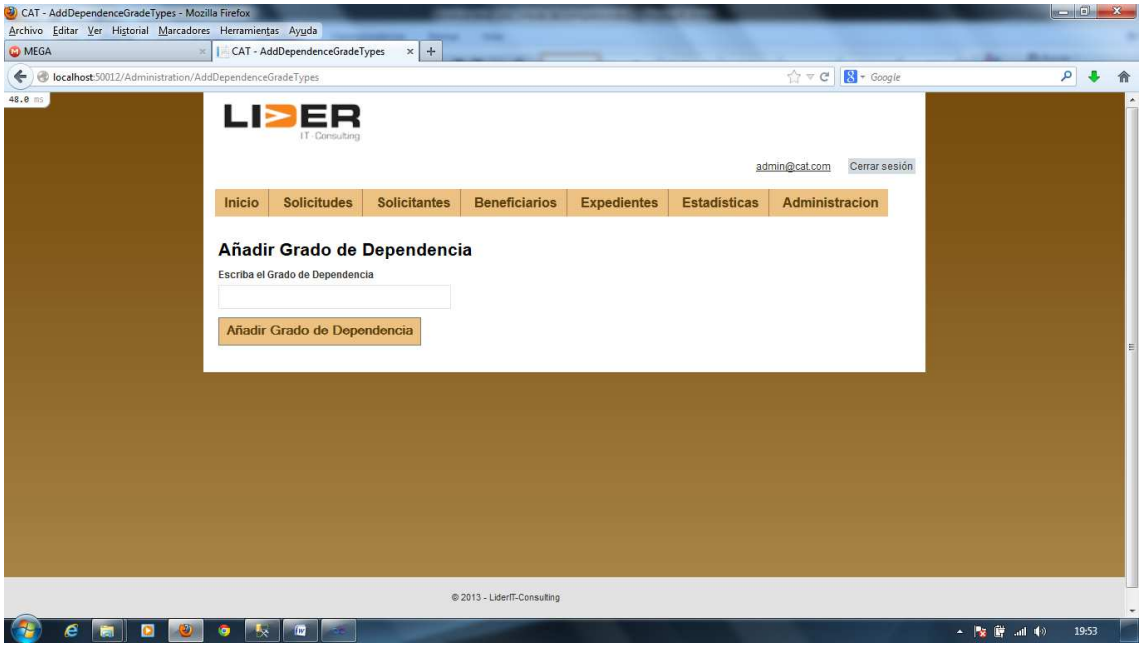


Figura 39. Pantalla añadir grado de dependencia

5.2.24 Administrar Tipo de Discapacidad

A través de esta pantalla se pueden crear/modificar/eliminar tipos de discapacidad.

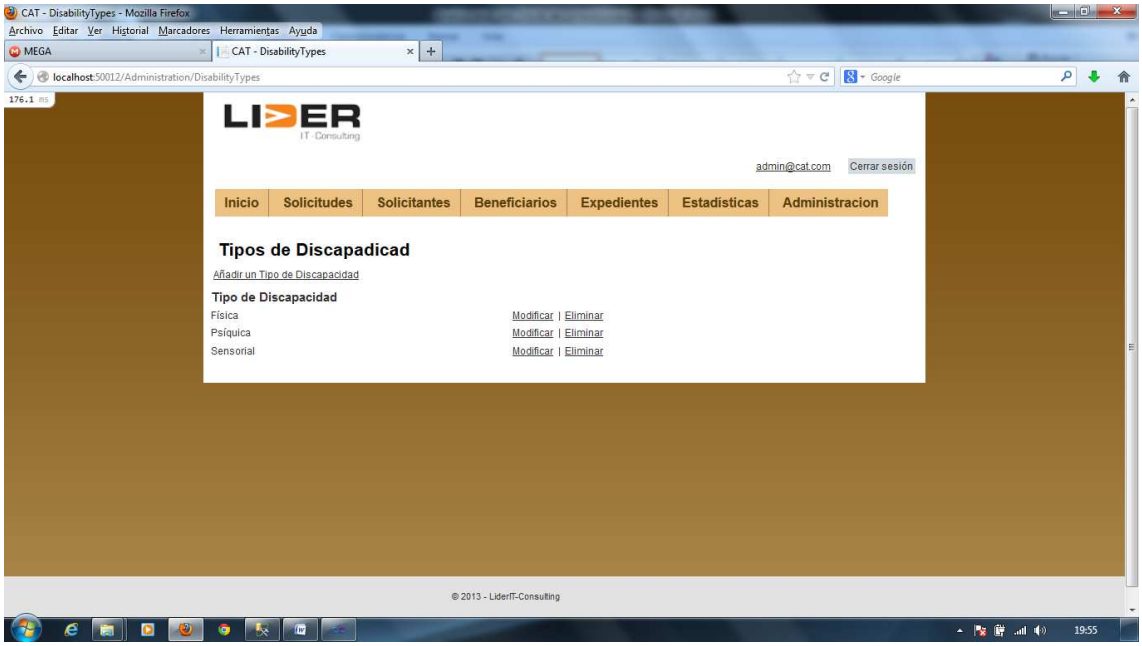


Figura 40. Pantalla administrar tipo de discapacidad

5.2.25 Añadir Tipo de Discapacidad

A través de esta pantalla se añade el texto del nuevo tipo de dependencia a agregar.

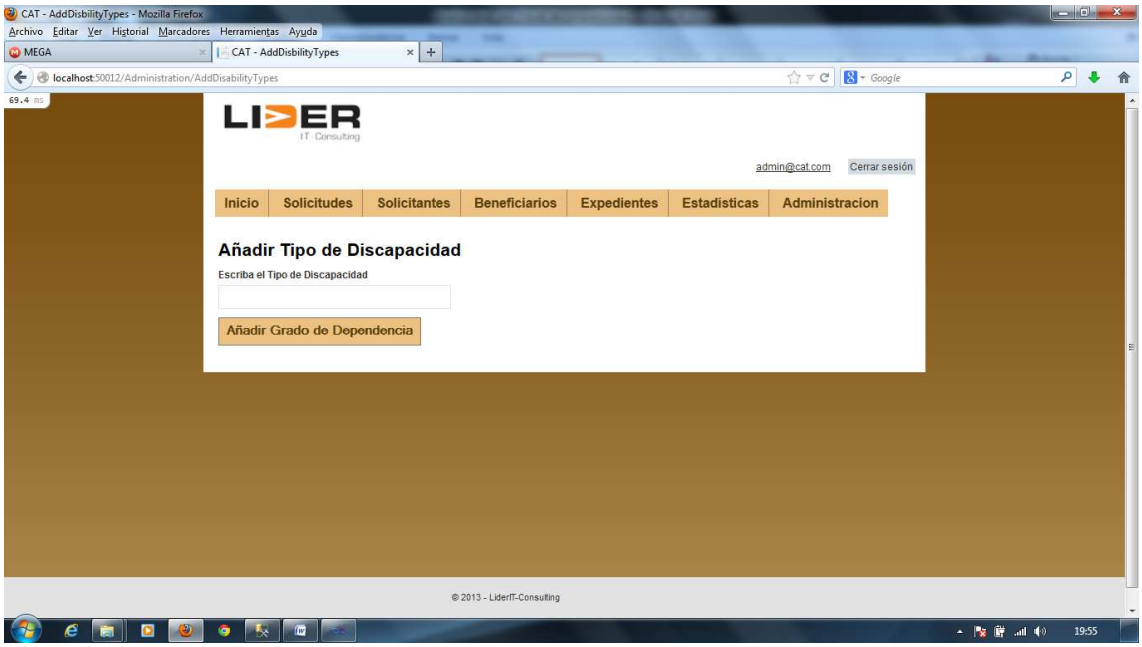


Figura 41. Pantalla administrar tipo discapacidad

5.2.26 Administrar Tipo de Entidad

A través de esta pantalla se pueden crear/modificar/eliminar tipos de entidad.

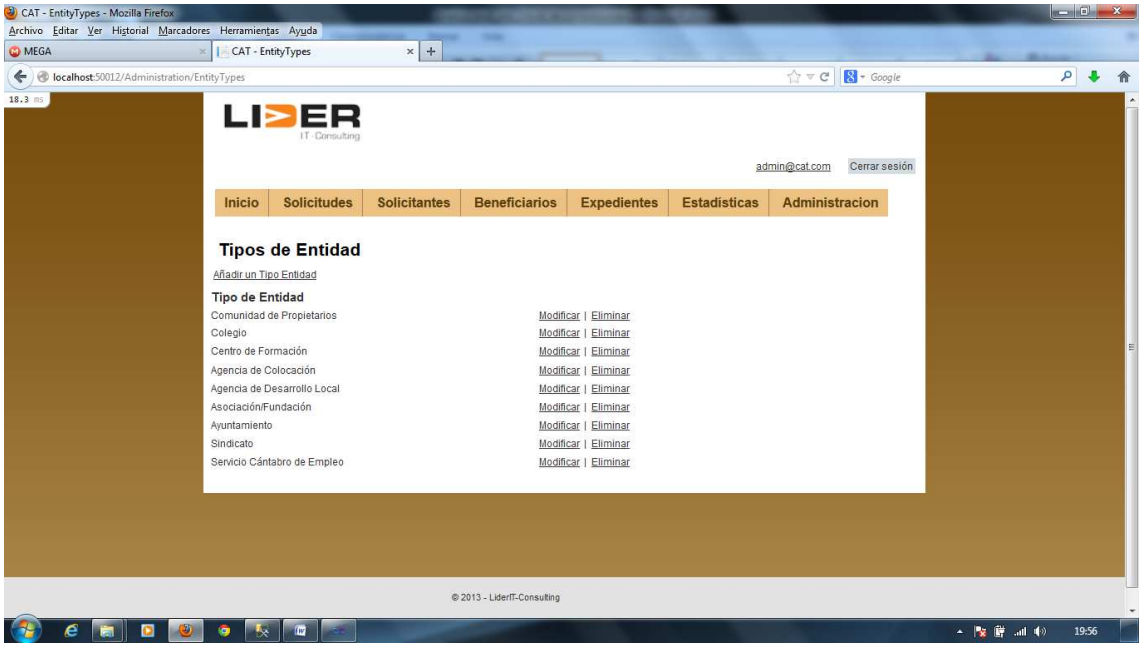


Figura 42. Pantalla administrar tipos de entidad

5.2.27 Añadir Tipo de Entidad

A través de esta pantalla se añade el texto del nuevo tipo de entidad a agregar.

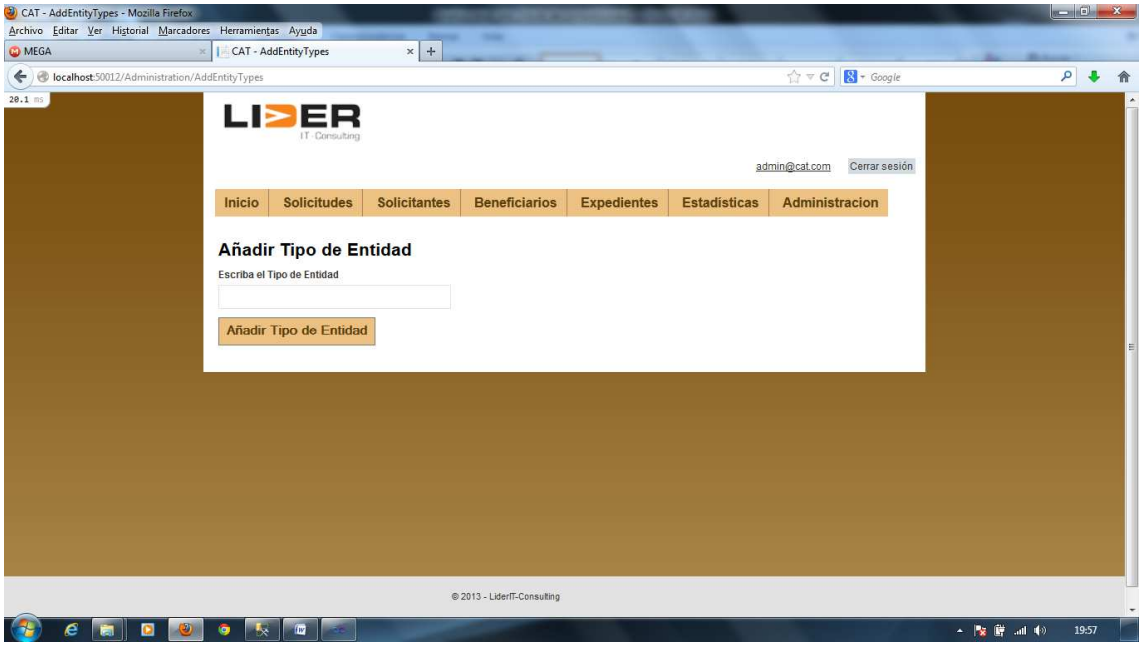


Figura 43. Pantalla añadir tipo de entidad

5.2.28 Administrar Situación Funcional

A través de esta pantalla se pueden crear/modificar/eliminar situaciones funcionales.

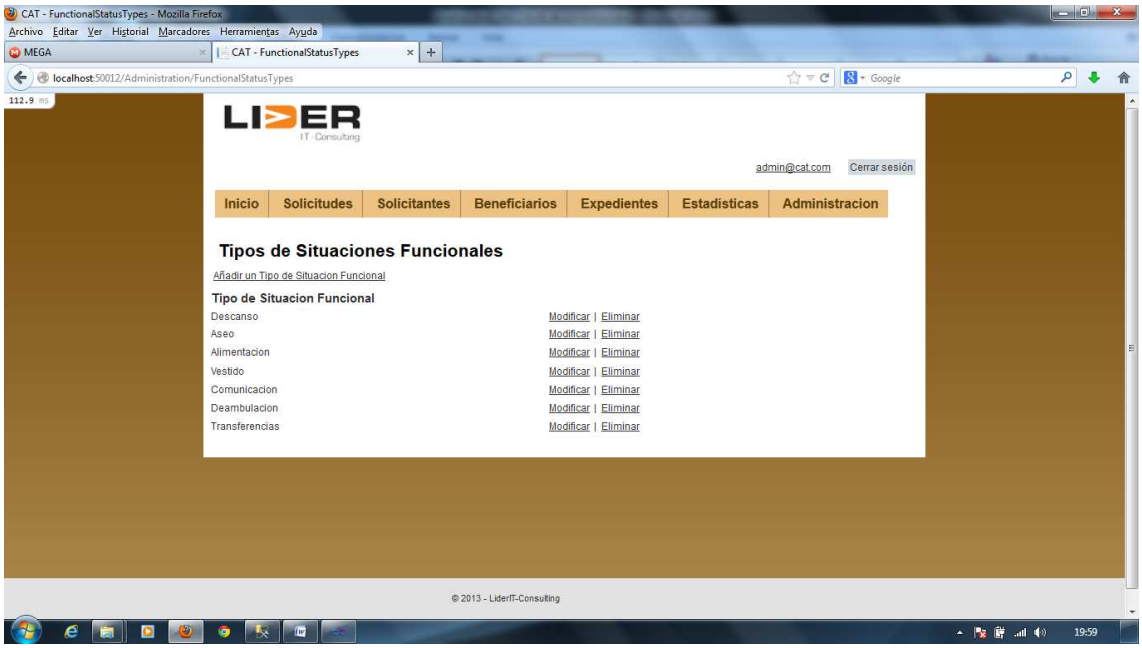


Figura 44. Pantalla administrar tipos de situaciones funcionales

5.2.29 Añadir Situación Funcional

A través de esta pantalla se añade el texto del nueva situación funcional a agregar.

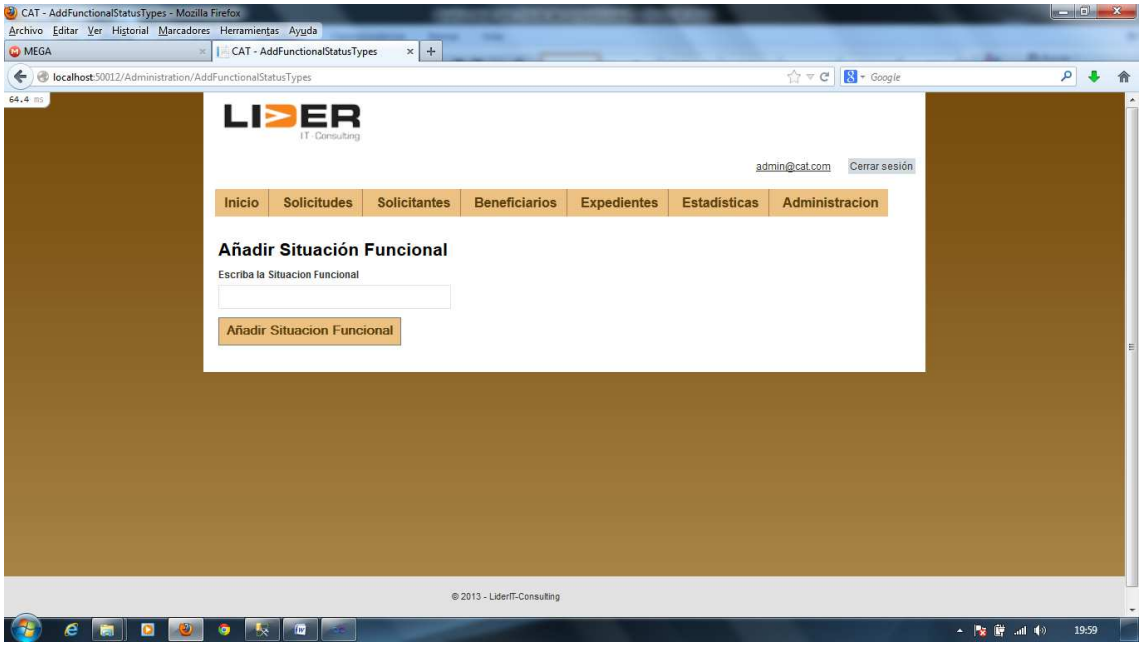


Figura 45. Pantalla añadir situación funcional

5.2.30 Administrar Puestos

A través de esta pantalla se pueden crear/modificar/eliminar puestos.

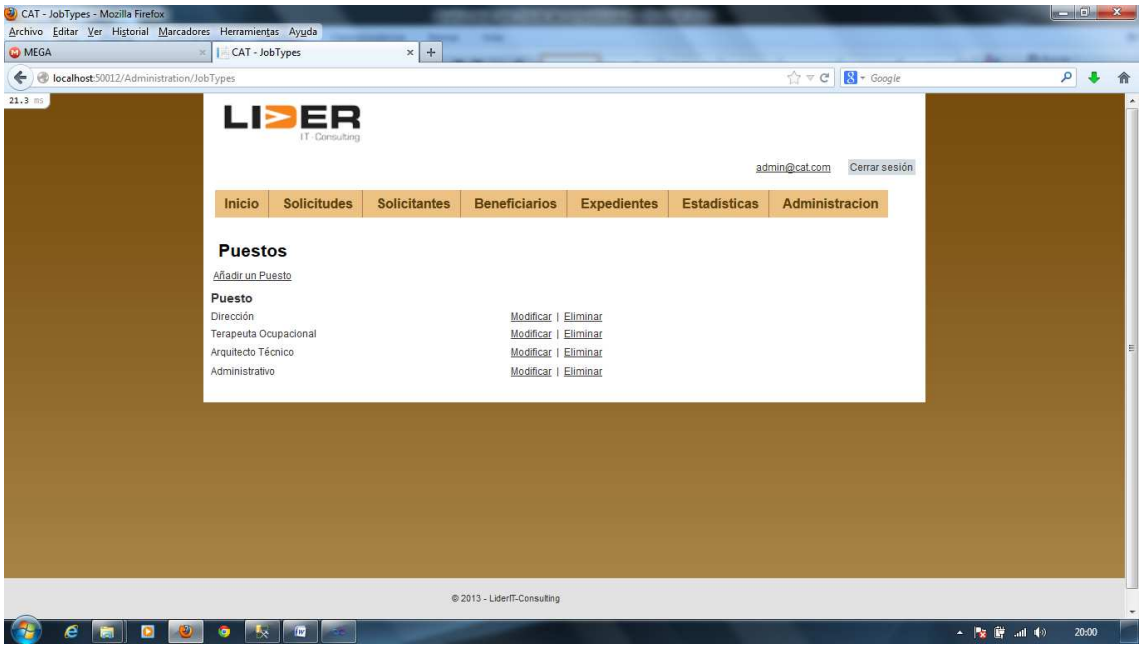


Figura 46. Pantalla administrar puestos

5.2.31 Añadir Puestos

A través de esta pantalla se añade el texto del nuevo puesto a agregar.

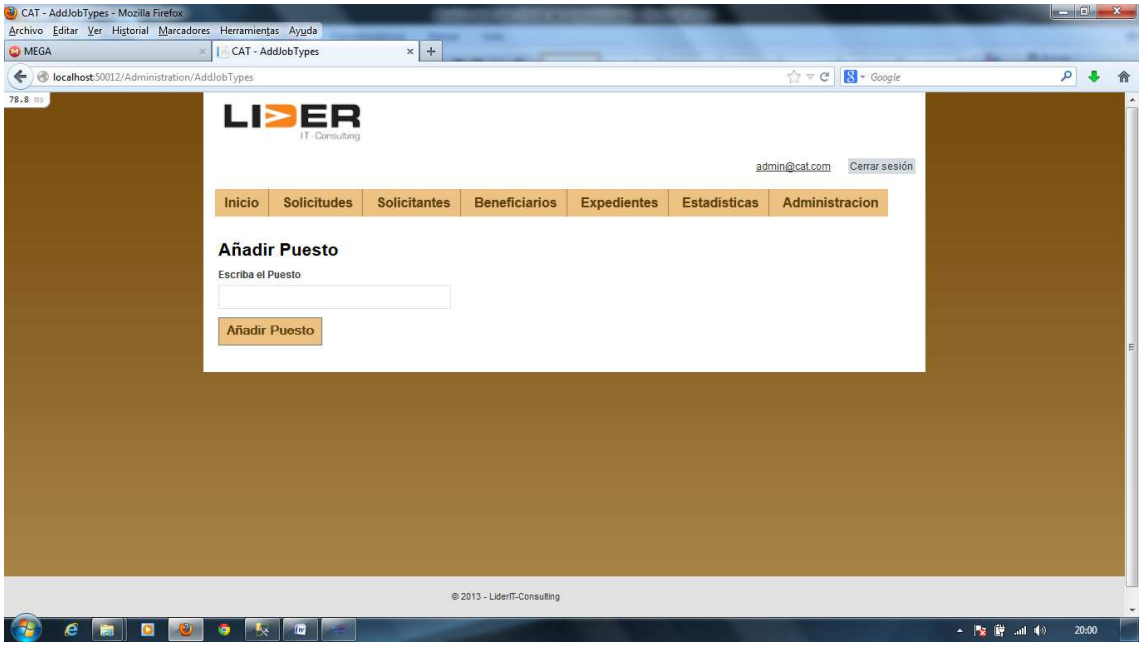


Figura 47. Pantalla añadir puesto

5.2.32 Administrar Tipo de Entrada de Solicitud

A través de esta pantalla se pueden crear/modificar/eliminar tipos de entrada de solicitud.

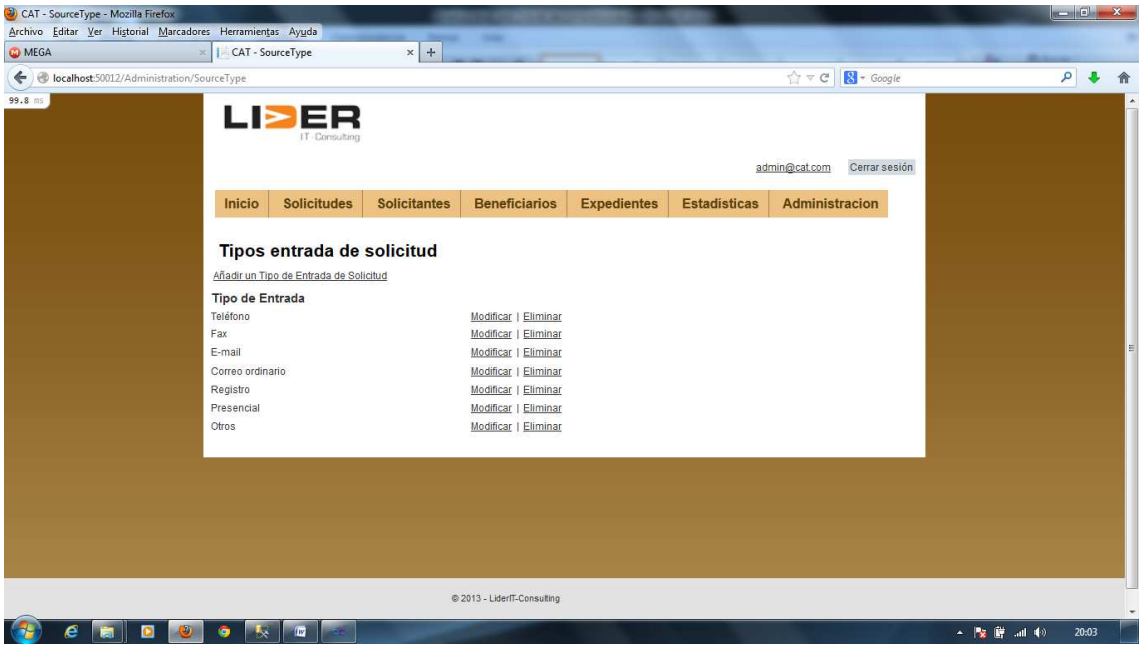


Figura 48. Pantalla administrar tipo de entrada de solicitud

5.2.33 Añadir Tipo de Entrada de Solicitud

A través de esta pantalla se añade el texto del nuevo tipo de entrada de solicitud a agregar.

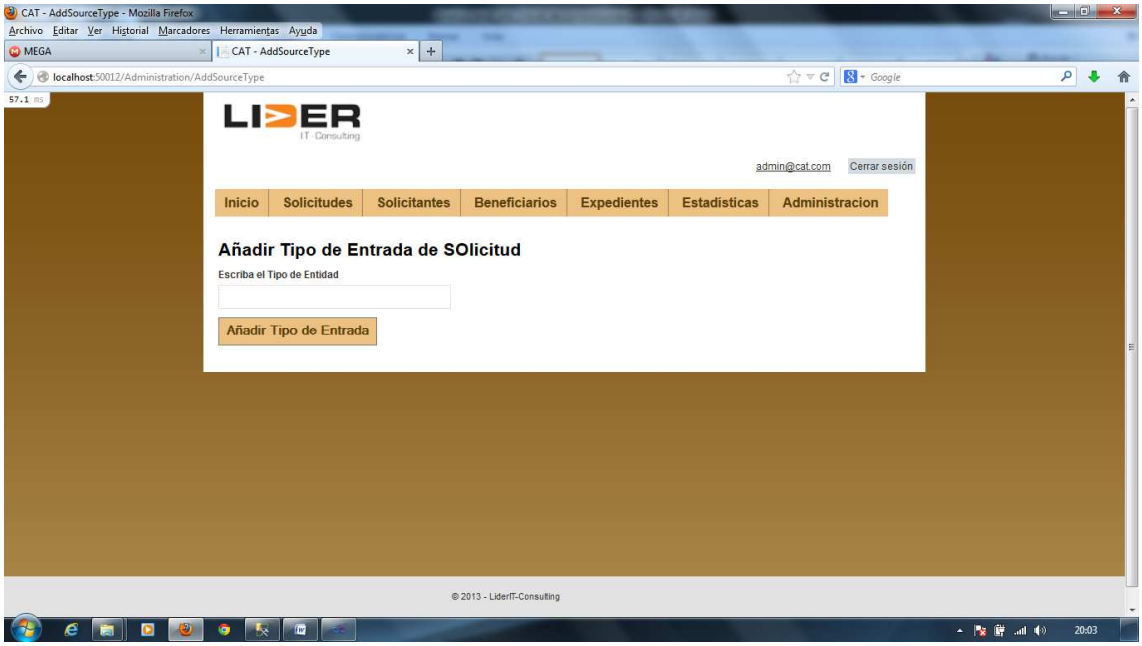


Figura 49. Pantalla añadir tipo de entrada de solicitud

5.2.34 Administrar Tipo de Solicitante

A través de esta pantalla se pueden crear/modificar/eliminar tipos de solicitante.

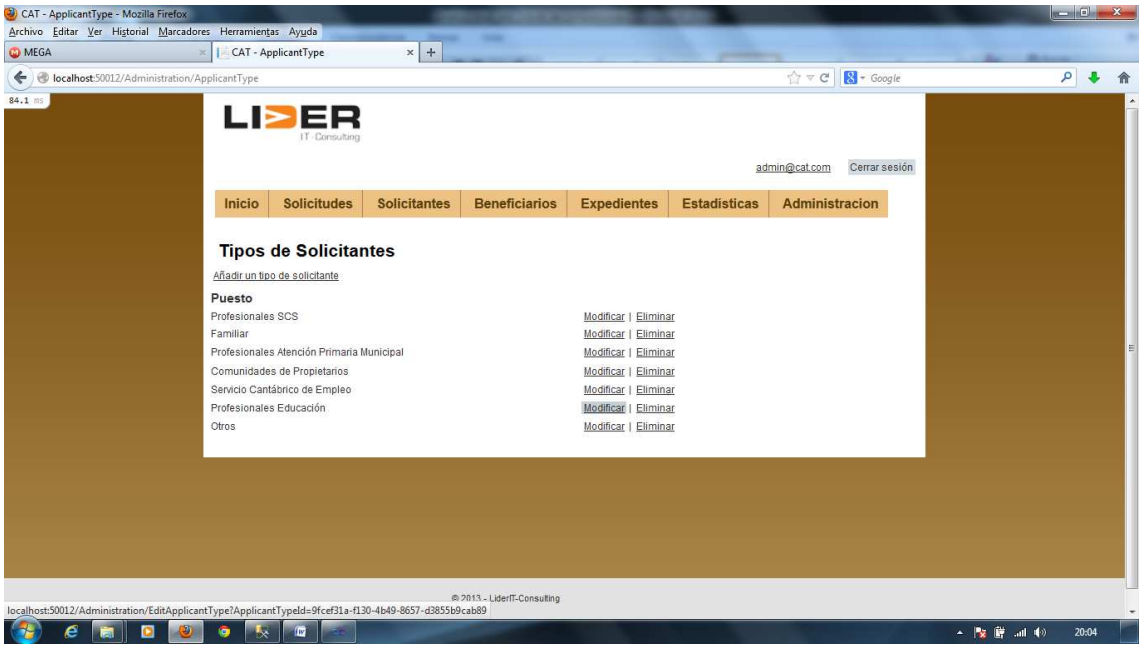


Figura 50. Pantalla tipo de solicitantes

5.2.35 Añadir Tipo de Solicitante

A través de esta pantalla se añade el texto del nuevo tipo de solicitante a agregar.

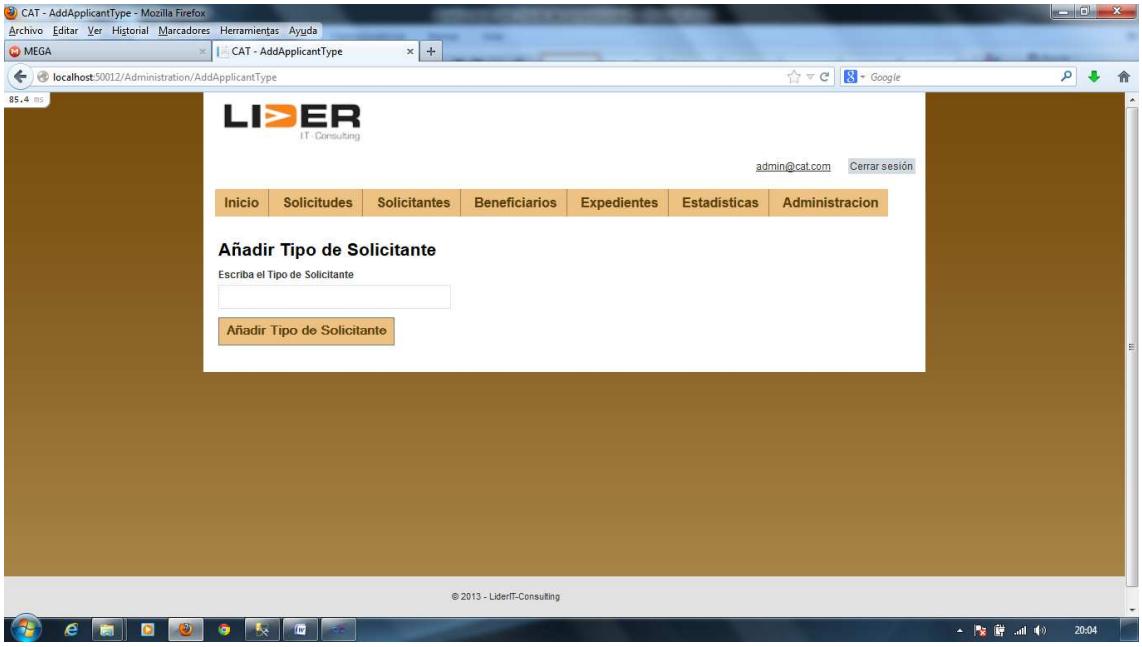


Figura 51. Pantalla añadir tipo de solicitantes

5.2.36 Administrar Tipo de Beneficiario

A través de esta pantalla se pueden crear/modificar/eliminar tipos de beneficiario.

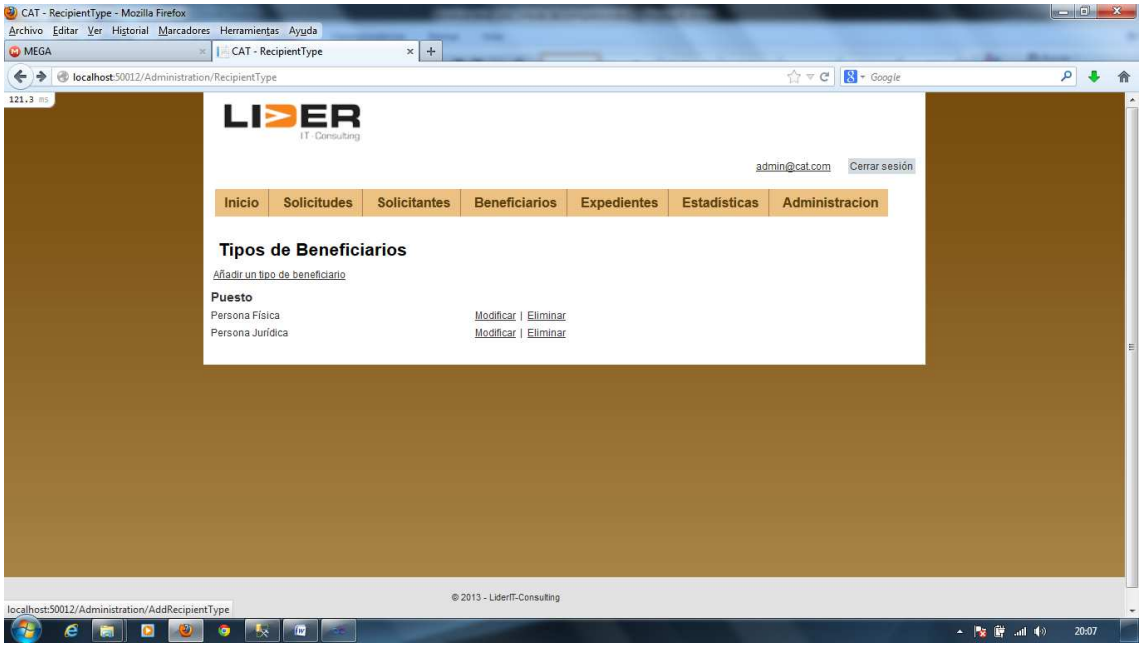


Figura 52. Pantalla tipo de beneficiarios

5.2.37 Añadir Tipo de Beneficiario

A través de esta pantalla se añade el texto del nuevo tipo de beneficiario a agregar.

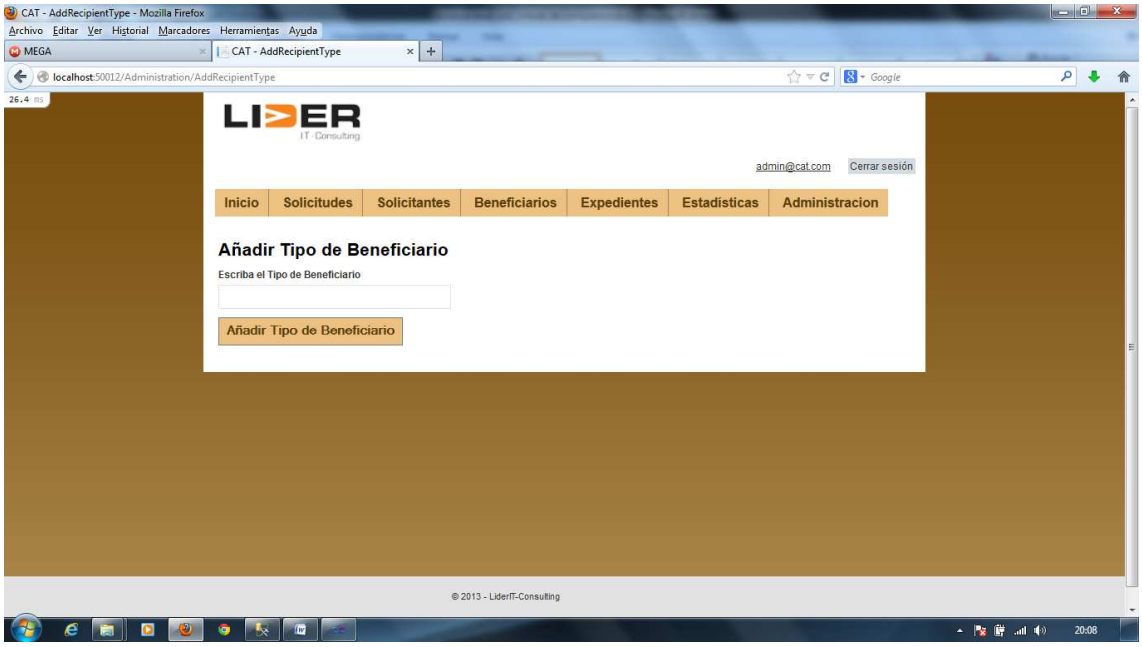


Figura 53. Pantalla añadir tipo de beneficiario

5.2.38 Administrar Relaciones con el Beneficiario

A través de esta pantalla se pueden crear/modificar/eliminar relaciones con el beneficiario.

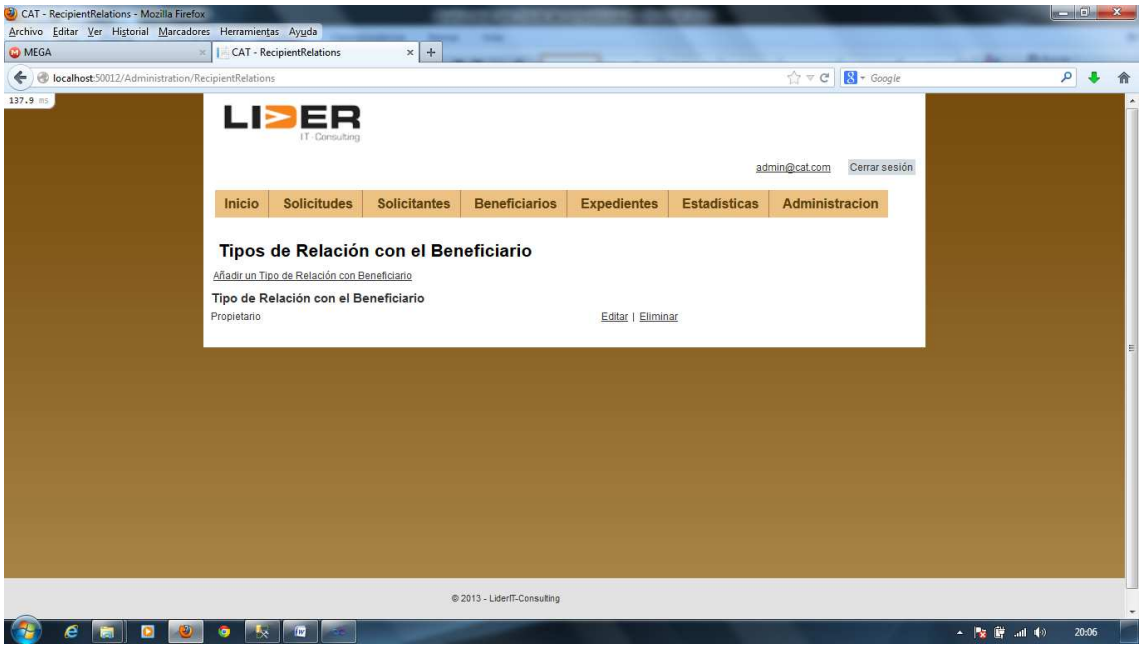


Figura 54. Pantalla tipo de relación con el beneficiario

5.2.39 Añadir Relación con el Beneficiario

A través de esta pantalla se añade el texto del nueva relación con el beneficiario a agregar.

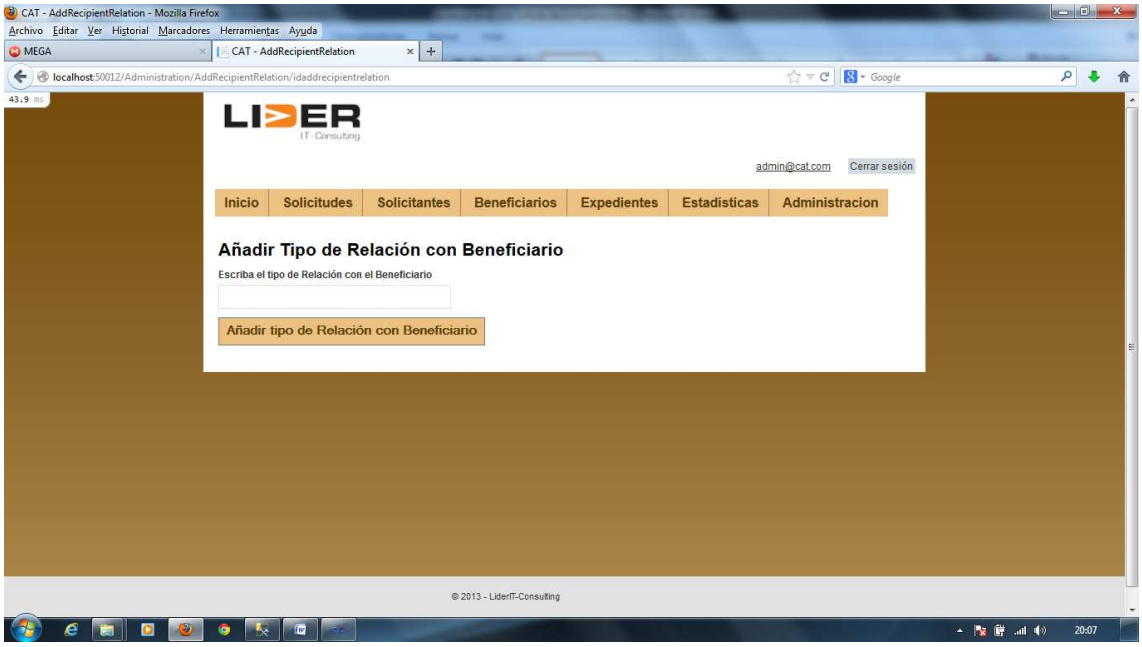


Figura 55. Pantalla añadir tipo de relación con el beneficiario

5.2.40 Administrar Usuarios

A través de esta pantalla se pueden crear/eliminar usuarios.

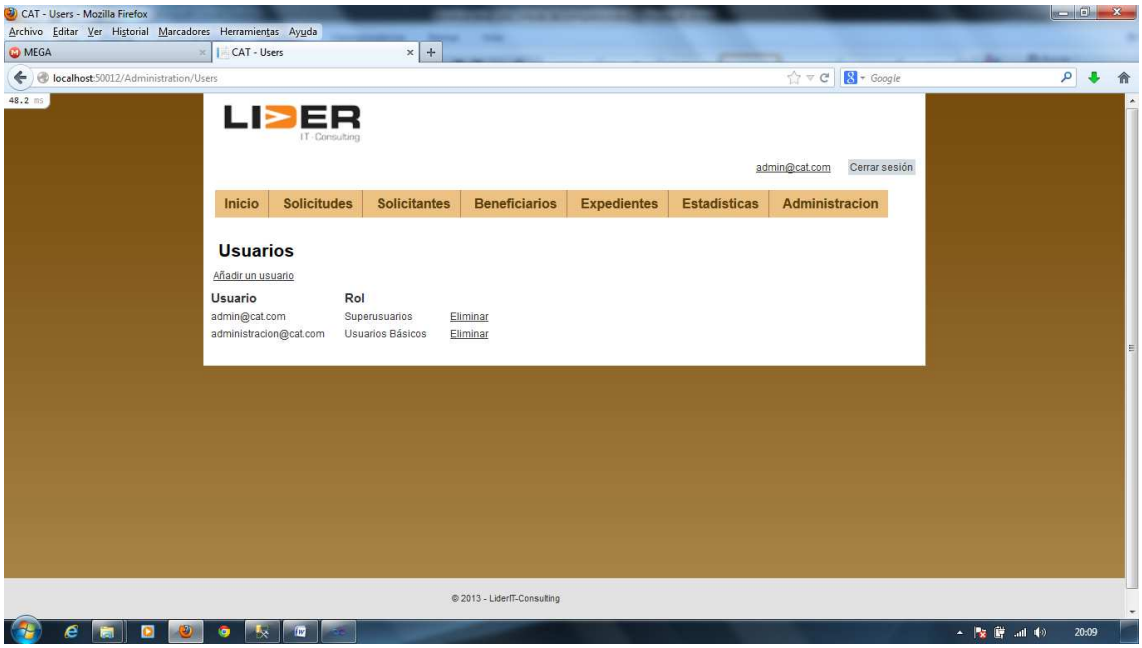


Figura 56. Pantalla administrar usuarios

5.2.41 Añadir Usuario

A través de esta pantalla se añade el nuevo usuario.

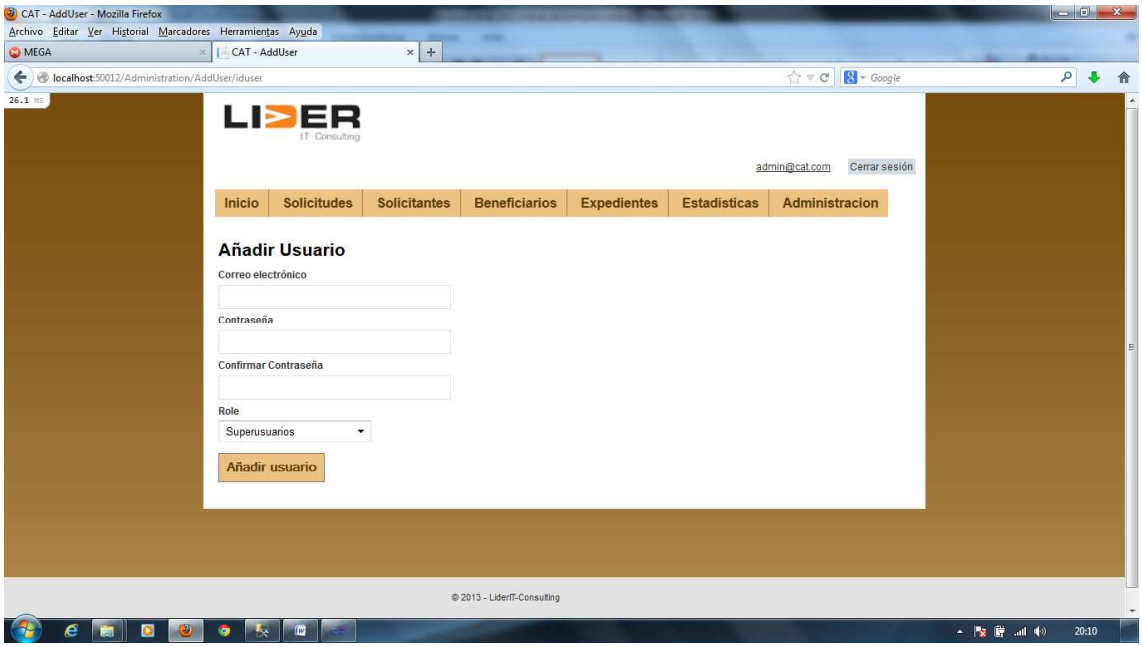
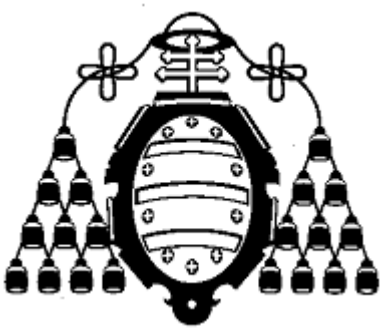


Figura 57. Pantalla añadir usuario



UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

HERRAMIENTA DE GESTIÓN DE EXPEDIENTES PARA EL CAT

DOCUMENTO Nº IV

DISEÑO

ZEUS PÉREZ GARCÍA
JUNIO 2012

ÁREA DE TELEMÁTICA
TUTOR: XICU XABIEL GARCÍA PAÑEDA

ÍNDICE DEL DISEÑO

ÍNDICE DEL DISEÑO

ÍNDICE DE FÍGURAS DEL DISEÑO

DISEÑO

1. Diseño de la arquitectura del sistema

1.1. División en capas del sistema

En nuestra aplicación podemos distinguir tres capas o niveles:

- *Capa de Interfaz*: En esta capa se encuentra el subsistema de la Vista, ya que es en el que se implementan las interfaces con las que interactuará el usuario.
- *Capa de Aplicación*: Ésta es la capa más compleja, y en este nivel se encuentran dos subsistemas de diseño: 'Modelo', 'Controlador', que se encargan del manejo de la información introducida por el usuario en la capa superior, reconstruyéndola y distribuyéndola de manera eficiente, para que por un lado se pueda mostrar al usuario en la capa de Interfaz y por otro lado realice accesos correctos a la base de datos a través de la capa de Datos.
- *Capa de Datos:* Al igual que la capa de nivel superior, ésta únicamente posee un subsistema, 'Acceso a la base de datos' que se encargará de accesos a la base de datos.

- *Capa de Seguridad*: Esta capa, a diferencia de las anteriores, es vertical en lugar de horizontal, es decir, es parte de varios subsistemas. Primeramente necesita una vista para traducir los datos, y posteriormente cotejarlos con la información almacenada en la base de datos a través del subsistema de Acceso a la Base de Datos. Además en el subsistema Controlador se definirá que roles permiten realizar ciertas acciones. Esta capa se encarga principalmente del control de acceso de los usuarios a la aplicación y a las funciones de la misma.

1.2. División en subsistemas de diseño

La división en subsistemas está condicionada a la arquitectura de programación utilizada. En este caso en concreto se ha utilizado una aplicación Modelo Vista-Controlador, por lo tanto los subsistemas coinciden en parte con los elementos de dicha arquitectura.

- *Subsistema 'Vista'*: Contiene las pantallas con las que va a interaccionar el usuario del programa.

- *Subsistema 'Controlador'*: En este módulo se hace referencia a todas las acciones que realiza la aplicación, sirviendo de nexo entre la vista y el acceso a la base de datos.
- *Subsistema 'Modelo'*: En este módulo se encuentran las estructuras de datos de las entidades concernientes a la aplicación.
- *Subsistema 'Validación'*: Aquí se dispondrá de las funcionalidades necesarias para realizar las comprobaciones pertinentes de la aplicación y así verificar un correcto comportamiento y uso de la misma. Este subsistema esta integrado a su vez en otros subsistemas. Existe una validación en la base de datos, que solo permitirá almacenar los datos tal y como están definidos al crear la base de datos y que seria del lado del servidor. Por otro lado en ciertos campos es necesario crear un modelo para dicha

validación. Y por último el controlador también realiza sus propias validaciones siendo estás del lado del cliente.

- *Subsistema de 'Acceso a la Base de Datos'*: Este módulo es el único de esta capa (la capa inferior del sistema) que es la encargada de realizar las lecturas y escrituras de la base de datos. Su objetivo es el de preservar la consistencia de la base de datos evitando problemas de 'uso incorrecto' por parte del usuario o de otras clases, o una modificación provisional (causada por el manejo de métodos de otros subsistemas).

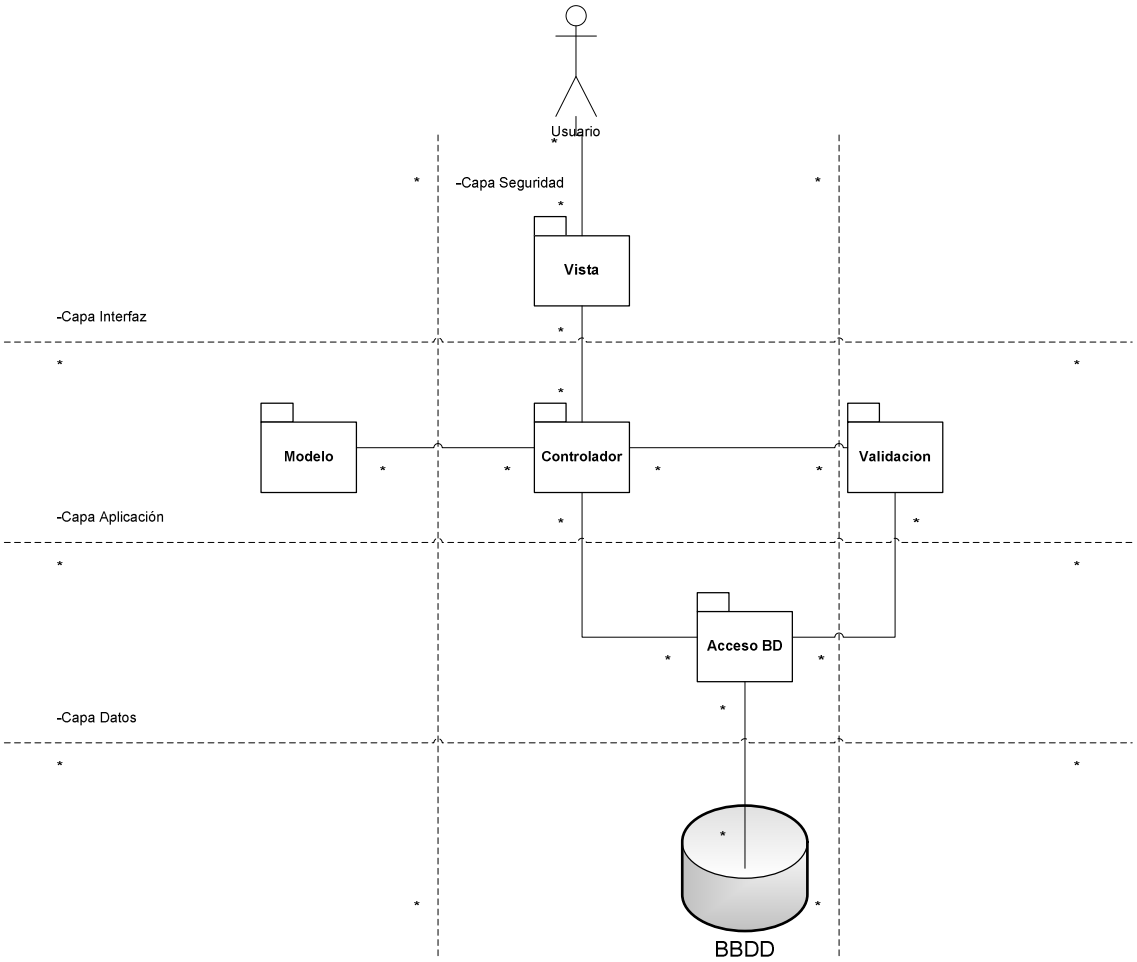


Figura 1. Diseño en capas

2. Base de datos

2.1 Diagrama

Para el correcto funcionamiento de la aplicación es necesario que esté conectada con una base de datos que contenga las tablas y las relaciones correctas para guardar toda la información manejada por la aplicación.

A continuación se muestra el diagrama de la base de datos con las entidades principales y las secundarias:

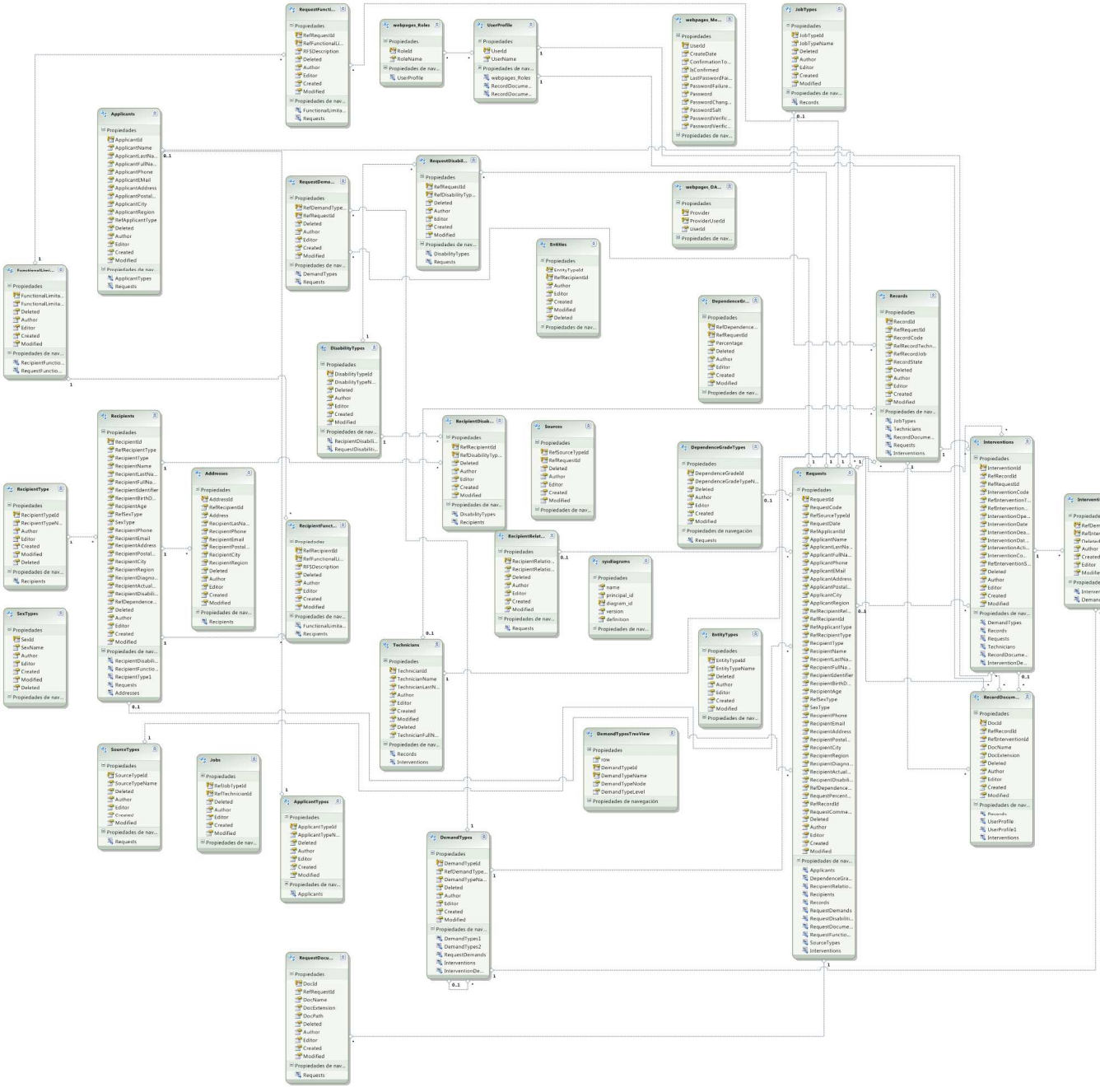


Figura 2. Diagrama de la base de datos

Para poder apreciar con mayor claridad las relaciones entre las entidades principales a continuación se muestra el diagrama de la base de datos incluyendo solo dichas entidades:

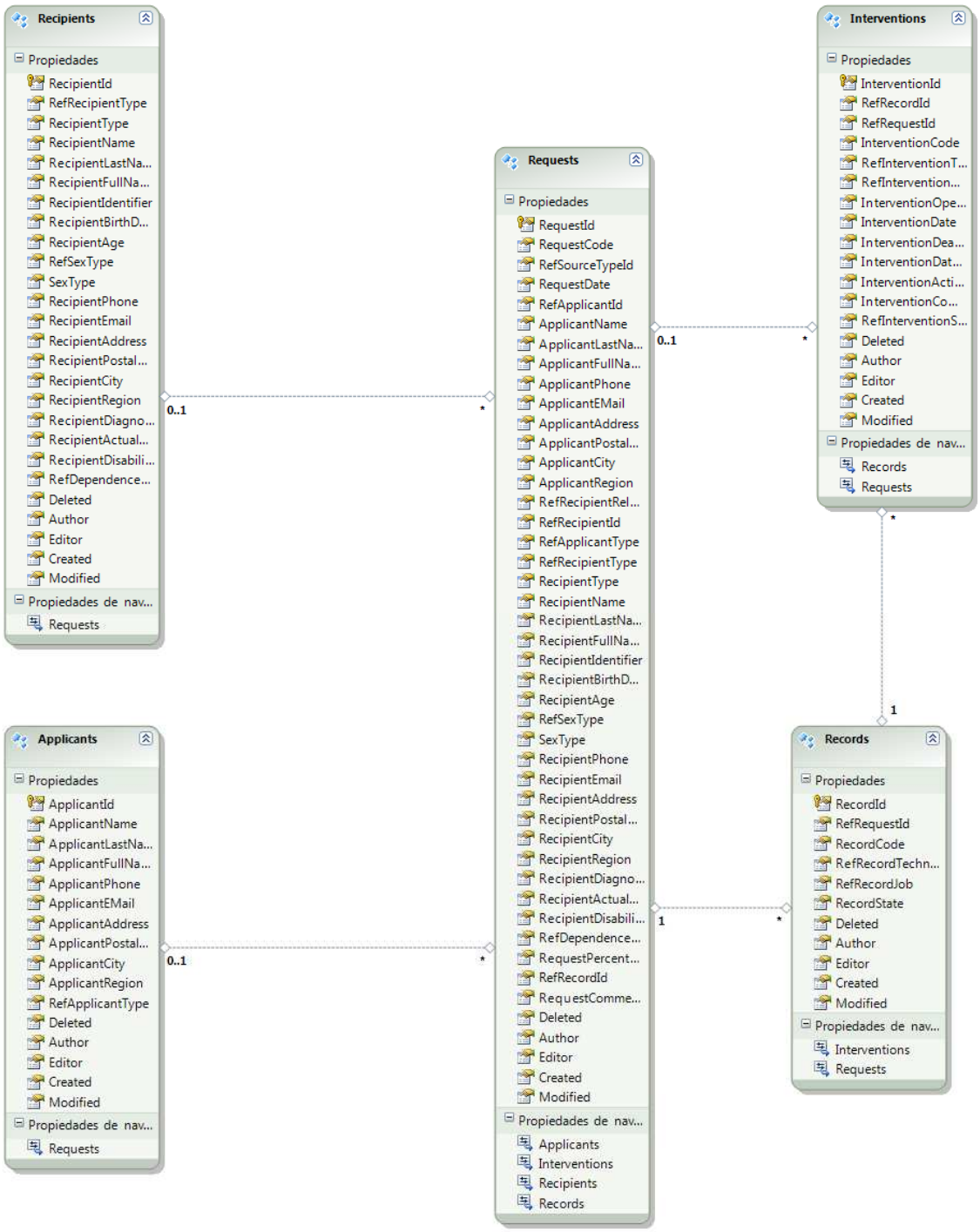


Figura 3. Diagrama base de datos (II)

2. 2 Tablas

A continuación se muestran en detalle todas las tablas de la base de datos, enumerando los elementos que las componen y el tipo de cada uno de ellos.

2.2.1 Entidades principales

A continuación se muestran las tablas correspondientes a las entidades principales de la aplicación:

La ¡Error! No se encuentra el origen de la referencia. **Applicants** presenta los campos de la lista de Solicitantes.

Campo	Descripción	Tipo
ApplicantId	Identificador del Solicitante.	Uniqueidentifier
ApplicantName	Nombre del Solicitante.	Texto(255)
ApplicantLastName	Apellidos del Solicitante.	Texto (255)
ApplicantFullName	Nombre y Apellidos del Solicitante.	Calculado
ApplicantPhone	Teléfono del Solicitante.	Texto (15)
ApplicantEmail	Email del Solicitante.	Texto (255)
ApplicantAdress	Dirección del Solicitante.	Texto (255)
ApplicantPostalCode	Código Postal del Solicitante.	Texto(5)
ApplicantCity	Población del Solicitante.	Texto(255)
ApplicantRegion	Provincia del Solicitante.	Texto(255)
RefApplicantType	Referencia al Tipo de Solicitante.	Uniqueidentifier
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **Recipients** presenta los campos de la lista de Beneficiarios.

Campo	Descripción	Tipo
RecipientId	Identificador del Beneficiario.	Uniqueidentifier
RefRecipientType	Referencia al Tipo de Beneficiario.	Uniqueidentifier
RecipientType	Tipo de Beneficiario.	Int
RecipientName	Nombre del Beneficiario.	Texto(255)
RecipientLastName	Apellidos del Beneficiario.	Texto (255)
RecipientFullName	Nombre y Apellidos del Beneficiario.	Calculado
RecipientIdentifier	DNI/CIF del Beneficiario.	Texto (10)
RecipientBirthDate	Fecha de nacimiento del Beneficiario.	DateTime
RecipientAge	Edad del Beneficiario.	Calculado
RefSexType	Referencia al Género del Beneficiario.	Uniqueidentifier
SexType	Género del Beneficiario.	Int
RecipientPhone	Teléfono del Beneficiario.	Texto (15)
RecipientEmail	Email del Beneficiario.	Texto (255)
RecipientAdress	Dirección del Beneficiario.	Texto (255)
RecipientPostalCode	Código Postal del Beneficiario.	Texto(5)
RecipientCity	Población del Beneficiario.	Texto(255)
RecipientRegion	Provincia del Beneficiario.	Texto(255)
RecipientDiagnostic	Diagnóstico del Beneficiario.	Texto (4000)
RecipientActualSituation	Situación Actual del Beneficiario.	Texto (4000)
RecipientDisabilityDescription	Descripción de la Discapacidad del Beneficiario.	Texto (4000)
RefDependenceGradeID	Referencia al Grado de Dependencia del Beneficiario.	Uniqueidentifier
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **Requests** presenta los campos de la lista de Solicitudes.

Campo	Descripción	Tipo
RequestId	Identificador de la Solicitud.	Uniqueidentifier
RequestCode	Código de la Solicitud.	Texto(15)
RefSourceTypeId	Referencia al Tipo de Entrada de la Solicitud.	Uniqueidentifier
RequestDate	Fecha de la Solicitud.	DateTime
RefApplicantId	Referencia al Identificador del Solicitante.	Uniqueidentifier
ApplicantName	Nombre del Solicitante.	Texto(255)
ApplicantLastName	Apellidos del Solicitante.	Texto (255)
ApplicantFullName	Nombre y Apellidos del Solicitante.	Calculado
ApplicantPhone	Teléfono del Solicitante.	Texto (15)
ApplicantEmail	Email del Solicitante.	Texto (255)
ApplicantAdress	Dirección del Solicitante.	Texto (255)
ApplicantPostalCode	Código Postal del Solicitante.	Texto(5)
ApplicantCity	Población del Solicitante.	Texto(255)
ApplicantRegion	Provincia del Solicitante.	Texto(255)
RefRecipientType	Referencia al Tipo de Beneficiario.	Uniqueidentifier
RefRecipientType	Referencia al Tipo de Beneficiario.	Uniqueidentifier
RefRecipientId	Referencia al Identificador del Beneficiario.	Uniqueidentifier
RecipientType	Tipo de Beneficiario.	Int
RecipientName	Nombre del Beneficiario.	Texto(255)
RecipientLastName	Apellidos del Beneficiario.	Texto (255)
RecipientFullName	Nombre y Apellidos del Beneficiario.	Calculado
RecipientIdentifier	DNI/CIF del Beneficiario.	Texto (10)
RecipientBirthDate	Fecha de nacimiento del Beneficiario.	DateTime
RecipientAge	Edad del Beneficiario.	Calculado
RefSexType	Referencia al Género del Beneficiario.	Uniqueidentifier
SexType	Género del Beneficiario.	Int
RecipientPhone	Teléfono del Beneficiario.	Texto (15)
RecipientEmail	Email del Beneficiario.	Texto (255)
RecipientAdress	Dirección del Beneficiario.	Texto (255)
RecipientPostalCode	Código Postal del Beneficiario.	Texto(5)
RecipientCity	Población del Beneficiario.	Texto(255)

RecipientRegion	Provincia del Beneficiario.	Texto(255)
RecipientDiagnostic	Diagnóstico del Beneficiario.	Texto (4000)
RecipientActualSituation	Situación Actual del Beneficiario.	Texto (4000)
RecipientDisabilityDescription	Descripción de la Discapacidad del Beneficiario.	Texto (4000)
RefDependenceGradeID	Referencia al Grado de Dependencia del Beneficiario.	Uniqueidentifier
RequestPercentageDisability	Porcentaje de Discapacidad del Beneficiario.	Decimal
RefRecordId	Referencia al Grado de Dependencia del Beneficiario.	Uniqueidentifier
RequestComments	Observaciones de la Solicitud.	Texto(4000)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **Records** presenta los campos de la lista de Expedientes.

Campo	Descripción	Tipo
RecordId	Identificador del Expediente.	Uniqueidentifier
RefRequestId	Referencia al Identificador de la Solicitud.	Uniqueidentifier
RecordCode	Código del Expediente.	Text(10)
RefRecordTechnician	Referencia al Identificador del Técnico.	Uniqueidentifier
RefRecordJob	Referencia al Puesto del Técnico.	Uniqueidentifier
RecordState	Estado del Expediente.	int
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **Interventions** presenta los campos de la lista de Intervenciones.

Campo	Descripción	Tipo
InterventionId	Identificador de la Intervención.	Uniqueidentifier
RefRecordId	Referencia al Identificador del Expediente.	Uniqueidentifier
RefRequestId	Referencia al Identificador de la Solicitud.	Uniqueidentifier
InterventionCode	Código del Expediente.	Text(10)
RefInterventionTechnician	Referencia al Identificador del Técnico.	Uniqueidentifier
RefInterventionDemand	Referencia a la Demanda de la Intervención.	Uniqueidentifier
InterventionOpeningDate	Fecha de inicio de la Intervención.	DateTime
InterventionDate	Fecha de la intervención	DateTime
InterventionDeadline	Fecha de cierre de la Intervención.	DateTime
InterventionDateVisit	Fecha de visita.	DateTime
InterventionActions	Acciones a realizar en la Intervención.	Text(4000)
InterventionComments	Comentarios de la Intervención.	Text(4000)
RefInterventionStatus	Estado de la Intervención.	Uniqueidentifier
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

2.2.2 Entidades secundarias

A continuación se muestran las tablas correspondientes a las entidades secundarias de la aplicación:

La ¡Error! No se encuentra el origen de la referencia. **Addresses** presenta los campos de la lista de Direcciones de los Beneficiarios.

Campo	Descripción	Tipo
AddressId	Identificador de la Dirección.	Uniqueidentifier
RefRecipientId	Referencia al Identificador dl Beneficiario.	Uniqueidentifier
Address	Dirección.	Text(255)
RecipientLastName	Nombre del Beneficiario.	Text(255)
RecipientPhone	Telefono del Beneficiario.	Text(15)
RecipientEmail	Email del Beneficiario.	Text(255)
RecipientPostalCode	Código Postal de la Dirección.	Text(5)
RecipientCity	Población de la Dirección.	Text(255)
RecipientRegion	Provincia de la Direccióon	Text(255)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **ApplicantTypes** presenta los campos de la lista de Tipos de Solicitantes.

Campo	Descripción	Tipo
ApplicantTypeId	Identificador del Tipo de Solicitate.	Uniqueidentifier
ApplicantTypeName	Nombre del Tipo de Solicitante.	Uniqueidentifier
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **DemandTypes** presenta los campos de la lista de Tipos de Demandas.

Campo	Descripción	Tipo
DemandTypeId	Identificador del Tipo de Demanda.	Uniqueidentifier
RefDemandTypeId	Referencia al Tipo de Demanda.	Uniqueidentifier
DemandTypeName	Nombre del Tipo de Demanda.	Text(255)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **DependenceGrade** presenta los campos de la lista de los Grados de Dependencia.

Campo	Descripción	Tipo
RefDependenceGradeTypeId	Referencia al Tipo de Grado de Dependencia.	Uniqueidentifier
RefRequestId	Referencia a la Solicitud.	Uniqueidentifier
Percentage	Porcentaje del Grado de Dependencia.	Float
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **DependenceGradeTypes** presenta los campos de la lista de los Tipos de Grado de Dependencia.

Campo	Descripción	Tipo
DependenceGradeId	Identificador del Tipo de Grado de Dependencia.	Uniqueidentifier
DependenceGradeTypeName	Nombre del Tipo de Grado de Dependencia.	Text(255)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **DisabilityTypes** presenta los campos de la lista de los Tipos de Discapacidad.

Campo	Descripción	Tipo
DisabilityTypeId	Identificador del Tipo de Discapacidad.	Uniqueidentifier
DisabilityTypeName	Nombre del Tipo de Discapacidad.	Text(255)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **EntityTypes** presenta los campos de la lista de los Tipos de Entidad.

Campo	Descripción	Tipo

EntityTypeId	Identificador del Tipo de Entidad.	Uniqueidentifier
EntityTypeName	Nombre del Tipo de Entidad.	Text(255)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **FunctionalLimitationTypes** presenta los campos de la lista de los Tipos de Limitaciones Funcionales.

Campo	Descripción	Tipo
FunctionalLimitationTypeId	Identificador del Tipo de Limitaciones Funcionales.	Uniqueidentifier
FunctionalLimitationTypeName	Nombre del Tipo de Limitaciones Funcionales.	Text(255)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **InterventionsDemands** presenta los campos de la lista de las Demandas de las Intervenciones.

Campo	Descripción	Tipo
RefDemandTypeId	Referencia al Tipo de Demanda.	Uniqueidentifier
RefInterventionId	Referencia a la Intervencion.	Uniqueidentifier
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **InterventionStatus** presenta los campos de la lista de los Estados de la Intervención.

Campo	Descripción	Tipo
InterventionStatusId	Identificador del Estado de la Intervención.	Uniqueidentifier
InterventionStatusName	Nombre del Estado de la Intervención.	Text(50)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **Jobs** presenta los campos de la lista de los Puestos que tienen los Técnicos.

Campo	Descripción	Tipo
RefJobTypeId	Referencia al Tipo de Puesto.	Uniqueidentifier
RefTechnicianId	Referencia al Técnico.	Uniqueidentifier
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **JobTypes** presenta los campos de la lista de los Tipos de Puestos.

Campo	Descripción	Tipo
JobTypeId	Identificador del Tipo de Puesto.	Uniqueidentifier
JobTypeName	Nombre del Tipo de Puesto.	Text(255)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **RecipientDisabilities** presenta los campos de la lista de las Discapacidades de los Beneficiarios.

Campo	Descripción	Tipo
RefRecipientId	Referencia al Beneficiario.	Uniqueidentifier
RefDisabilityTypeId	Referencia al Tipo de Discapacidad.	Uniqueidentifier
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **RecipientFunctionalLimitations** presenta los campos de la lista de las Limitaciones Funcionales de los Beneficiarios.

Campo	Descripción	Tipo
RefRecipientId	Referencia al Beneficiario.	Uniqueidentifier
RefFunctionalLimitationTypeId	Referencia al Tipo de Limitación Funcional.	Uniqueidentifier
RFSDescription	Descripción de la Limitación Funcional.	Text(255)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del	DateTime

	elemento.

La ¡Error! No se encuentra el origen de la referencia. **RecipientRelations** presenta los campos de la lista de las Relaciones de los Solicitantes con los Beneficiarios.

Campo	Descripción	Tipo
RecipientRelationId	Identificador de la Relación.	Uniqueidentifier
RecipientRelationName	Nombre de la Relación.	Text(255)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **RecipientTypes** presenta los campos de la lista de los Tipos de Beneficiarios.

Campo	Descripción	Tipo
RecipientTypeId	Identificador del Tipo de Beneficiario.	Uniqueidentifier
RecipientTypeName	Nombre del Tipo de Beneficiario.	Text(255)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **RecordDocuments** presenta los campos de la lista de los Documentos de los Expedientes.

Campo	Descripción	Tipo
DocId	Identificador del Documento.	Uniqueidentifier
RefRecordId	Referencia al Expediente.	Uniqueidentifier
RefInterventionId	Referencia a la Intervención.	Uniqueidentifier
DocName	Nombre del Documento.	Text(255)
DocExtension	Extensión del Documento.	Text(15)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **RequestDemands** presenta los campos de la lista de las Demandas de las Solicitudes.

Campo	Descripción	Tipo
RefDemandTypeId	Referencia al Tipo de Demanda.	Uniqueidentifier
RefRequestId	Referencia a la Solicitud.	Uniqueidentifier
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime

Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **RequestDisabilities** presenta los campos de la lista de las Discapacidades de las Solicitudes.

Campo	Descripción	Tipo
RefRequestId	Referencia a la Solicitud.	Uniqueidentifier
RefDisabilityTypeId	Referencia al Tipo de Discapacidad.	Uniqueidentifier
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **RequestDocuments** presenta los campos de la lista de los Documentos de las Solicitudes.

Campo	Descripción	Tipo
DocId	Identificador del Documento.	Uniqueidentifier
RefRequestId	Referencia a la Solicitud.	Uniqueidentifier
DocName	Nombre del Documento.	Text(255)
DocExtension	Extensión del Documento.	Text(15)
DocPath	Ruta del Documento.	Text(4000)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del	DateTime

	elemento.

La ¡Error! No se encuentra el origen de la referencia. **RequestFunctionalLimitations** presenta los campos de la lista de las Limitaciones Funcionales de los Beneficiarios.

Campo	Descripción	Tipo
RefRequestId	Referencia a la Solicitud.	Uniqueidentifier
RefFunctionalLimitationTypeId	Referencia al Tipo de Limitación Funcional.	Uniqueidentifier
RFSDescription	Descripción de la Limitación Funcional.	Text(255)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **SexTypes** presenta los campos de la lista de los Tipos de Genero.

Campo	Descripción	Tipo
SexId	Identificador del Tipo de Género.	Uniqueidentifier
SexName	Nombre del Tipo de Genero.	Text(50)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **SourceTypes** presenta los campos de la lista de los Tipos de Entrada de Solicitud.

Campo	Descripción	Tipo
SourceTypeId	Identificador del Tipo de Entrada.	Uniqueidentifier
SourceTypeName	Nombre del Tipo de Entrada.	Text(255)
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

La ¡Error! No se encuentra el origen de la referencia. **Technicians** presenta los campos de la lista de los Técnicos.

Campo	Descripción	Tipo
TechnicianId	Identificador del Técnico.	Uniqueidentifier
TechnicianName	Nombre del Técnico.	Text(255)
TechnicianLastName	Apellidos del Técnico.	Text(255)
TechnicianFullName	Nombre y Apellidos del Técnico.	Calculado
Deleted	Indicador de elemento eliminado.	Bit
Author	Creador del elemento.	Int
Editor	Último editor del elemento.	Int
Created	Fecha de creación del elemento.	DateTime
Modified	Fecha de última modificación del elemento.	DateTime

3. Diagramas de secuencia

3.1. Nueva Solicitud

3.1.1 Nueva Solicitud sin Beneficiario/Solicitante existente

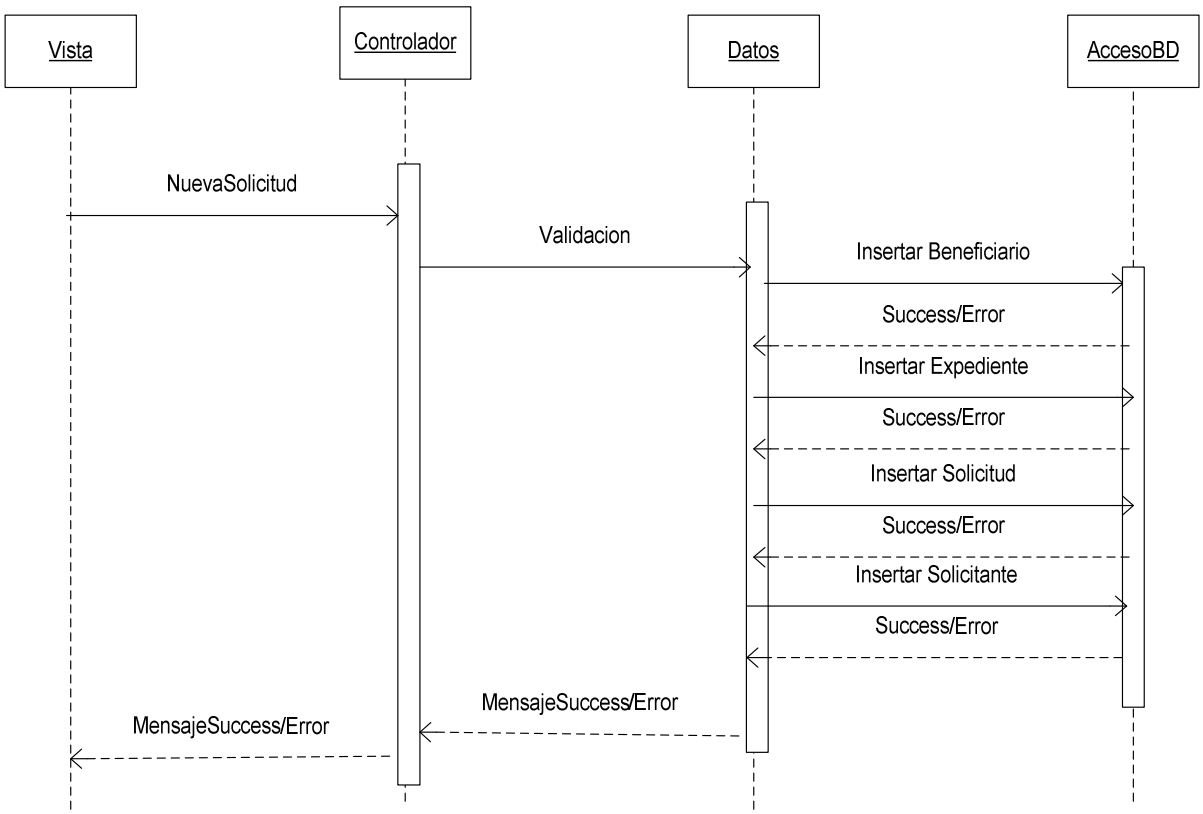


Figura 4. Diagrama de secuencia: nueva solicitud

Al crear una nueva solicitud se validarán los datos introducidos en el formulario, posteriormente a través de funciones creadas en la capa de Datos se accederá a la base de

datos insertando un nuevo beneficiario, expediente, solicitud y solicitante. Se devolverá a la vista un mensaje sobre la validación o un resumen de la solicitud creada.

3.1.2 Nueva Solicitud con Beneficiario existente

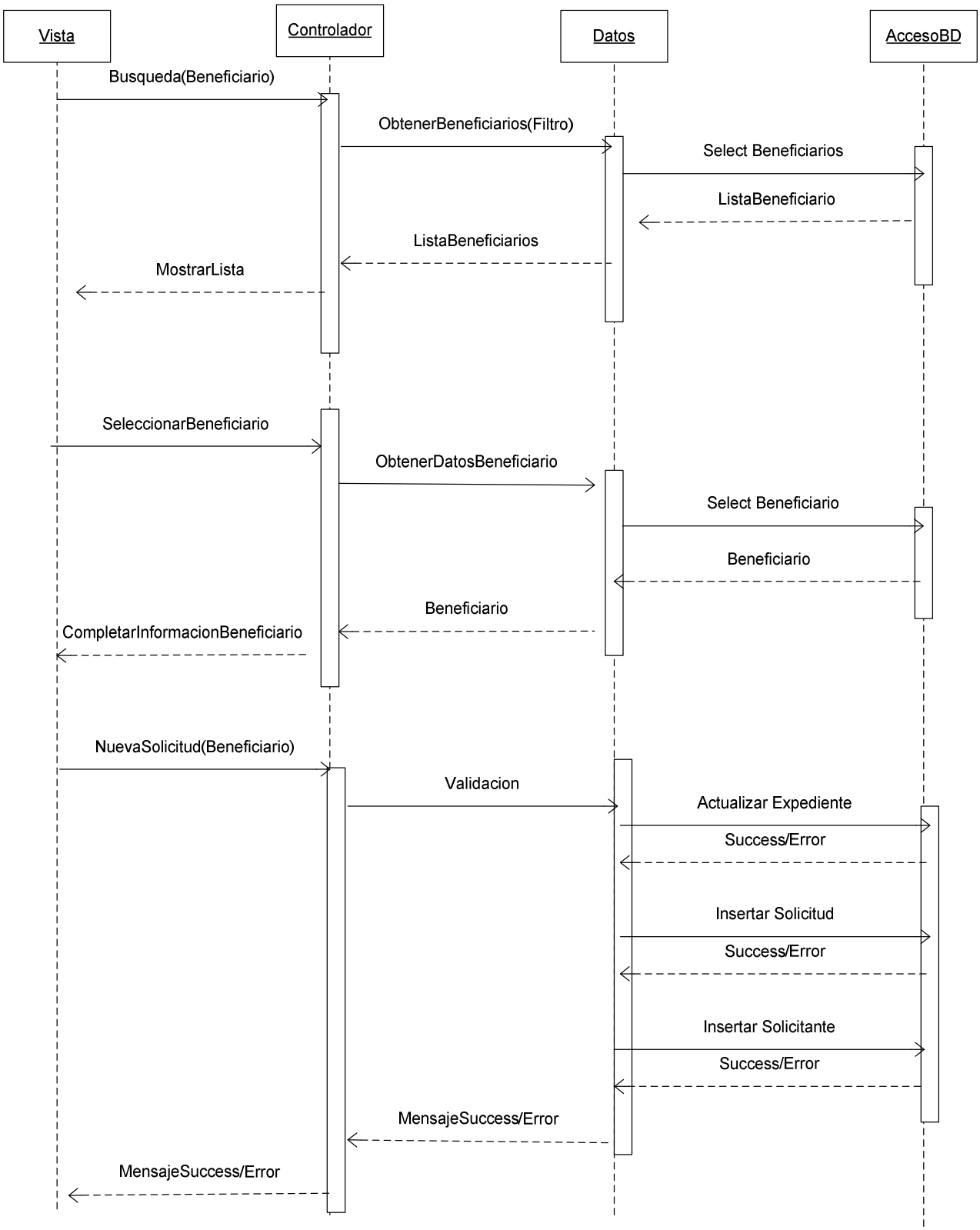


Figura 5. Diagrama de secuencia: nueva solicitud (II)

Al crear una nueva solicitud con un beneficiario existente primeramente se buscará el beneficiario introduciendo los parámetros en la vista, y posteriormente el controlador

normalizará los datos introducidos y realizará la consulta a través de la capa de datos en la base de datos. Los resultados de dicha consulta se mostrarán en la vista, donde el usuario seleccionará el beneficiario en concreto. Se accederá a la base de datos para obtener toda la información del beneficiario y el controlador la tratará y mostrará a través de la vista los datos del formulario. Seguidamente se rellenará el resto del formulario validando los datos introducidos. A través de funciones creadas en la capa de Datos se accederá a la base de datos actualizando el expediente y creando una solicitud y un solicitante. Se devolverá a la vista un mensaje sobre la validación o un resumen de la solicitud creada.

3.1.1. Nueva Solicitud con Solicitante existente

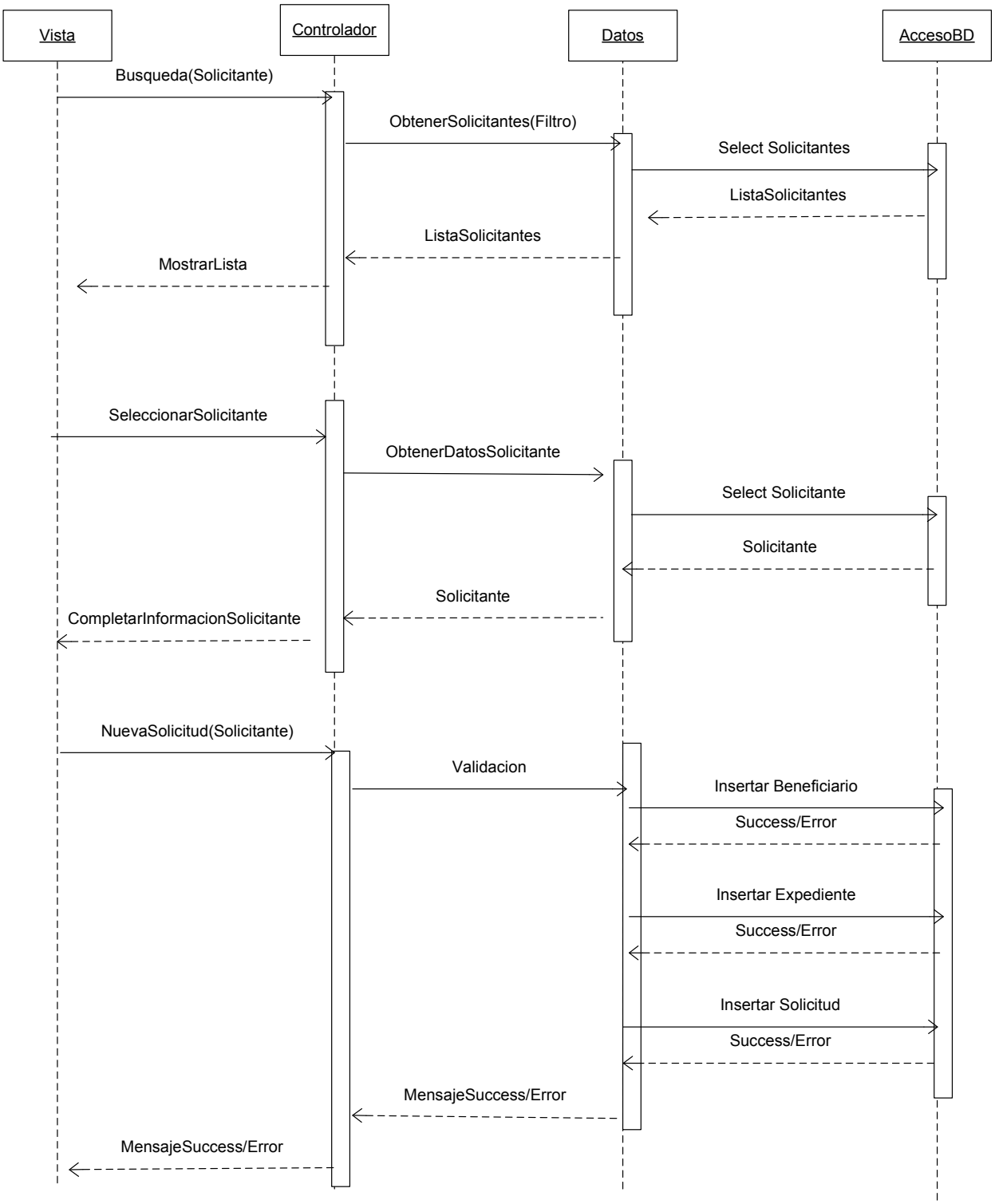


Figura 58. Diagrama de secuencia: nueva solicitud (III)

Al crear una nueva solicitud con un solicitante existente primeramente se buscará el solicitante introduciendo los parámetros en la vista, y posteriormente el controlador

normalizará los datos introducidos y realizará la consulta a través de la capa de datos en la base de datos. Los resultados de dicha consulta se mostrarán en la vista, donde el usuario seleccionará el solicitante en concreto. Se accederá a la base de datos para obtener toda la información del solicitante y el controlador la tratará y mostrará a través de la vista los datos del formulario. Seguidamente se rellenará el resto del formulario validando los datos introducidos. A través de funciones creadas en la capa de Datos se accederá a la base de datos insertando un nuevo expediente, beneficiario y solicitud. Se devolverá a la vista un mensaje sobre la validación o un resumen de la solicitud creada.

3.2. Nueva Intervención

3.2.1. Nueva Intervención

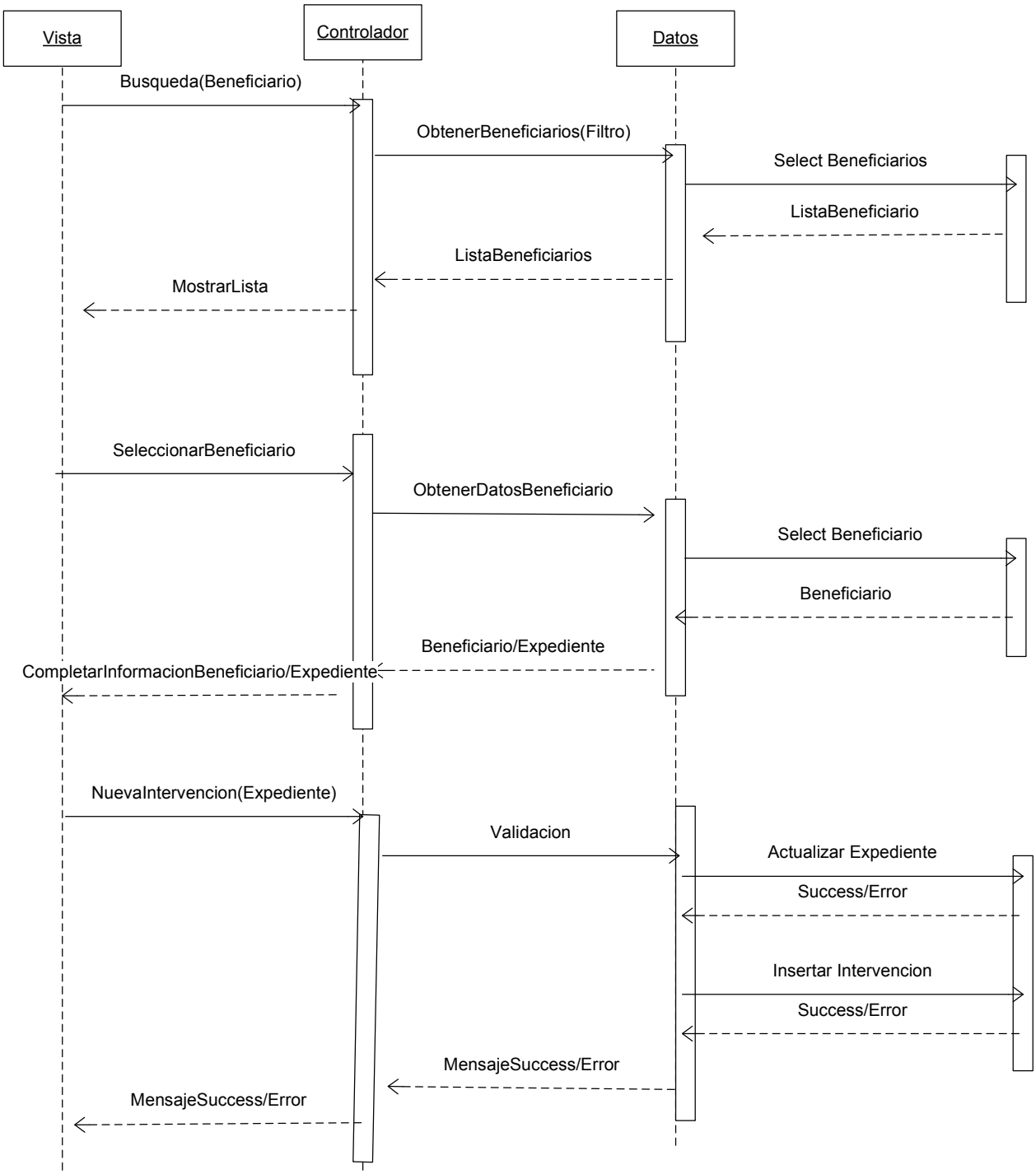


Figura 759. Diagrama de secuencia: nueva intervención

Al crear una nueva intervención primeramente se buscará el beneficiario introduciendo los parámetros en la vista, y posteriormente el controlador normalizará los datos introducidos y realizará la consulta a través de la capa de datos en la base de datos. Los resultados de dicha consulta se mostrarán en la vista, donde el usuario seleccionará el beneficiario en concreto. Se accederá a la base de datos para obtener toda la información del beneficiario y su expediente correspondiente y el controlador la tratará y mostrará a través de la vista los datos del formulario. Seguidamente se rellenará el resto del formulario validando los datos introducidos. A través de funciones creadas en la capa de Datos se accederá a la base de datos actualizando el expediente y creando una intervención. Se devolverá a la vista un mensaje sobre la validación o un resumen de la solicitud creada.

Al crear una nueva intervención primeramente se buscará el beneficiario introduciendo los parámetros en la vista, y posteriormente el controlador normalizará los datos introducidos y realizará la consulta a través de la capa de datos en la base de datos. Los resultados de dicha consulta se mostrarán en la vista, donde el usuario seleccionará el beneficiario en concreto. Se accederá a la base de datos para obtener toda la información del beneficiario y su expediente correspondiente y el controlador la tratará y mostrará a través de la vista los datos del formulario. A continuación se rellenará una lista en la vista con las solicitudes relativas a dicho beneficiario, permitiendo seleccionar una y realizar la consulta a la base de datos obteniendo la información de dicha solicitud. Seguidamente se rellenará el resto del formulario validando los datos introducidos. A través de funciones creadas en la capa de Datos se accederá a la base de datos actualizando el expediente, relacionando la solicitud con la intervención y creando una intervención. Se devolverá a la vista un mensaje sobre la validación o un resumen de la solicitud creada.

3.3. Búsqueda

3.3.1. Consultar datos

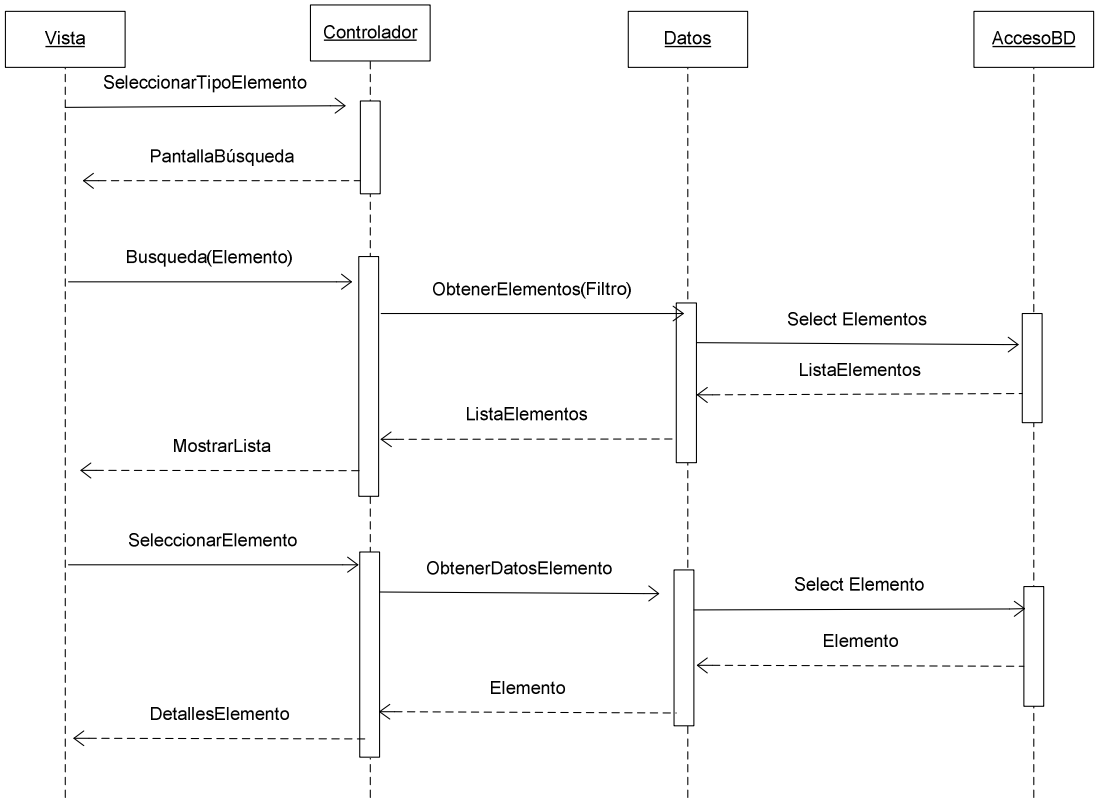


Figura 8. Diagrama de secuencia: consultar datos

Para realizar una consulta primeramente se seleccionará en la vista el tipo de elemento a buscar (expediente, intervención, solicitante, beneficiario, solicitud). Posteriormente el controlador redirigirá al usuario hacía otra vista con un formulario que permite introducir los filtros de búsqueda para seleccionar el elemento concreto a buscar. Con los filtros se hace una consulta en la base de datos que mostrará una lista con todos los elementos que cumplan esos parámetros y se permitirá escoger el elemento en concreto que desea consultar el usuario, lo que desencadenará otra consulta para obtener toda la información sobre el elemento en concreto si se acabará mostrando en la vista.

3.3.2. Generar informe

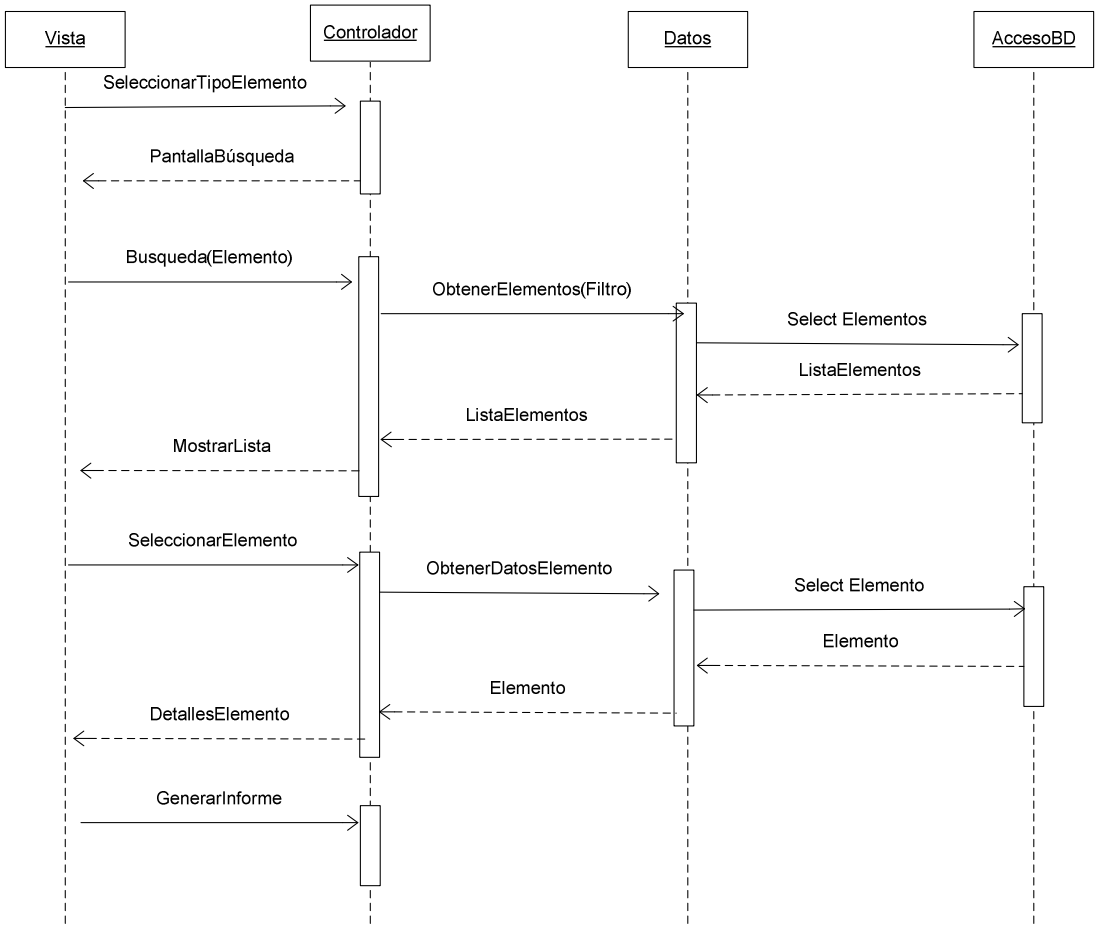


Figura 9. Diagrama de secuencia: generar informe

Para generar un informe primeramente se seleccionará en la vista el tipo de elemento a buscar (expediente, intervención, solicitante, beneficiario, solicitud). Posteriormente el controlador redirigirá al usuario hacía otra vista con un formulario que permite introducir los filtros de búsqueda para seleccionar el elemento concreto a buscar. Con los filtros se hace una consulta en la base de datos que mostrará una lista con todos los elementos que cumplan esos parámetros y se permitirá escoger el elemento en concreto que desea consultar el usuario, lo que desencadenará otra consulta para obtener toda la información sobre el elemento en concreto si se acabará mostrando en la vista. En esta pantalla existe la opción de generar informe, la cual guardará en un directorio previamente configurado un informe con los datos del elemento consultado.

3.3.3. Modificar datos

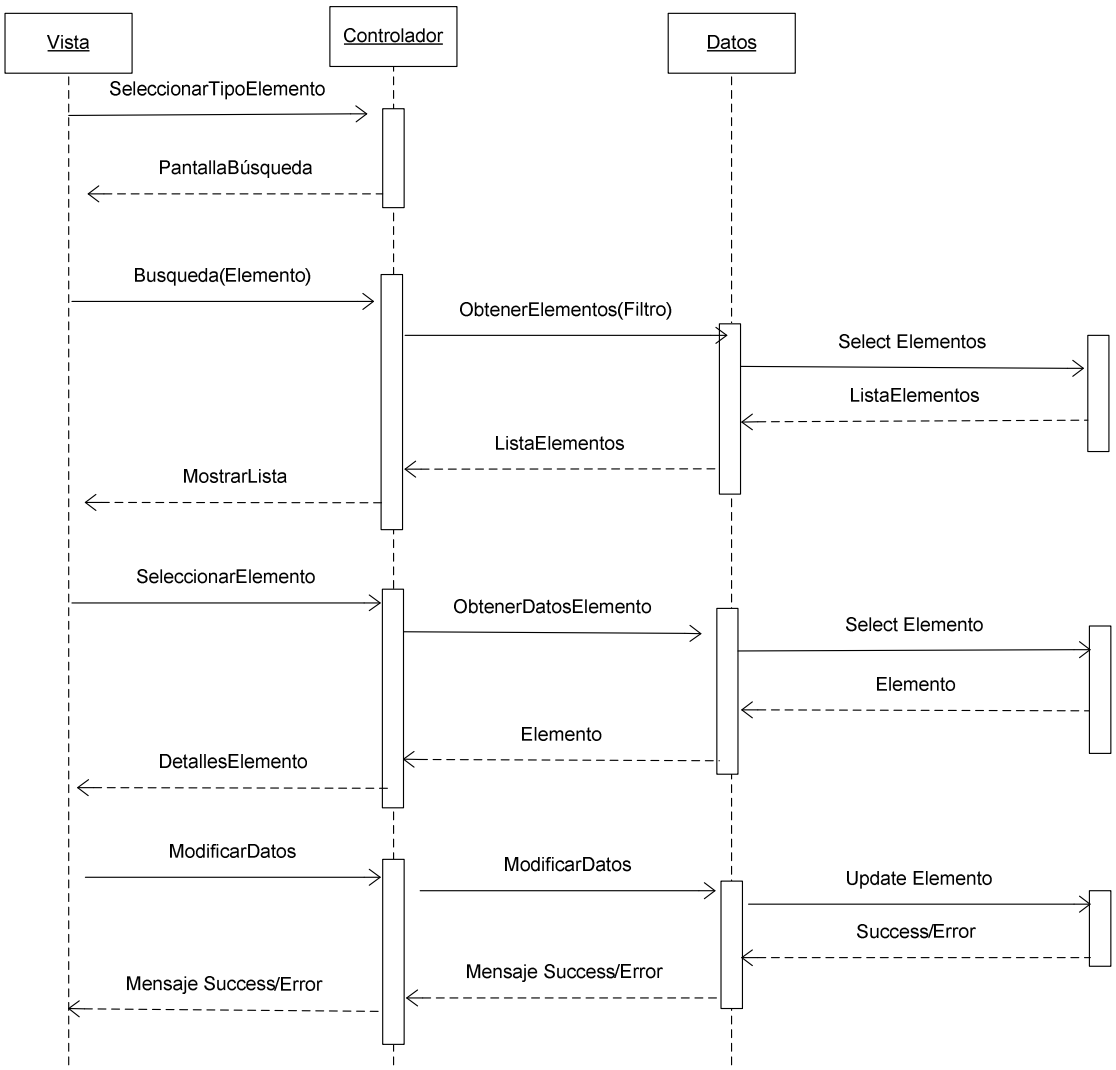


Figura 10. Diagrama de secuencia: modificar datos

Para modificar un elemento existente en el sistema primeramente se seleccionará en la vista el tipo de elemento a buscar (expediente, intervención, solicitante, beneficiario, solicitud). Posteriormente el controlador redirigirá al usuario hacía otra vista con un formulario que permite introducir los filtros de búsqueda para seleccionar el elemento concreto a buscar. Con los filtros se hace una consulta en la base de datos que mostrará una lista con todos los elementos que cumplan esos parámetros y se permitirá escoger el elemento en concreto que desea consultar el usuario, lo que desencadenará otra consulta para obtener toda la información sobre el elemento en concreto si se acabará mostrando en la vista. En esta pantalla el formulario permitirá modificar ciertos campos del elemento, realizando posteriormente la actualización en la base de datos.

3.4. Estadística

3.4.1. Mostrar estadística

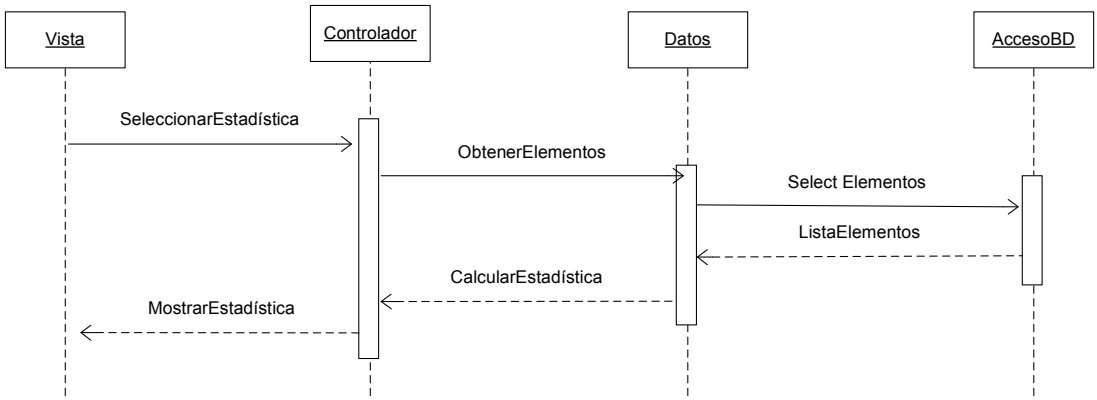


Figura 11. Diagrama de secuencia: mostrar estadística

Para consultar una estadística, primeramente se seleccionará la estadística a mostrar. El siguiente paso es obtener de la base de datos todos los elementos que se necesiten para calcular la estadística. Una vez obtenido el conjunto de elementos se accederá a los campos necesarios para calcular la estadística en el controlador y se mostrará por pantalla el resultado.

3.5. Administración

3.5.1. Crear

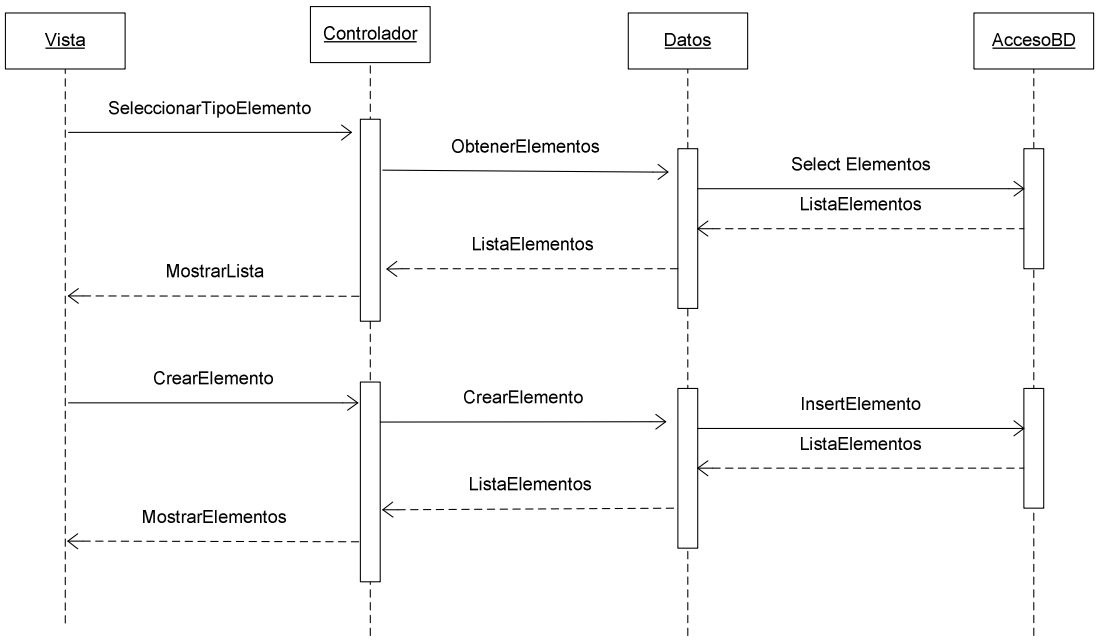


Figura 12. Diagrama de secuencia: crear

Para crear un elemento primeramente se seleccionara el tipo de elementos a administrar de las opciones posibles en la vista, realizando una consulta en la base de datos y mostrando los elementos existentes de ese tipo de datos a administrar. Seguidamente se seleccionará la opción de crear un nuevo elemento, introduciendo el nombre, creándose una nueva entrada correspondiente en la tabla de la base de datos y volviendo a mostrar en la vista la lista de todos los elementos incluyendo el elemento creado.

3.5.2. Modificar

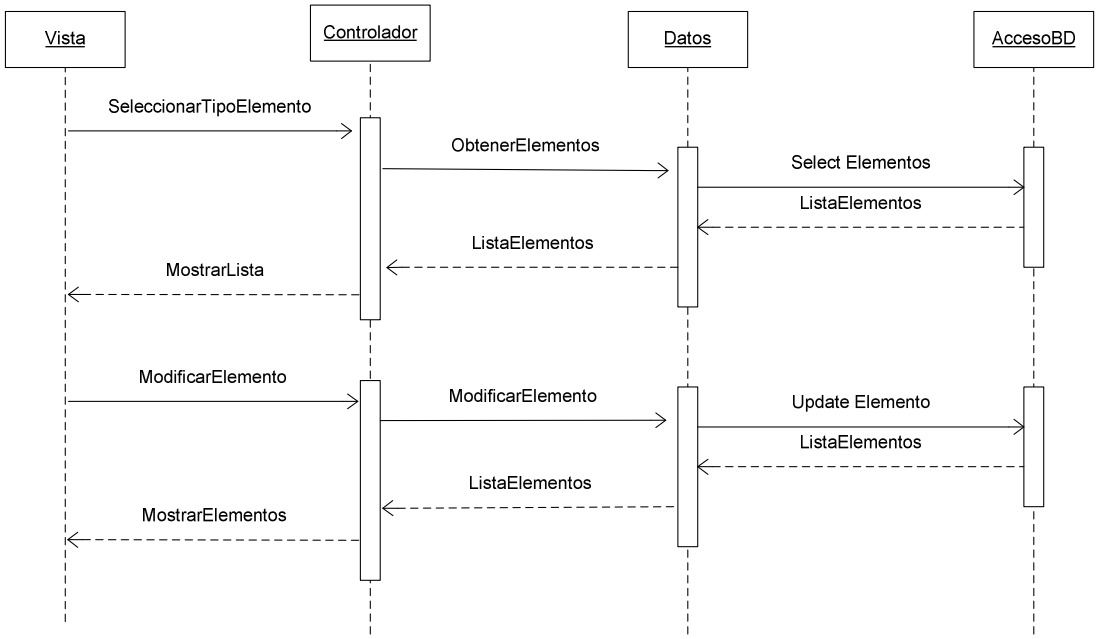


Figura 13. Diagrama de secuencia: modificar

Para modificar un elemento primeramente se seleccionara el tipo de elementos a administrar de las opciones posibles en la vista, realizando una consulta en la base de datos y mostrando los elementos existentes de ese tipo de datos a administrar. Seguidamente se seleccionará la opción de modificar el elemento deseado, introduciendo el nombre, actualizándose la entrada correspondiente en la tabla de la base de datos y volviendo a mostrar en la vista la lista de todos los elementos incluyendo la modificación del elemento.

3.5.3. Eliminar

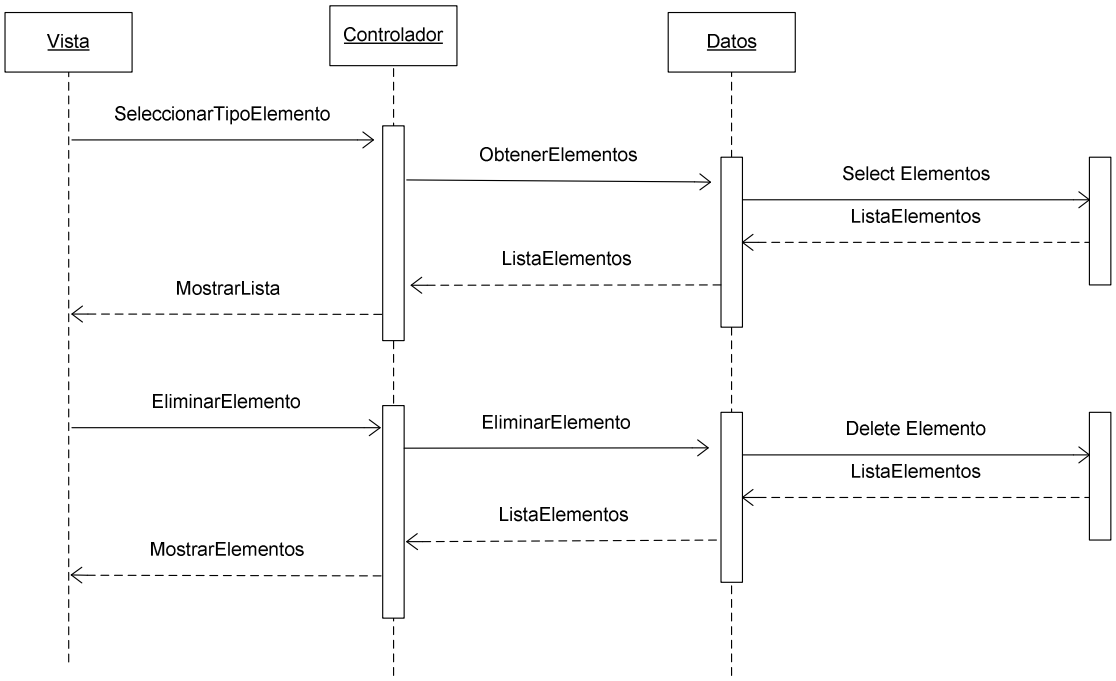


Figura 14. Diagrama de secuencia: eliminar

Para eliminar un elemento primeramente se seleccionara el tipo de elementos a administrar de las opciones posibles en la vista, realizando una consulta en la base de datos y mostrando los elementos existentes de ese tipo de datos a administrar. Seguidamente se seleccionará la opción de eliminar el elemento deseado, introduciendo el nombre, borrándose entrada correspondiente en la tabla de la base de datos y volviendo a mostrar en la vista la lista de todos los elementos excluyendo el elemento

4. Diagrama de estados

A continuación se muestra el diagrama de estados para las tareas de crear una nueva aplicación, nueva intervención, consulta de información, administración de la aplicación y estadística, así se facilitara la compresión del funcionamiento de la aplicación.

4.1. Nueva Solicitud

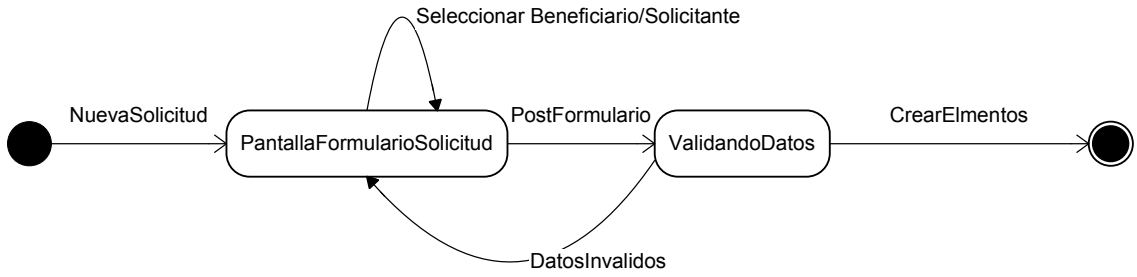


Figura 15. Diagrama de estados: nueva solicitud

Para introducir una nueva solicitud la aplicación permanecerá en la pantalla de formulario de solicitud, permitiendo acciones como seleccionar un beneficiario o solicitante. En este estado se podrá realizar la acción de Enviar Formulario (POST), lo que pasará al estado de validación de datos. En caso de que los datos introducidos sean inválidos se volverá a la pantalla de formulario de solicitud con unos mensajes sobre los datos incorrectos, y en caso de que sean válidos se crearán los elementos correspondientes en el sistema y en la base de datos.

4.2. Nueva Intervención

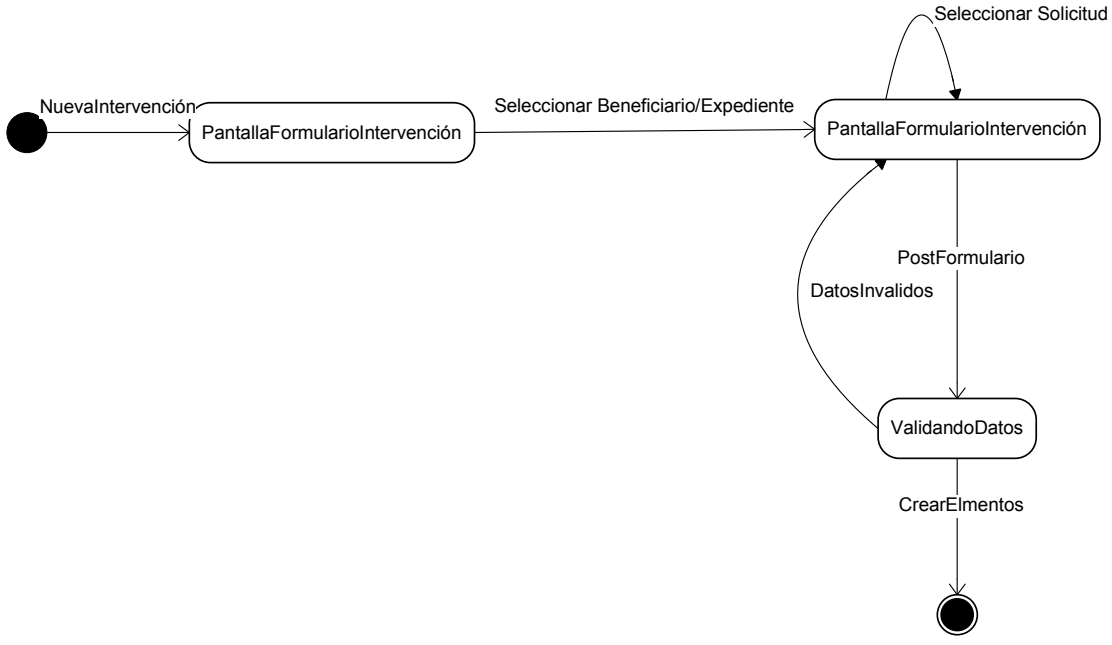


Figura 16. Diagrama de estados: nueva intervención

Al crear una nueva intervención la aplicación permanecerá en el estado de la pantalla de formulario de la intervención. Al seleccionar el expediente/beneficiario se autocompletará las posibles solicitudes a seleccionar. En este estado se podrá realizar la acción de Enviar Formulario (POST), lo que pasará al estado de validación de datos. En caso de que los datos introducidos sean inválidos se volverá a la pantalla de formulario de intervención con unos mensajes sobre los datos incorrectos, y en caso de que sean válidos se crearán los elementos correspondientes en el sistema y en la base de datos.

4.3. Búsqueda

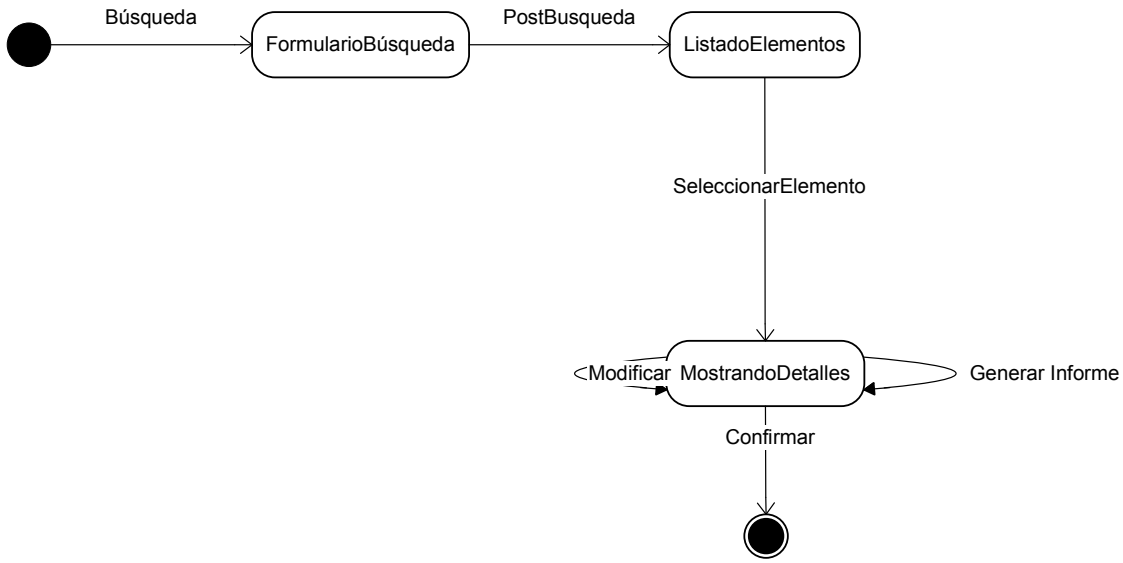


Figura 17. Diagrama de estados: búsqueda

Pará realizar una búsqueda el sistema permanecerá en la pantalla del formulario del filtro de búsqueda. Para realizar la búsqueda se realizará un POST de los datos a través de un botón, pasando al estado del listado de elementos. Una vez se seleccione uno pasará a una pantalla que muestra los detalles del elemento. Desde esa pantalla se podrá modificar algún dato o generar el informe si el elemento es susceptible de generación de informes.

4.4. Estadística

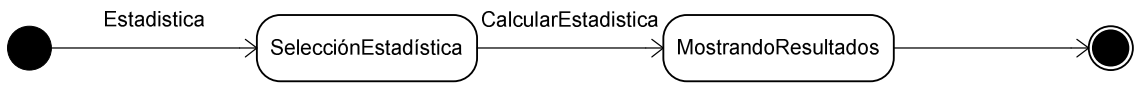


Figura 18. Diagrama de estados: estadística

Para calcular una estadística el sistema permanecerá en la pantalla de selección de estadística. Al seleccionar una la aplicación calculara la estadística y pasará a otra pantalla donde mostrará los resultados obtenidos.

4.5. Administración

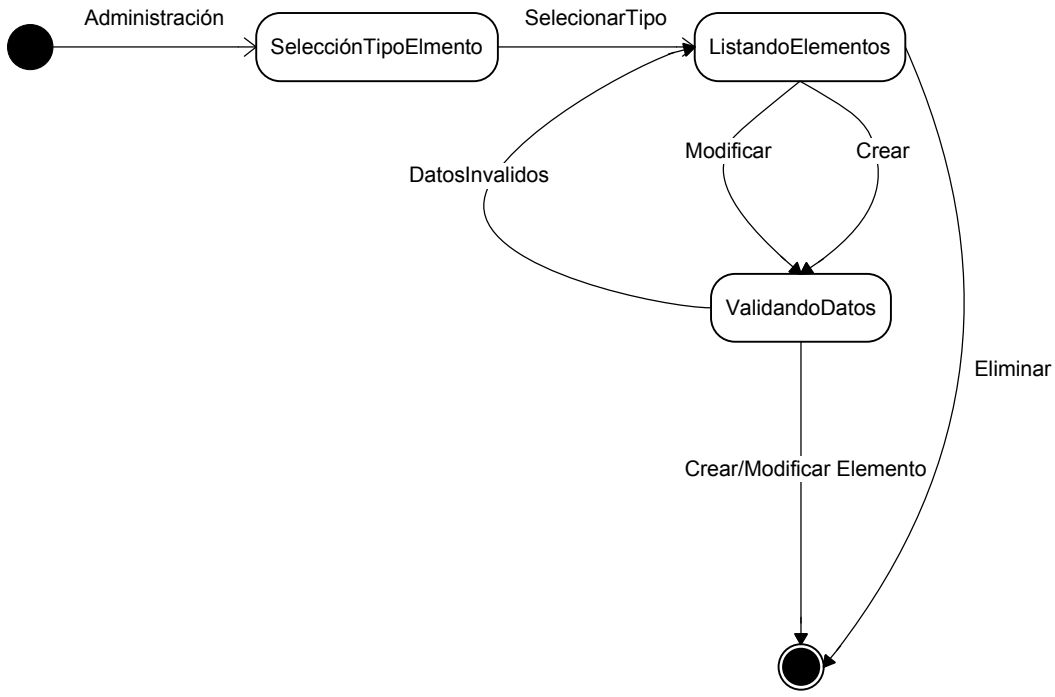


Figura 19. Diagrama de estado: administración

Para administrar los elementos configurables de la aplicación se accederá a la pantalla de selección de tipo de elemento. Cuando se seleccione un tipo de elemento se accederá al estado donde aparecerá por pantalla la lista con los elementos de ese tipo existentes en el sistema. En este punto se podrá eliminar un elemento, crear o modificar un elemento. Para crear o modificar un elemento se validará el texto introducido.

5. Diagrama de componentes

Para su mejor compresión, se ha dividido el diagrama de componentes desde el punto de vista de:

- **Gestión de expedientes**: Componente representa la aplicación.

- **Beneficiarios, expedientes, solicitudes, solicitantes, intervenciones:** Cada uno representa un componente que representa un elemento del mismo tipo con el que se maneja la estructura relacionada con dicho elemento.
- **Seguridad:** Representa los mecanismos de seguridad de la aplicación.
- **Base de datos:** Componente que representa la propia base de datos utilizada en la aplicación.

Para la realización de los siguientes diagramas se considera la clases que representan los elementos como componentes.

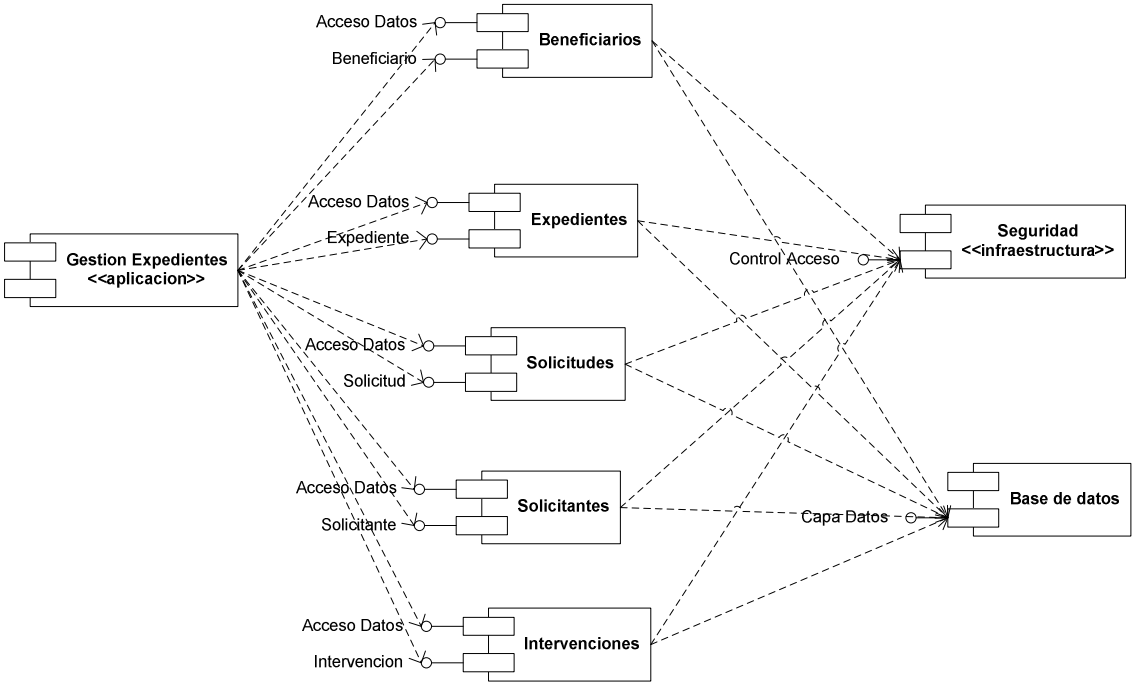


Figura 20. Diagrama de componentes

Mediante la aplicación y a través de la interfaz de cada elemento se accede a la información de los mismos, que tienen un control de acceso proporcionado por el componente de seguridad y que se relacionan directamente con la base de datos para modificar, crear y eliminar información.

6. Diseño del plan de pruebas

6.1. Pruebas de la aplicación

6.1.1. Pruebas de accesibilidad

Tienen como misión comprobar si la aplicación es capaz de acceder a todas sus opciones de trabajo.

Caso de prueba 1

Objetivos	Comprobar el acceso a la aplicación mediante la identificación.
Entradas	El identificador y la contraseña de usuario
Proceso de prueba	Desde la interfaz de la aplicación se accede a la capa de seguridad y comprueba el identificador y la contraseña del usuario.
Salida deseada	Permite el acceso a la aplicación en caso de que los datos introducidos sean correctos o permite volver a introducir los datos en caso de que sean erróneos.

Caso de prueba 2

Objetivos	Comprobar el acceso a la función de crear una solicitud.
Entradas	Formulario de entrada de “nueva solicitud”.
Proceso de prueba	Se rellenará el formulario de entrada y se enviará a la capa de datos.
Salida deseada	Crear una nueva solicitud (y beneficiario, expediente, solicitante en caso de no existir) o mostrar de nuevo el formulario con mensajes de validación si alguno de los datos introducidos no es válido.

Caso de prueba 3

Objetivos	Comprobar el acceso a la función de buscar.
Entradas	Filtro de búsqueda.
Proceso de prueba	Se realizará una consulta con el filtro introducido a la base de datos.
Salida deseada	Mostrara los elementos existentes que coincidan con el filtro.

Caso de prueba 4

Objetivos	Comprobar el acceso a la función de crear una intervención.
Entradas	Formulario de entrada de “nueva intervencion”.
Proceso de prueba	Se rellenará el formulario de entrada y se enviará a la capa de datos.
Salida deseada	Crear una nueva solicitud o mostrar de nuevo el formulario con mensajes de validación si alguno de los datos introducidos no es válido o si no existe beneficiario/expediente.

Caso de prueba 5

Objetivos	Comprobar el acceso a la función de estadística.
Entradas	Tipo de estadística.
Proceso de prueba	Se realizará una consulta con el filtro introducido a la base de datos y se realizarán los cálculos necesarios.
Salida deseada	Mostrar la estadística solicitada.

Caso de prueba 6

Objetivos	Comprobar el acceso a la función de administración.
Entradas	El texto de la nueva opción o la opción de eliminar.
Proceso de prueba	Se accederá a la base de datos para crear/modificar/eliminar la tabla a administrar.
Salida deseada	Modificación de la base de datos en concordancia con la entrada introducida.

6.2. *Pruebas caja negra*

Las pruebas de caja negra tienen como misión comprobar si la aplicación cumple los requisitos funcionales establecidos.

6.2.1. Clases de equivalencia

Nota: IC01, IC02,… se refiere a Identificador de Clase

ENTRADA	CLASES VALIDAS	CLASES INVALIDAS
Nombre de solicitante en nueva solicitud, número de caracteres	>= 1 **IC01**	Empty **IC02**
Nombre de beneficiario en nueva solicitud, número de caracteres	>= 1 **IC03**	Empty **IC04**
DNI de beneficiario en nueva solicitud	^(([A-Z]\d{8})|(\d{8}[A-Z])) **IC05**	Distinto **IC06**
Tipo de demanda en nueva solicitud	Si **IC07**	No **IC08**
Diagnostico en nueva solicitud	Si **IC09**	No **IC10**
Grado de dependencia en nueva solicitud	Si **IC11**	No **IC12**
Beneficiario/Expediente en nueva intervención	Si **IC13**	No **IC14**

SALIDA	CLASES VALIDAS	CLASES INVALIDAS
Se crea una nueva solicitud si los datos introducidos son válidos	Si **IC15**	No hay
Se crea un nuevo beneficiario y expediente si al crear una nueva solicitud no existía dicho beneficiario y expediente	Si **IC16**	No hay
Se crea un nuevo solicitante si al crear una nueva solicitud no existía dicho solicitante	Si **IC17**	No hay
Se crea una nueva intervención si los datos introducidos son válidos	Si **IC18**	No hay
La opción de buscar devuelve una lista	Si **IC19**	No hay
La opción administrar modifica la base de datos	Si **IC20**	No hay
La opción estadística devuelve los cálculos deseados	Si **IC21**	No hay

6.2.2. Conjunto de pruebas

Caso de prueba 7

Objetivos	Crear una nueva solicitud sin solicitante ni beneficiario previo
Entradas	Formulario de entrada con todos los datos completos
Proceso de prueba	Se rellenará el formulario de entrada y se enviará a la capa de datos que añadirá los elementos necesarios a la base de datos.
Salida deseada	Se crea una nueva solicitud, solicitante, beneficiario y expediente con los datos introducidos.
Clases cubiertas	IC01, IC03, IC05, IC07, IC09, IC11, IC15, IC16, IC17

Caso de prueba 8

Objetivos	Crear una nueva solicitud
Entradas	No se introduce el nombre del solicitante
Proceso de prueba	Se rellenará el formulario de entrada y se enviará a la capa de datos que devolverá un error
Salida deseada	Se muestra un mensaje de validación.
Clases cubiertas	IC02

Caso de prueba 9

Objetivos	Crear una nueva solicitud
Entradas	No se introduce el nombre del beneficiario
Proceso de prueba	Se rellenará el formulario de entrada y se enviará a la capa de datos que devolverá un error
Salida deseada	Se muestra un mensaje de validación.
Clases cubiertas	IC04

Caso de prueba 10

Objetivos	Crear una nueva solicitud
Entradas	No se introduce el DNI con formato correcto
Proceso de prueba	Se rellenará el formulario de entrada y se enviará a la capa de datos que devolverá un error
Salida deseada	Se muestra un mensaje de validación.
Clases cubiertas	IC06

Caso de prueba 11

Objetivos	Crear una nueva solicitud
Entradas	No se introduce el tipo de demanda
Proceso de prueba	Se rellenará el formulario de entrada y se enviará a la capa de datos que devolverá un error
Salida deseada	Se muestra un mensaje de validación.
Clases cubiertas	IC08

Caso de prueba 12

Objetivos	Crear una nueva solicitud
Entradas	No se introduce el diagnostico
Proceso de prueba	Se rellenará el formulario de entrada y se enviará a la capa de datos que devolverá un error
Salida deseada	Se muestra un mensaje de validación.
Clases cubiertas	IC10

Caso de prueba 13

Objetivos	Crear una nueva solicitud
Entradas	No se introduce el grado de dependencia
Proceso de prueba	Se rellenará el formulario de entrada y se enviará a la capa de datos que devolverá un error
Salida deseada	Se muestra un mensaje de validación.
Clases cubiertas	IC12

Caso de prueba 14

Objetivos	Crear una nueva intervención
Entradas	Formulario de entrada con todos los datos completos
Proceso de prueba	Se rellenará el formulario de entrada y se enviará a la capa de datos que añadirá una intervención a la base de datos.
Salida deseada	Se crea una nueva intervención.
Clases cubiertas	IC13, IC18

Caso de prueba 15

Objetivos	Crear una nueva intervención
Entradas	No se introduce beneficiario/expediente
Proceso de prueba	Se rellenará el formulario de entrada y se enviará a la capa de datos que devolverá un error
Salida deseada	Se muestra un mensaje de validación.
Clases cubiertas	IC14

Caso de prueba 16

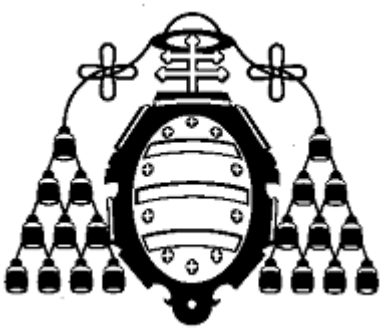
Objetivos	Realizar una búsqueda de un beneficiario
Entradas	Se introduce parcialmente el nombre del beneficiario en el filtro
Proceso de prueba	Se rellenará el formulario de entrada y se realizará la consulta devolviendo los que cumplen con la condición.
Salida deseada	Devuelve una lista con los beneficiarios que contienen la cadena introducida
Clases cubiertas	IC19

Caso de prueba 17

Objetivos	Modificar un tipo de entrada
Entradas	Se selecciona la opción modificar de la tabla de tipo de entrada en administración y se introduce la nueva cadena.
Proceso de prueba	Se realizará una modificación en la base de datos.
Salida deseada	Muestra los tipos de entrada actualizados.
Clases cubiertas	IC20

Caso de prueba 18

Objetivos	Realizar una estadística de las solicitudes
Entradas	Se selecciona la opción de estadística de solicitudes
Proceso de prueba	Se realizará la consulta a la base de datos y se realizarán cálculos con el resultado de la consulta.
Salida deseada	Muestra por pantalla los datos deseados.
Clases cubiertas	IC21



UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

HERRAMIENTA DE GESTIÓN DE EXPEDIENTES PARA EL CAT

DOCUMENTO Nº V

MANUAL TÉCNICO

ZEUS PÉREZ GARCÍA
JUNIO 2012

ÁREA DE TELEMÁTICA
TUTOR: XICU XABIEL GARCÍA PAÑEDA

ÍNDICE DEL MANUAL TÉCNICO

ÍNDICE DEL MANUAL TÉCNICO

MANUAL TÉCNICO

1. Recursos necesarios

1.1. Conocimientos necesarios

- Para los desarrolladores
  - Aplicaciones
    - Sistema Operativo Windows
    - Microsoft Visual Studio .Net 2010
    - Microsoft SQL Server 2008 R2
  - Tecnologías
    - C#
    - JavaScript
    - AJAX
    - JSon
    - JQuery
    - HTML
    - CSS
    - SQL
    - ASP
    - XML

- Para el instalador y responsable del sistema
  - Sistema Operativo Windows Server 2003

- Para el usuario de la aplicación
  - Microsoft Internet Explorer 5.0 o navegador equivalente.

1.2. Requisitos hardware

El hardware necesario es:

- **PC**: Se recomiendan como requisitos mínimos un servidor con procesador Intel Xeon CPU E5506 a una frecuencia de 2.14 GHz y con 4GB de memoria RAM o similar.

1.3. Requisitos software

El software necesario para la instalación y uso del proyecto es:

- **Sistema operativo Windows Server 2003 o superior**.

- Motor de Base de Datos SQL Server 2008 o superior.

2. Manual de instalación

En este caso la aplicación creada va dirigida a instalarse en un servidor para que sea accesible vía web por los usuarios de la organización. Por lo tanto la instalación de la aplicación debe realizarse por los técnicos de sistemas de la organización donde se implante, dependiendo en gran medida del tipo de servidor en el que se pretenda instalar.

Para el desarrollo de la misma, se debe abrir el archivo .sln con Microsoft Visual Studio 2010 o superior. Como punto importante para que se ejecute correctamente, hay que tener en cuenta que debe estar conectada dicha aplicación con la base de datos. Por lo tanto es necesario configurar el enlace con el motor de la base de datos. Para ello hay que modificar la *connectionstring* del archivo *CATWeb.Model\App.config* según el equipo y el motor de base de datos con el que se este desarrollando la aplicación.

3. Manual del programador

La aplicación ha sido implementada utilizando .NET 2010. Bajo esta tecnología se ha desarrollado una aplicación web Modelo Vista-Controlador, utilizando tecnologías como C#, HTML, JSON, AJAX, JavaScript, JQuery ASP y SQL.

A continuación se enumeran las funciones y acciones implementadas en la aplicación se explica su funcionamiento.

3.1. Interfaces

En CATWeb.Model\Interfaces están definidos los prototipos de los métodos a utilizar por la aplicación, aunque la implementación se encuentra en CATWeb.Model\Repositories.

3.1.1. IAddresses

IEnumerable<Addresses> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de direccciones de la base de datos.

Addresses GetAddress(Guid AddressId)

- **Necesita:** el identificador de una dirección concreta.
- **Produce:** devuelve elemento de la tabla correspondiente con el identificador de entrada y todos sus campos.

Guid AddAddress(Addresses address)

- **Necesita:** una estructura del tipo dirección.
- **Produce:** inserta el elemento de entrada en la base de datos y devuelve el identificador de dicho elemento.

List<Addresses> ListAddresses(Guid RecipientId)

- **Necesita:** el identificador de un beneficiario.
- **Produce:** devuelve una lista con las direcciones correspondientes al beneficiario con el identificador de entrada.

3.1.2. IApplicants

IEnumerable<Applicants> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de solicitantes de la base de datos.

Applicants GetApplicant(Guid? ApplicantId)

- **Necesita:** el identificador de un solicitante concreto.
- **Produce:** devuelve elemento de la tabla correspondiente con el identificador de entrada y todos sus campos.

Guid AddApplicant(Applicants applicant)

- **Necesita:** una estructura del tipo solicitante.
- **Produce:** inserta el elemento de entrada en la base de datos y devuelve el identificador de dicho elemento.

void UpdateApplicant(Applicants applicant)

- **Necesita:** una estructura del tipo solicitante existente en la base de datos.
- **Produce:** actualiza el elemento de entrada en la base de datos .

void DeleteApplicant(Guid ApplicantId)

- **Necesita:** el identificador del solicitante a borrar.
- **Produce:** marca como eliminado en la base de datos el solicitante con el identificador de entrada.

List<Applicants> ListApplicant(string ApplicantString)

- **Necesita:** un string que será la subcadena contenida en el nombre del solicitante a buscar.
- **Produce:** una lista de solicitantes que contengan en su nombre la cadena de entrada.

IEnumerable<Applicants> Search(string SortOrder, bool AscOrder, int? page)

- **Necesita:** un string que indica el campo por el que se ordenará la lista devuelta, un booleano que indica si el orden es ascendente o descendente y un entero que indicará el número de resultados por página.
- **Produce:** una lista enumerable con todos los solicitantes ordenada según los parámetros de entrada.

IEnumerable<Applicants> Search(string SortOrder, bool AscOrder, int? page, string LastName, string Name, string City)

- **Necesita:** un string que indica el campo por el que se ordenará la lista devuelta, un booleano que indica si el orden es ascendente o descendente, un entero que índica el número de elementos mostrados por página y tres strings, de los cuales solo uno de ellos tendrá valor e índica el campo y subcadena mediante la cual se buscarán los elementos.
- **Produce:** una lista enumerable de soliciantes ordenada según los parámetros de entrada y con los elementos que contengan en el campo indicado del elemento la cadena introducida.

3.1.3. IApplicantTypes

IEnumerable<ApplicantTypes> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de tipos de solicitantes de la base de datos.

Guid AddApplicantType(string ApplicantType)

- **Necesita:** un string que indica el texto a añadir en la tabla.
- **Produce:** añade en la tabla de la base de datos una entrada con el texto de entrada.

void UpdateApplicantType(Guid TypeId, string TypeName)

- **Necesita:** un identificador del tipo de soliciante a modificar y un string que indica el nuevo texto del tipo de solicitante.
- **Produce:** modifica e la tabla de la base de datos la entrada que coincide con el identificador de entrada y actualiza el elemento con el texto de entrada.

void DeleteApplicantType(Guid TypeId)

- **Necesita:** el identificador del tipo de solicitante a eliminar.
- **Produce:** marca como eliminado en la base de datos el tipo de solicitante que coincide con el identificador de entrada.

3.1.4. IDemandTypes

IEnumerable<DemandTypes> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de tipos de demanda de la base de datos.

IEnumerable<DemandTypes> GetAllItemsRoot()

- **Produce:** una lista enumerable con todos los tipos de demanda que no tienen padre en la estructura de arból.

IEnumerable<DemandTypes> GetAllItemsChild(Guid parentDemandId)

- **Necesita:** un identificador del tipo de demanda del que se quieren obtener los hijos.
- **Produce:** una lista enumerable con todos los hijos del tipo de demanda que coincide con el identificador de entrada.

Guid AddDemandType(string DemandType)

- **Necesita:** una cadena con el texto del tipo de demanada a añadir.
- **Produce:** añade en la tabla de tipo de demandas una entrada con el texto de entrada.

Guid AddDemandType(string DemandTypeName, Guid? FatherDemandId)

- **Necesita:** una cadena con el texto del tipo de demanada a añadir y el identificador del tipo de demanda padre del elemento a añadir.
- **Produce: :** añade en la tabla de tipo de demandas una entrada con el texto de entrada y que sea hijo del elemento con el identificador dado en la estructura de tipo arbol.

void UpdateDemandType(Guid TypeId, string TypeName)

- **Necesita:** el identificador del tipo de demanda a actualizar y una cadena con el nuevo texto.
- **Produce:** actualiza el texto del tipo de demanda que coincide con el identificador de entrada con el texto de entrada.

void DeleteDemandType(Guid TypeId)

- **Necesita:** el identificador de tipo de demanda a eliminar.
- **Produce:** marca como eliminado en la base de datos el elemento que coincide con el identificador de entrada.

3.1.5. IDependenceGradesTypes

IEnumerable<DependenceGradeTypes> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de tipos de grados de dependencia de la base de datos.

Guid AddDependenceGradeType(string DependenceGradeType)

- **Necesita:** un string con el texto del tipo de grado de dependencia a añadir.
- **Produce:** inserta un tipo de grado de dependencia en la base de datos con el texto de entrada.

void UpdateDependenceGradeType(Guid TypeId, string TypeName)

- **Necesita:** el identificador del tipo de grad de dependencia modificar y un string con el nuevo texto.
- **Produce:** actualiza el elemento que coincide con el identificador de entrada con el text de entrada.

void DeleteDependenceGradeType(Guid TypeId)

- **Necesita:** el identificador del tipo de grado de dependencia a elimina.
- **Produce:** marca como eliminado en la base de datos el elemento que coincide con el identificador de entrada.

3.1.6. IDisabilityTypes

IEnumerable<DisabilityTypes> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de tipos de discapacidad de la base de datos.

Guid AddDisabilityType(string DisabilityType)

- **Necesita:** un string con el texto del tipo de discapacidad a agregar.
- **Produce:** inserta en la base de datos un tipo de discapacidad con el texto de entrada.

void UpdateDisabilityType(Guid TypeId, string TypeName)

- **Necesita:** el identificador del tipo de discapacidad a modificar y un string con el nuevo texto.
- **Produce:** actualiza el elemento que coincide con el identificador de entrada con el texto de entrada.

void DeleteDisabilityType(Guid TypeId)

- **Necesita:** el idenificador del tipo de discapacidad a eliminar.
- **Produce:** marca como eliminado el elemento que coincide con el identificador de entrada en la base de datos.

3.1.7. IEntityTypes

IEnumerable<EntityTypes> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de tipos de entidad de la base de datos.

IEnumerable<EntityTypes> GetDropDownItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de tipos de entidad de la base de datos ordenada por el nombre.

Guid AddEntityType(string EntityType)

- **Necesita:** un string con el texto del tipo de entidad a añadir.
- **Produce:** inserta en la base de datos un tipo de entidad con el texto de entrada.

void UpdateEntityType(Guid TypeId, string TypeName)

- **Necesita:** el identificador del tipo de entidad a modificar.
- **Produce:** actualiza el elemento que coincide con el identificador de entrada con el texto de entrada.

void DeleteEntityType(Guid TypeId)

- **Necesita:** el identificador del tipo de entidad a eliminar.
- **Produce:** marca como eliminado en la base de datos el elemento que coincide con el identificador de entrada.

3.1.8. IFunctionalLimitationTypes

IEnumerable<FunctionalLimitationsTypes> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de tipos de limitaciones funcionales de la base de datos.

Guid AddFunctionalLimitationType(string FunctionalStatusType)

- **Necesita:** un string con el texto del tipo de limitación funcional a añadir.
- **Produce:** inserta en la base de datos un tipo de limitación funcional con el texto de entrada.

void UpdateFunctionalLimitationType(Guid TypeId, string FunctionalLimitationTypeName)

- **Necesita:** el identificador del tipo de limitación funcional a modificar y un string con el nuevo texto.

- **Produce:** actualiza el elemento que coincide con el identificador de entrada con el texto de entrada.

void DeleteFunctionalLimitationType(Guid TypeId)

- **Necesita:** el identificador del tipo de limitación funcional a eliminar.
- **Produce:** marca como eliminado en la base de datos el elemento que coincide con el identificador de entrada.

3.1.9. IInterventionDemands

IEnumerable<InterventionDemands> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de demandas de las intervenciones de la base de datos.

void AddInterventionDemands(Guid DemandId, Guid InterventionId)

- **Necesita:** el identificador del tipo de demanda y el identificador de la intervención.
- **Produce:** inserta en la tabla de demandas de las intervenciones un elemento que relaciona los identificadores de entrada.

Boolean ExistsDemand(Guid DemandTypeId, Guid InterventionId)

- **Necesita:** el identificador del tipo de demanda a comprobar y el identificador de la intervención.
- **Produce:** un booleano que indica si la intervención de entrada es del tipo de demanda de entrada.

void DeleteInterventionDemand(Guid DemandId, Guid InterventionId)

- **Necesita:** el identificador de tipo de demanda y el identificador de intervención del elemento a eliminar.
- **Produce:** marca como eliminado en la tabla de demandas de intervenciones el elemento que tenga como clave ambos identificadores.

3.1.10. IInterventions

IEnumerable<Interventions> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla intervenciones de la base de datos.

Interventions GetIntervention(Guid InterventionId)

- **Necesita:** el identificador de la intervención.
- **Produce:** el elemento de la base de datos que coincide con el identificador de entrada.

Guid AddIntervention (Interventions intervention)

- **Necesita:** un elemento de tipo intervencion.
- **Produce:** inserta en la base de datos la intervencion de entrada.

void UpdateIntervention(Interventions intervention)

- **Necesita:** un elemento de tipo intervención con un identificador existente.
- **Produce:** modifica el elemento que tenga un identificador coincidente con el elemento de entrada, sobreecribiendo dicho elemento con los datos de entrada.

List<Interventions> ListIntervention(Guid RecordId)

- **Necesita:** el identificador de un expediente.
- **Produce:** una lista de intervenciones que pertenezcan al expediente con el identificador de entrada.

void DeleteIntervention(Guid InterventionId)

- **Necesita:** el identificador de la intervención a eliminar.
- **Produce:** marca como eliminado el elemento que coincide con el identificador de entrada.

3.1.11. IJobTypes

IEnumerable<JobTypes> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de tipos de puestos de la base de datos.

IEnumerable<JobTypes> GetDropDownItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de tipos de puestos de la base de datos ordenada por nombre.

Guid AddJobType(string JobType)

- **Necesita:** un string con el texto del nuevo tipo de puesto a añadir.
- **Produce:** inserta en la base de datos un tipo de puesto con el texto de entrada.

void UpdateJobType(Guid TypeId, string TypeName)

- **Necesita:** el identificador del tipo de puesto a modificar y un string con el nuevo texto.
- **Produce:** actualiza el elemento que coincide con el identificador de entrada con el texto de entrada.

void DeleteJobType(Guid TypeId)

- **Necesita:** el identificador del tipo de puesto a eliminar.
- **Produce:** marca como eliminado en la base de datos el elemento que coincide con el identificador de entrada.

3.1.12. IRecipientDisabilities

IEnumerable<RecipientDisabilities> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de discapacidades de los beneficiarios de la base de datos.

void AddRecipientDisability(Guid DisabilityTypeId, Guid RefRcipientId)

- **Necesita:** el identificador del tipo de discapacidad y el identificador del beneficiario.
- **Produce:** inserta en la tabla de discapacidades de beneficiarios un elemento relacionando los identificadores de entrada.

RecipientDisabilities GetRecipientDisability(Guid RefRecipientId)

- **Necesita:** el identificador de un beneficiario.
- **Produce:** las discapacidades que tiene el beneficiario con el identificador de entrada.

RecipientDisabilities HasDisability(Guid refRecipientId, Guid DisabilityTypeId)

- **Necesita:** el identificador de un beneficiario y el identificador de la discapacidad a comprobar.
- **Produce:** los detalles de la discapacidad del beneficiario en caso de que la tenga.

3.1.13. IRecipientFunctionalLimitations

IEnumerable<RecipientFunctionalLimitations> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de limitaciones funcionales de los beneficiarios de la base de datos.

void AddRecipientFunctionalLimitation(Guid RefRecipientId, Guid FunctionalLimitationTypeId, string RFSDescription)

- **Necesita:** el identificador del beneficiario, el identificador del tipo de limitación funcional y un string con la descripción de la limitación funcional.
- **Produce:** inserta en la tabla de limitaciones funcionales de beneficarios un elemento con los datos de entrada.

List<RecipientFunctionalLimitations> GetRecipientFunctionalLimitation(Guid RefRecipientId)

- **Necesita:** el identificador de un beneficiario.
- **Produce:** las limitaciones funcionales del beneficiario con el identificador de entrada.

3.1.14. IRecipientRelations

IEnumerable<RecipientRelations> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de los tipos de relaciones con los beneficiarios de la base de datos.

Guid AddRecipientRelation(string RecipientRelationName)

- **Necesita:** un string con el texto del nuevo tipo de relación con el beneficiario a añadir.
- **Produce:** inserta en la base de datos un tipo de relación con el beneficiario con el texto de entrada.

void UpdateRecipientRelation(Guid TypeId, string RecipientRelationName)

- **Necesita:** el identificador de un tipo de relación con el beneficiario y un string con el nuevo texto.
- **Produce:** actualiza el elemento con el identificador de entrada con el texto de entrada.

void DeleteRecipientRelation(Guid TypeId)

- **Necesita:** el identificador con el tipo de relación con el beneficiario a eliminar.
- **Produce:** marca como eliminado el tipo de relación con el beneficiario con el identificador de entrada.

3.1.15. IRecipients

IEnumerable<Recipients> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de beneficiarios de la base de datos.

Recipients GetRecipient(Guid? RecipientId)

- **Necesita:** el identificador de un beneficario.
- **Produce:** el beneficiario con el identificador de entrada.

Guid AddRecipient(Recipients recipient)

- **Necesita:** un estructura beneficiario a insertar.
- **Produce:** inserta en la base de datos el beneficario de entrada.

void UpdateRecipient(Recipients recipient)

- **Necesita:** una estructura beneficiario con un identificador existente en la base de datos.
- **Produce:** actualiza el elemento con el identificador del beneficiario de entrada sobreescribiendo los datos de entrada.

void DeleteRecipient(Guid RecipientId)

- **Necesita:** el identificador de un beneficiario a eliminar.
- **Produce:** marca como eliminado en la base de datos el beneficiario con el identificador de entrada.

List<Recipients> ListRecipients(string RecipientString)

- **Necesita:** un string con la subcadena a buscar.
- **Produce:** una lista de beneficiarios que contienen el texto de entrada en el nombre.

IEnumerable<Recipients> Search(string SortOrder, bool AscOrder, int? Page)

- **Necesita:** un string que indica el campo a ordenar, un booleano que indica si el orden es ascendente o descendente y un entero que indica el número de elementos por página.
- **Produce:** una lista enumerable de beneficiarios ordenada según los parámetros de entrada.

IEnumerable<Recipients> Search(string SortOrder, bool AscOrder, int? Page, string LastName, string Name, string RecipientIdentifier, string City)

- **Necesita:** un string que indica el campo por el que se ordenará la lista devuelta, un booleano que indica si el orden es ascendente o descendente, un entero que índica el número de elementos mostrados por página y tres strings, de los cuales solo uno de ellos tendrá valor e índica el campo y subcadena mediante la cual se buscarán los elementos.
- **Produce:** una lista enumerable de beneficiarios ordenada según los parámetros de entrada y con los elementos que contengan en el campo indicado del elemento la cadena introducida.

3.1.16. IRecipientTypes

IEnumerable<RecipientType> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de tipos beneficiarios de la base de datos.

Guid AddRecipientType(string RecipientType)

- **Necesita:** un string con el texto del tipo de beneficiarios a insertar.
- **Produce:** inserta un tipo de beneficiario con el texto de entrada.

void UpdateRecipientType(Guid TypeId, string TypeName)

- **Necesita:** el identificador del tipo de beneficiario a identificar y un string con el nuevo texto.
- **Produce:** actualiza el tipo de beneficiario con el identificador de entrada con el texto de entrada.

void DeleteRecipientType(Guid TypeId)

- **Necesita:** el identificador del tipo de beneficiario a eliminar.
- **Produce:** marca como eliminado el tipo de beneficiario con el identificador de entrada.

3.1.17. IRecords

IEnumerable<Records> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de expedientes de la base de datos.

Records GetRecord(Guid RecordId)

- **Necesita:** el identificador de un expediente.
- **Produce:** el expediente con el identificador de entrada.

Guid AddRecord(Records record)

- **Necesita:** una estructura expediente a insertar.
- **Produce:** inserta en la base de datos el expediente de entrada.

List<Records> ListRecords(string RecordCode)

- **Necesita:** un string con la subcadena a buscar.
- **Produce:** una lista de elementos cuyo codigo contiene la cadena de entrada.

IEnumerable<Records> Search(string sortOrder, bool ascOrder, int? page)

- **Necesita:** un string que indica el campo a ordenar, un booleano que indica si el orden es ascendente o descendente y un entero que indica el número de elementos por página.
- **Produce:** una lista enumerable de expedientes ordenada según los parámetros de entrada.

IEnumerable<Records> Search(string sortOrder, bool ascOrder, int? page,int state, Guid? jobType, string recordCode, string requestCode, string applicant, string recipient, Guid? technician, string city)

- **Necesita:** un string que indica el campo por el que se ordenará la lista devuelta, un booleano que indica si el orden es ascendente o descendente, un entero que índica el número de elementos mostrados por página y dos identificadores y cinco strings, de los cuales solo uno de ellos tendrá valor e índica el campo y subcadena mediante la cual se buscarán los elementos.
- **Produce:** una lista enumerable de expedientes ordenada según los parámetros de entrada y con los elementos que contengan en el campo indicado del elemento la cadena introducida.

void UpdateRecord(Guid recordId, Guid refRecordTechnician, Guid refRecordJob, int recordState)

- **Necesita:** un identificador de expediente, un identificador de técnico del expdiente, un identificador con el puesto del técnico y un entero que indica el estado del expediente.
- **Produce:** actualiza el expediente con el identificador de entrada con los datos de entrada.

3.1.18. IRequestDemand

IEnumerable<RequestDemands> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de demandas de las solicitudes de la base de datos.

void AddRequestDemand(Guid RefDemandId, Guid RefRequestId)

- **Necesita:** un identificador con el tipo de demanda y un indentificador de la solicitud.

- **Produce:** inserta un elemento en la tabla de demandas de solicitudes relacionando los identificadores de entrada.

RequestDemands GetRequestDemand(Guid RefRequestId)

- **Necesita:** un identificador de una solicitud.
- **Produce:** las demandas de la solicitud con el identificador de entrada.

void DeleteRequestDemand(Guid RequestDemandId, Guid RequestId)

- **Necesita:** el identificador del tipo de demanda y de la solicitud a eliminar.
- **Produce:** marca como eliminado el elemento cuya clave son los identificadores de entrada.

Boolean ExistsDemand(Guid DemandType, Guid RequestId)

- **Necesita:** el identificador del tipo de demanda y de la solicitud a comprobar.
- **Produce:** los detalles de la demanda de la solicitud en caso de que sea de ese tipo de demanda.

3.1.19. IRequestDisabilities

IEnumerable<RequestDisabilities> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de discapacidades de la solicitud de la base de datos.

void AddRequestDisability(Guid DisabilityTypeId, Guid RefRequestId)

- **Necesita:** un identificador del tipo d discapacidad y un identificador de la solicitud.
- **Produce:** inserta una demanda de solicitud con los identificadores de entrada.

RequestDisabilities GetRequestDisability(Guid RefRequestId)

- **Necesita:** el identificador de una solicitud.
- **Produce:** las discapacidades de la solicitud con el identificador de entrada.

void DeleteRequestDisability(Guid RequestDemandId, Guid RequestId)

- **Necesita:** un identificador del tipo de demanda y un identificador de la solicitud.
- **Produce:** marca como eliminado el tipo de demanda de la solicitud cuya clave son los identificadores de entrada.

3.1.20. IRequestFunctionalLimitations

IEnumerable<RequestFunctionalLimitations> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de limitaciones funcionales de solicitudes de la base de datos.

void AddRequestFunctionalLimitation(Guid RefRequestId, Guid FunctionalLimitationTypeId, string RFSDescription)

- **Necesita:** un identificador de la solicitud, un identificador del tipo de limitación funcional y un string con la descripción de la limitación funcional.
- **Produce:** inserta un tipo de limitación funcional de solicitud con los identificadores como clave y con la descripción de la entrada.

List<RequestFunctionalLimitations> GetRequestFunctionalLimitation(Guid RefRequestId)

- **Necesita:** el identificador de una solicitud.
- **Produce:** una lista con las limitaciones funcionales de la solicitud con el identificador de entrada.

void DeleteRequestFunctionalLimitation(Guid FunctionalLimitationTypeId, Guid RequestId)

- **Necesita:** el identificador del tipo de limitación funcional y el de la solicitud.
- **Produce:** marca como eliminado en la base de datos el elemento que coincide con los identificadores de entrada.

3.1.21. IRequest

IEnumerable<Requests> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de solicitudes de la base de datos.

Requests GetRequest(Guid RequestId)

- **Necesita:** el identificador de una solicitud.
- **Produce:** el elemento solicitud cuyo identificador es el de entrada.

Guid AddRequest(Requests request, Recipients recipient, Applicants applicant)

- **Necesita:** una estructura de tipo solicitud, una estructura de tipo beneficiario y una estructura de tipo solicitante.
- **Produce:** inserta una solicitud enla base de datos con los elementos de entrada.

Guid AddRequest(Requests request, Recipients recipient, Applicants applicant, IEnumerable<RequestFunctionalLimitations> funtionalLimitations, IEnumerable<DemandTypes> demands)

- **Necesita:** una estructura de tipo solicitud, una estructura de tipo beneficiario, una estructura de tipo solicitante, un lista de limitaciones funcionales de la solicitud y una lista de tipos de demandas de la solicitud.
- **Produce:** inserta una solicitud enla base de datos con los elementos de entrada, agregando las limitaciones funcionales y los tipos de demanda en las respectivas tablas.

List<Requests> ListRequests(Guid RecordId)
DataTable GetAllItemsDT(string city, string applicant, string recipient, DateTime? dateSince, DateTime? dateUntil)

- **Necesita:** tres strings, de los cuales solo uno tiene valor, que indican la ciudad, solicitante o beneficiario a buscar y dos fechas que indican el intervalo a buscar.
- **Produce:** una lista de solicitudes que contiene la ciudad/solicitante/beneficiario de entrada y cuyas fechas estan dentro del intervalo indicado.

IEnumerable<Requests> FilterRequests(IEnumerable<Requests> listRequest, string recipient, string applicant, string city, DateTime? dateSince = null, DateTime? dateUntil = null)

- **Necesita:** una lista de solicitudes, tres strings, de los cuales solo uno tiene valor, que indican la ciudad, solicitante o beneficiario a buscar y dos fechas que indican el intervalo a buscar.
- **Produce:** un subconjunto (sublista) de la lista de solicitudes de entrada que cotiene la ciudad/solicitante/beneficiario de entrada y cuyas fechas estan dentro del intervalo indicado.

IEnumerable<Requests> OrderRequest(IEnumerable<Requests> listRequest,string stringOrder)

- **Necesita:** una lista de solicitudes y un string que indica el campo de ordenacion.
- **Produce:** la lista de solicitudes de entrada ordenada por el campo indicado en la entrada.

IEnumerable<Requests> Search(string sortOrder,bool ascOrder, int? page)

- **Necesita:** un string que indica el campo de ordenacion, un booleano que indica orden ascendente o descendente y un entero que indica el número de elementos por página.
- **Produce:** una lista de solicitudes con todos los elmentos de la tabla ordenada según los parámetros indicados.

IEnumerable<Requests> Search(string sortOrder, bool ascOrder,int? page, DateTime? dateSince, DateTime? dateUntil, string Applicant, string Recipient, string City)

- **Necesita:** un string que indica el campo de ordenacion, un booleano que indica orden ascendente o descendente, un entero que indica el número de elementos por página, dos fechas que indican el intervalo de búsqueda y tres strings, solo uno de los cuales con valor, que indica la cadena a buscar y el campo.
- **Produce:** una lista de solicitudes que contiene la ciudad/solicitante/beneficiario de entrada y cuyas fechas estan dentro del intervalo indicado y ordenado según los parámetros de entrada.

Boolean ExistsRequestRecipient(string Recipient)

- **Necesita:** un string que indica el nombre del beneficiario.
- **Produce:** un booleano que indica si algún beneficiario que contenga la cadena de entrada tiene solicitudes asociadas.

void DeleteRequest(Guid RequestId)

- **Necesita:** un identificador de la solicitud a eliminar.
- **Produce:** marca como eliminada la solicitud con el identificador de entrada.

void UpdateRequest(Requests request, Applicants applicant, Recipients recipient)

- **Necesita:** una estructura de tipo solicitud, una estructura de tipo solicitante y una estructura de tipo beneficiario.
- **Produce:** actualiza la solicitud con el mismo identificador que la solicitud de entrda con los datos de entrada.

void UpdateRecordRequest(Guid RecordId, Guid RequestId)

- **Necesita:** un identificador de expediente y un identificador de solicitud.
- **Produce:** actualiza el identificador de expediente de la solicitud con el idenificador de entrada.

3.1.22. IRoles

IEnumerable<webpages_Roles> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de roles de usuario de la base de datos.

void AddUserToRole(string UserName, int RoleId)

- **Necesita:** una cadena que indica el nombre de usuario y un identificador de un tipo de rol.
- **Produce:** agrega el usuario con el nombre de entrada a el rol con el identificador de entrada.

void DeleteUserFromRoles(string UserName)

- **Necesita:** una cadena que indica el nombre.
- **Produce:** elimina el usuario con el nombre de entrada de la tabla de roles.

3.1.23. ISexTypes

IEnumerable<SexTypes> GetAllItems()

- **Necesita:** una lista enumerable con todos los elementos de la tabla de tipos de género de la base de datos.

Guid AddSexType(string JobType)

- **Necesita:** un string con el texto del nuevo tipo de género.
- **Produce:** inserta en la base de datos un nuevo género con el texto de entrada.

void UpdateSexType(Guid TypeId, string TypeName)

- **Necesita:** un identificador del tipo de género a modificar y un string con el nuevo texto.
- **Produce:** actualiza el tipo de género con el identificador de entrada con el texto de entrada.

void DeleteSexType(Guid TypeId)

- **Necesita:** un identificador del tipo de género a eliminar.
- **Produce:** marca como eliminado el tipo de género con el identificador de entrada.

3.1.24. ISourceTypes

IEnumerable<SourceTypes> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de tipos de entrada de la base de datos.

IEnumerable<SourceTypes> GetDropDownItems()

- **Produce:** una lista enumerable con los tipos de entrada de la base de datos ordenada por el nombre.

Guid AddSourceType(string SourceType)

- **Necesita:** un string con el texto del tipo de entrada a insertar.
- **Produce:** inserta un nuevo tipo de entrada en la base de datos con el texto de entrada.

void UpdateSourceType(Guid TypeId, string TypeName)

- **Necesita:** un identificador con el tipo de entrada a modificar y un string con el nuevo texto.
- **Produce:** actualiza el tipo de entrada con el identificador de entrada con el texto de entrada.

void DeleteSourceType(Guid TypeId)

- **Necesita:** un identificador del tipo de entrada a eliminar.
- **Produce:** marca como eliminado el tipo de entrada con el identificador de entrada.

3.1.25. IStatistics

int GetApplicantCount()

- **Produce:** el número de elementos de la tabla de solicitantes no marcados como eliminados.

int GetRecipientCount()

- **Produce:** el número de elementos de la tabla de beneficiarios no marcados como eliminados.

int GetRequestCount()

- **Produce:** el número de elementos de la tabla de solicitudes no marcados como eliminados.

int GetRecordCount()

- **Produce:** el número de elementos de la tabla de expedientes no marcados como eliminados.

int GetInterventionCount()

- **Produce:** el número de elementos de la tabla de intervenciones no marcados como eliminados.

int GetApplicantCount(DateTime startDate, DateTime endDate)

- **Necesita:** dos fechas que indican el intervalo.
- **Produce:** el número de elementos de la tabla de solicitantes no marcados como eliminados y dentro del intervalo de entrada.

int GetRecipientCount(DateTime startDate, DateTime endDate)

- **Necesita:** dos fechas que indican el intervalo.
- **Produce:** el número de elementos de la tabla de beneficiarios no marcados como eliminados y dentro del intervalo de entrada.

int GetRequestCount(DateTime startDate, DateTime endDate)

- **Necesita:** dos fechas que indican el intervalo.
- **Produce:** el número de elementos de la tabla de solicitudes no marcados como eliminados y dentro del intervalo de entrada.

int GetRecordCount(DateTime startDate, DateTime endDate)

- **Necesita:** dos fechas que indican el intervalo.
- **Produce:** el número de elementos de la tabla de expedientes no marcados como eliminados y dentro del intervalo de entrada.

int GetInterventionCount(DateTime startDate, DateTime endDate)

- **Necesita:** dos fechas que indican el intervalo.
- **Produce:** el número de elementos de la tabla de intervenciones no marcados como eliminados y dentro del intervalo de entrada.

Dictionary<string, int> GetRequestStatistics()

- **Produce:** un diccionario nombre[número] con las estadísticas de las solicitudes.

Dictionary<string, int> GetRecordStatistics()

- **Produce:** un diccionario nombre[número] con las estadísticas de los expedientes.

Dictionary<string, int> GetInterventionStatistics()

- **Produce:** un diccionario nombre[número] con las estadísticas de las intervenciones.

Dictionary<string, int> GetApplicantStatistics()

- **Produce:** un diccionario nombre[número] con las estadísticas de los solicitantes.

Dictionary<string, int> GetRecipientStatistics()

- **Produce:** un diccionario nombre[número] con las estadísticas de los beneficiarios.

int GetRecipientsBySex(int sexType, DateTime startDate, DateTime endDate)

- **Necesita:** un entero que indica el tipo de sexo y dos fechas que indican el intervalo.
- **Produce:** el número de beneficiarios del sexo indicado con solicitudes en el intervalo de entrada.

IEnumerable<Interventions> GetInterventions(Guid? demanda, Guid? actuacion, DateTime startDate, DateTime endDate)

- **Necesita:** un identificador del tipo de demanda, un identificador de actuación y dos fechas que indican el intervalo.
- **Produce:** una lista de intervenciones con el tipo de demanda y la accion indicadas en la entrada y dentro del intervalo de entrada.

IEnumerable<Records> GetRecordsBetweenDates(DateTime StartDate, DateTime EndDate)

- **Necesita:** dos fechas que indican el intervalo.
- **Produce:** una lista de los expedientes abiertos dentro del intervalo de entrada.

3.1.26. ITechnicians

IEnumerable<Technicians> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla técnicos de la base de datos.

IEnumerable<Technicians> GetDropDownItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de técnicos de la base de datos ordenada por el nombre.

Guid AddTechnician(string TechnicianName, string TecnicianLastName)

- **Necesita:** un string con el nombre y un string con los apellidos del técnico a agregar.
- **Produce:** inserta en la base de datos un técnico con el nombre y apellidos indicados en la entrada.

void DeleteTechnician(int TechnicianId)

- **Necesita:** el identificador del técnico a eliminar.
- **Produce:** marca como eliminado el técnico con el identificador de enrada.

3.1.27. IUsers

IEnumerable<UserProfile> GetAllItems()

- **Produce:** una lista enumerable con todos los elementos de la tabla de usuarios de la base de datos.

DataTable GetAllItemsDT()

- **Produce:** una tabla de datos con todos los elements de la tabla de usuarios de la base de datos.

int AddUser(string UserName, string password)

- **Necesita:** un string con el nombre de usuario y un string con el password.
- **Produce:** inserta el usuario con el nombre de usuario y password de entrada.

void DeleteUser(int UserId)

- **Necesita:** el identificador de usuario a eliminar.
- **Produce:** marca como eliminado el usuario con el identificador de entrada.

UserProfile GetUser(int UserId)

- **Necesita:** un identificador de usuario.
- **Produce:** una estructura de tipo perfil de usuario con los datos del usuario con el identificador de entrada.

3.2. Utilities

public void RequestReport(Guid ApplicantId, Guid ReccipientId, Guid RequestId, string absoluteReportPath)

- **Necesita:** un identificador de solicitante, un idenificador de beneficiario, un identificador de solicitud y un string con la ruta donde se guardará el informe.
- **Produce:** un informe word con los datos del beneficiario,solicitud y solicitante de entrada.

public static string RemoveDiacritics(this string s)

- **Necesita:** un string a normalizar.

- **Produce:** un string normalizado del string de entrada para realizar comparaciones en la base de datos.

3.3. Controlers/Actions

3.3.1. AdministrationController

public ActionResult Index()

- **Produce:** la vista de la administración de la aplicación.

[HttpPost]
public ActionResult AddDemandTypes(string DemandType, string FatherDemandId)

- **Necesita:** un string con el nuevo tipo de demanda y un string del padre en caso de tenerlo.
- **Produce:** agrega un tipo de demanda con el texto de entrada y como hijo del padre indicado (si se ha indicado padre alguno).

public ActionResult EditDemandTypes(Guid DemandTypeID)

- **Necesita:** el identificador de un tipo de demanda.
- **Produce:** la vista de modificacion del tipo de demanda de entrada.

[HttpPost]
public ActionResult EditDemandTypes(Guid DemandTypeId , string DemandTypeName)

- **Necesita:** el identificador de un tipo de demanda y un string con el nuevo texto.
- **Produce:** actualiza el tipo de demanda de entrada con el texto de entrada.

public ActionResult DeleteDemandTypes(int DemandTypeId)

- **Necesita:** un identificador con el tipo de demanda.
- **Produce:** elimina el tipo de demanda de entrada.

public ActionResult DependenceGradeTypes()

- **Produce:** la vista de administración del tipo de grados de dependencia de la base de datos.

public ActionResult AddDependenceGradeTypes()

- **Produce:** la vista para insertar un nuevo tipo de grado de dependencia.

[HttpPost]
public ActionResult AddDependenceGradeTypes(string DependenceGradeType)

- **Necesita:** un string con el texto del elemento a agregar.
- **Produce:** inserta un nuevo tipo de grado de dependencia con el texto de entrada.

public ActionResult EditDependenceGradeTypes(Guid DependenceGradeId)

- **Necesita:** el identificador del elemento a modificar.
- **Produce:** la vista de edición del tipo de grado de dependencia .

[HttpPost]
public ActionResult EditDependenceGradeTypes(Guid DependenceGradeId, string DependenceGradeTypeName)

- **Necesita:** el identificador del elemento a modificar y un string con el nuevo texto.
- **Produce:** actualiza el tipo de grado de dependencia con el texto de entrada.

public ActionResult DeleteDependenceGradeTypes(Guid DependenceGradeTypeId)

- **Necesita:** el identificador del elemento a eliminar.
- **Produce:** elimina el tipo de grado de dependencia con el identificador de entrada.

public ActionResult DisabilityTypes()

- **Produce:** la vista de administración del tipo de discapacidad de la base de datos.

public ActionResult AddDisabilityTypes()

- **Produce:** la vista para insertar un nuevo tipo de discapacidad.

[HttpPost]
public ActionResult AddDisabilityTypes(string DisabilityType)

- **Necesita:** un string con el texto del elemento a agregar.

- **Produce:** inserta un nuevo tipo de discapacidad con el texto de entrada.

public ActionResult EditDisabilityTypes(Guid DisabilityTypeId)

- **Necesita:** el identificador del elemento a modificar.
- **Produce:** la vista de edición del tipo de discapacidad .

[HttpPost]
public ActionResult EditDisabilityTypes(Guid DisabilityTypeId, string DisabilityTypeName)

- **Necesita:** el identificador del elemento a modificar y un string con el nuevo texto.
- **Produce:** actualiza el tipo de discapacidad con el texto de entrada.

public ActionResult DeleteDisabilityTypes(Guid DisabilityTypeId)

- **Necesita:** el identificador del elemento a eliminar.
- **Produce:** elimina el tipo de discapacidad con el identificador de entrada.

public ActionResult EntityTypes()

- **Produce:** la vista de administración del tipo de entidad de la base de datos.

public ActionResult AddEntityTypes()

- **Produce:** la vista para insertar un nuevo tipo de entidad.

[HttpPost]
public ActionResult AddEntityTypes(string EntityType)

- **Necesita:** un string con el texto del elemento a agregar.
- **Produce:** inserta un nuevo tipo de entidad con el texto de entrada.

public ActionResult EditEntityTypes(Guid EntityTypeId)

- **Necesita:** el identificador del elemento a modificar.
- **Produce:** la vista de edición del tipo de entidad .

[HttpPost]
public ActionResult EditEntityTypes(Guid EntityTypeId, string EntityTypeName)

- **Necesita:** el identificador del elemento a modificar y un string con el nuevo texto.
- **Produce:** actualiza el tipo de entidad con el texto de entrada.

public ActionResult DeleteEntityTypes(Guid EntityTypeId)

- **Necesita:** el identificador del elemento a eliminar.
- **Produce:** elimina el tipo de entidad con el identificador de entrada.

public ActionResult FunctionalStatusTypes()

- **Produce:** la vista de administración del tipo de situaciones funcionales de la base de datos.

public ActionResult AddFunctionalStatusTypes()

- **Produce:** la vista para insertar un nuevo tipo de situación funcional.

[HttpPost]
public ActionResult AddFunctionalStatusTypes(string FunctionalStatusType)

- **Necesita:** un string con el texto del elemento a agregar.
- **Produce:** inserta un nuevo tipo de situación funcional con el texto de entrada.

public ActionResult EditFunctionalStatusTypes(Guid FunctionalStatusTypeId)

- **Necesita:** el identificador del elemento a modificar.
- **Produce:** la vista de edición del tipo de situación funcional .

[HttpPost]
public ActionResult EditFunctionalStatusTypes(Guid FunctionalLimitationTypeId, string FunctionalLimitationTypeName)

- **Necesita:** el identificador del elemento a modificar y un string con el nuevo texto.
- **Produce:** actualiza el tipo de situación funcional con el texto de entrada.

public ActionResult DeleteFunctionalStatusTypes(Guid FunctionalStatusTypeId)

- **Necesita:** el identificador del elemento a eliminar.

- **Produce:** elimina el tipo de situación funcional con el identificador de entrada.

public ActionResult JobTypes()

- **Produce:** la vista de administración del tipo de puestos de la base de datos.

public ActionResult AddJobTypes()

- **Produce:** la vista para insertar un nuevo tipo de puesto.

[HttpPost]
public ActionResult AddJobTypes(string JobType)

- **Necesita:** un string con el texto del elemento a agregar.
- **Produce:** inserta un nuevo tipo de puesto con el texto de entrada.

public ActionResult EditJobTypes(Guid JobTypeId)

- **Necesita:** el identificador del elemento a modificar.
- **Produce:** la vista de edición del tipo de puesto.

[HttpPost]
public ActionResult EditJobTypes(Guid JobTypeId, string JobTypeName)

- **Necesita:** el identificador del elemento a modificar y un string con el nuevo texto.
- **Produce:** actualiza el tipo de puestos con el texto de entrada.

public ActionResult DeleteJobTypes(Guid JobTypeId)

- **Necesita:** el identificador del elemento a eliminar.
- **Produce:** elimina el tipo de puesto con el identificador de entrada.

public ActionResult SourceType()

- **Produce:** la vista de administración del tipo de entrada de la base de datos.

public ActionResult AddSourceType()

- **Produce:** la vista para insertar un nuevo tipo de entrada.

[HttpPost]
public ActionResult AddSourceType(string SourceType)

- **Necesita:** un string con el texto del elemento a agregar.
- **Produce:** inserta un nuevo tipo de entrada con el texto de entrada.

public ActionResult EditSourceType(Guid SourceTypeId)

- **Necesita:** el identificador del elemento a modificar.
- **Produce:** la vista de edición del tipo de entrada .

[HttpPost]
public ActionResult EditSourceType(Guid SourceTypeId, string SourceTypeName)

- **Necesita:** el identificador del elemento a modificar y un string con el nuevo texto.
- **Produce:** actualiza el tipo de entrada con el texto de entrada.

public ActionResult DeleteSourceType(Guid SourceTypeId)

- **Necesita:** el identificador del elemento a eliminar.
- **Produce:** elimina el tipo de entrada con el identificador de entrada.

public ActionResult ApplicantType()

- **Produce:** la vista con el tipo de solicitante de la base de datos.

public ActionResult AddApplicantType()

- **Produce:** la vista para insertar un nuevo tipo de solicitante.

[HttpPost]
public ActionResult AddApplicantType(string ApplicantType)

- **Necesita:** un string con el texto del elemento a agregar.
- **Produce:** inserta un nuevo tipo de solicitante con el texto de entrada.

public ActionResult EditApplicantType(Guid ApplicantTypeId)

- **Necesita:** el identificador del elemento a modificar.
- **Produce:** la vista de edición del tipo de solicitante .

[HttpPost]
public ActionResult EditApplicantType(Guid ApplicantTypeId, string ApplicantTypeName)

- **Necesita:** el identificador del elemento a modificar y un string con el nuevo texto.
- **Produce:** actualiza el tipo de solicitante con el texto de entrada.

public ActionResult DeleteApplicantType(Guid ApplicantTypeId)

- **Necesita:** el identificador del elemento a eliminar.
- **Produce:** elimina el tipo de solicitante con el identificador de entrada.

public ActionResult RecipientType()

- **Produce:** la vista de administración del tipo de beneficiario de la base de datos.

public ActionResult AddRecipientType()

- **Produce:** la vista para insertar un nuevo tipo de beneficiario.

[HttpPost]
public ActionResult AddRecipientType(string RecipientType)

- **Necesita:** un string con el texto del elemento a agregar.
- **Produce:** inserta un nuevo tipo de beneficiario con el texto de entrada.

public ActionResult EditRecipientType(Guid RecipientTypeId)

- **Necesita:** el identificador del elemento a modificar.
- **Produce:** la vista de edición del tipo de beneficiario .

[HttpPost]
public ActionResult EditRecipientType(Guid RecipientTypeId, string RecipientTypeName)

- **Necesita:** el identificador del elemento a modificar y un string con el nuevo texto.
- **Produce:** actualiza el tipo de beneficiario con el texto de entrada.

public ActionResult DeleteRecipientType(Guid RecipientTypeId)

- **Necesita:** el identificador del elemento a eliminar.
- **Produce:** elimina el tipo de beneficiario con el identificador de entrada.

public ActionResult RecipientRelations()

- **Produce:** la vista de administración del tipo de relacion con el beneficiariode la base de datos.

public ActionResult AddRecipientRelation()

- **Produce:** la vista para insertar un nuevo tipo de relación con el beneficiario.

[HttpPost]
public ActionResult AddRecipientRelation(string RecipientRelationName)

- **Necesita:** un string con el texto del elemento a agregar.
- **Produce:** inserta un nuevo tipo de relación con el beneficiario con el texto de entrada.

public ActionResult EditRecipientRelation(Guid RecipientRelationID)

- **Necesita:** el identificador del elemento a modificar.
- **Produce:** la vista de edición del tipo de beneficiario .

[HttpPost]
public ActionResult EditRecipientRelation(Guid RecipientRelationId, string RecipientRelationName)

- **Necesita:** el identificador del elemento a modificar y un string con el nuevo texto.
- **Produce:** actualiza el tipo de relación con el beneficiario con el texto de entrada.

public ActionResult DeleteRecipientRelation(int RecipientRelationId)

- **Necesita:** el identificador del elemento a eliminar.
- **Produce:** elimina el tipo de relación con el beneficiario con el identificador de entrada.

public ActionResult Users()

- **Produce:** la vista de administracion de usuarios de la base de datos.

public ActionResult AddUser()

- **Produce:** la vista para insertar un nuevo usuario.

[HttpPost]
public ActionResult AddUser(UserModel model)

- **Necesita:** un modelo de usuario del elemento a agregar.
- **Produce:** inserta un nuevo tipo de discapacidad con los datos de entrada.

public ActionResult DeleteUser(int UserId)

- **Necesita:** el identificador del elemento a eliminar.
- **Produce:** elimina el usuario con el identificador de entrada.

3.3.2. ApplicantController

public ViewResult Index()

- **Produce:** la vista de solicitantes

[HttpPost]
public ViewResult Index(ApplicantSearchModel m)

- **Necesita:** una estructura de modelo de búqueda de solicitante.
- **Produce:** una vista con los solicitantes que coinciden con el modelo de entrada.

public ActionResult Create()

- **Produce:** la vista de nuevo solicitante.

[HttpPost]
public ActionResult Create(ApplicantModel m)

- **Necesita:** una estructura de modelo de solicitante.
- **Produce:** inserta un solicitante con los datos del modelo de entrada.

public ActionResult Edit(Guid ApplicantId)

- **Necesita:** el identificador del elemento a modificar.
- **Produce:** la vista de edición del solicitante .

[HttpPost]
public ActionResult Edit(Applicants Applicant)

- **Necesita:** una estructura de tipo soliciante.
- **Produce:** actualiza el elemento que coincide con el identificador de entrada con los datos de entrada.

public ActionResult Delete(Guid ApplicantId)

- **Necesita:** el identificador del elemento a eliminar.
- **Produce:** elimina el solicitante con el identificador de entrada.

3.3.3. InterventionController

public ActionResult Index(int? recipientId, int? requestId)

- **Necesita:** un identificador de beneficiario y un identificador de solicitud.
- **Produce:** la vista de intervenciones.

[HttpPost]
public ActionResult Index(NewInterventions newInter, HttpPostedFileBase fichero)

- **Necesita:** una estructura de tipo nueva intervención y un fichero.
- **Produce:** asocia el fichero de entrada con la intervención de entrada.

public ActionResult Create(Guid RequestId)

- **Necesita:** un identificador de solicitud que usará la vista.
- **Produce:** la vista de creación de intervención.

[HttpPost]
public ActionResult Create(NewInterventions model)

- **Necesita:** una estructura de nueva intervención.
- **Produce:** inserta una nueva solicitud con los datos de entrada.

public ActionResult SelectRecipient(string RName)

- **Necesita:** un string con el nombre del beneficiario.
- **Produce:** una vista con los beneficiarios que contienen el texto de entrada.

public ActionResult Detail(Guid InterventionId)

- **Necesita:** el identificador de una intervención.
- **Produce:** una vista con los detalles de la intervención con el identificador de entrada.

public ActionResult Edit(Guid InterventionId)

- **Necesita:** el identificador del elemento a modificar.
- **Produce:** la vista de edición de la intervención .

[HttpPost]
public ActionResult Edit (NewInterventions model)

- **Necesita:** una estructura de tipo nueva intervención.
- **Produce:** actualiza la intervención con el mismo identificador que la de entrada con los datos de entrada.

public ActionResult Delete(Guid InterventionId)

- **Necesita:** el identificador de una intervención a eliminar.
- **Produce:** elimina la intervención con el identificador de entrada.

3.3.4. RecipientController

public ViewResult Index()

- **Produce:** la vista de beneficiarios.

[HttpPost]
public ViewResult Index(RecipientSearchModel m)

- **Necesita:** una estructura de modelo de búqueda de beneficiario.
- **Produce:** una vista con los beneficiarios que coinciden con el modelo de entrada.

public ActionResult Delete(Guid RecipientId)

- **Necesita:** el identificador del beneficiario a eliminar.
- **Produce:** elimina el beneficiario con el identificador de entrada.

public ActionResult Create()

- **Produce:** la vista de creación de beneficiario.

[HttpPost]
public ActionResult Create(RecipientModel model)

- **Necesita:** una estructura de modelo de beneficiario.
- **Produce:** inserta un beneficirio con los datos del modelo de entrada.

public ActionResult Edit(Guid RecipientId)

- **Necesita:** el identificador del beneficiario a editar.
- **Produce:** la vista de edicición del beneficiario de entrada.

[HttpPost]
public ActionResult Edit(RecipientModel model)

- **Necesita:** una estructura de tipo beneficiario.
- **Produce:** actualiza el elemento que coincide con el identificador de entrada con los datos de entrada.

public ActionResult Detail(Guid RecipientId)

- **Necesita:** el identificador del beneficiario a mostrar.
- **Produce:** una vista con los detalles del beneficiario de entrada.

3.3.5. RecordController

public ViewResult Index()

- **Produce:** la vista de expedientes.

[HttpPost]
public ViewResult Index(RecordSearchModel m)

- **Necesita:** una estructura de tipo modelo de búsqueda de expediente.
- **Produce:** una vista con los expedientes que coinciden con el modelo de entrada.

public Guid addRecord(Records rec)

- **Necesita:** una estructura de tipo expediente a añadir.
- **Produce:** inserta un expediente con los datos de entrada.

public ActionResult Detail(Guid RecordId)

- **Necesita:** un identificador de expediente a mostrar.
- **Produce:** muestra los detalles del expediente con el identificador de entrada.

[HttpPost]
public ActionResult Detail(RecordModel rm)

- **Necesita:** una estructura de tipo modelo de expediente
- **Produce:** muestra los detalles del expediente de entrada agregando docuemtos adicionales si los hubiera.

public ActionResult Edit(Guid RecordId)

- **Necesita:** un identificador de un expediente a editar.
- **Produce:** la vista de edición del expediente con el identificador de entrada.

[HttpPost]
public ActionResult Edit(RecordUpdateModel m)

- **Necesita:** una estructura de tipo modelo de actualización de expediente.
- **Produce:** actualiza el expediente que coincide con el de entrada con los datos de entrada.

[HttpPost]
public JsonResult DeleteRecordDocument(Guid fileId)

- **Necesita:** un identificador de un fichero a eliminar.
- **Produce:** elimina el fichero con el identificador de entrada.

public JsonResult GetApplicantRecord(string term)

- **Necesita:** un string con el nombre del solicitante.

- **Produce:** muestra los expedientes del solicitante con el nombre de entrada a través del objeto JSon.

public JsonResult GetRecipientRecord(string term)

- **Necesita:** un string con el nombre del beneficiario.
- **Produce:** muestra el expediente del beneficiario con el nombre de entrada a través del objeto JSon.

3.3.6. RequestController

public ViewResult Index()

- **Produce:** la vista de solicitudes.

[HttpPost]
public ViewResult Index(RequestSearchModel m)

- **Necesita:** una estructura de tipo de modelo de búsqueda de solicitud.
- **Produce:** una vista con la lista de solicitudes que coinciden con el modelo de entrada.

public ActionResult Create(Guid? AppId = null, Guid? RecId = null)

- **Necesita:** un identificador de solicitante y un identificador de expediente a utiliar por la vista.
- **Produce:** la pantalla de creación de solicitudes.

[HttpPost]
public ActionResult Create(RequestModel newRequest)

- **Necesita:** una estructura de tipo modelo de solicitud a agregar.
- **Produce:** inserta una solicitud con los datos de entrada.

public ActionResult SearchRequest(string City, string Applicant, string Recipient, DateTime? DateSince, DateTime? DateUntil)

- **Necesita:** tres strings, uno de los cuales tendrá valor, que indican el texto y campo a buscar y dos fechas que indican el intervalo.
- **Produce:** una vista con las solicitudes que coinciden los los parámetros de búsqueda de la entrada en el intervalo de entrada.

public ActionResult SearchApplicant(string AName, string Rid)

- **Necesita:** un string con el nombre del solicitante y un identificador de beneficiario a utilizar por la vista.
- **Produce:** una vista con los solicitantes que contienen el texto de entrada.

public ActionResult SearchRecipient(string RName, string Aid)

- **Necesita:** un string con el nombre del beneficiario y un identificador de solicitante a utilizar por la vista.
- **Produce:** una vista con los beneficiarios que contienen el texto de entrada.

public JsonResult GetApplicantProperties(Guid id)

- **Necesita:** un identificador del solicitante.
- **Produce:** un objeto JSon con las propiedades del solicitante.

public JsonResult GetRecipientProperties(Guid id)

- **Necesita:** un identificador del beneficiario.
- **Produce:** un objeto JSon con las propiedades del beneficiario.

public ActionResult DetailRequest(Guid RequestId)

- **Necesita:** un identificador de solicitud.
- **Produce:** un objeto JSon con los detalles de la solicitud.

public ActionResult Edit(Guid RequestId)

- **Necesita:** el identificador de la solicitud a editar.
- **Produce:** la vista de edición de la solicitud.

[HttpPost]
public ActionResult Edit(RequestModel model)

- **Necesita:** una estructura de tipo modelo de solicitud a modificar.
- **Produce:** actualiza la solicitud que tiene el mismo identificador de la entrada con los datos de entrada.

public ActionResult Delete(Guid RequestId)

- **Necesita:** el identificador de la solicitud a eliminar.
- **Produce:** eliminar la solicitud con el identificador de entrada.

public ActionResult Detail(Guid RequestId)

- **Necesita:** el identificador de la solicitud a mostrar.
- **Produce:** muestra los detalles de la solicitud con el identificador de entrada.

public JsonResult GetIdentifiers(string term)

- **Necesita:** un string con el código de una solicitud.
- **Produce:** un objeto JSon con los identificadores de solicitud, beneficiario y solicitate de la solicitud.

public JsonResult GetApplicantCity(string term)

- **Necesita:** un string con el texto a buscar.
- **Produce:** un objeto JSon con los solicitantes cuya población contenga el texto de entrada.

public JsonResult GetApplicantRegion(string term)

- **Necesita:** un string con el texto a buscar.
- **Produce:** un objeto JSon con los solicitantes cuya provincia contenga el texto de entrada.

public JsonResult GetApplicantName(string term)

- **Necesita:** un string con el texto a buscar.
- **Produce:** un objeto JSon con los solicitantes cuyo nombre contenga el texto de entrada.

public JsonResult GetApplicantLastName(string term)

- **Necesita:** un string con el texto a buscar.
- **Produce:** un objeto JSon con los solicitantes cuyos apellidos contengan el texto de entrada.

public JsonResult GetRecipientName(string term)

- **Necesita:** un string con el texto a buscar.
- **Produce:** un objeto JSon con los beneficiarios cuyo nombre contenga el texto de entrada.

public JsonResult GetRecipientLastName(string term)

- **Necesita:** un string con el texto a buscar.
- **Produce:** un objeto JSon con los beneficiarios cuyos apellidos contengan el texto de entrada.

3.3.7. SolicitudesController

public ViewResult Index(string sortOrder,string City, string Applicant,string Recipient,DateTime? DateSince=null, DateTime? DateUntil=null, RequestIndexModel model=null)

- **Necesita:** tres strings, uno de los cuales tendrá valor, que indican el campo y el texto a buscar, dos fechas que indican el intervalo a buscar y una estructura de tipo modelo de solicitud.
- **Produce:** una vista con las solicitudes que coincidan con los parámetros de entrada y estén en el intervalo de entrada.

public ActionResult Create(Guid? AppId = null, Guid? RecId = null)

- **Necesita:** un identificador de solicitante y un identificador de expediente.
- **Produce:** la vista de creación de solicitud.

[HttpPost]
public ActionResult Create(RequestModel newRequest)

- **Necesita:** una estructura de tipo modelo de solicitud insertar.
- **Produce:** inserta una solicitud con los datos de entrada.

public ActionResult SearchRequest(string City, string Applicant, string Recipient, DateTime? DateSince, DateTime? DateUntil)

- **Necesita:** tres strings que indican el campo y el texto a buscar y dos fechas que indican el intervalo de búsqueda.
- **Produce:** una vista con las solicitudes que coinciden con los parámetros de entrada y en el intervalo de entrada.

public ActionResult SearchApplicant(string AName, string Rid)

- **Necesita:** un string con el nombre del solicitante y un identificador de beneficiario a utilizar por la vista.
- **Produce:** una vista con los solicitantes que contienen el texto de entrada.

public ActionResult SearchRecipient(string RName, string Aid)

- **Necesita:** un string con el nombre del beneficiario y un identificador de solicitante a utilizar por la vista.
- **Produce:** una vista con los beneficiarios que contienen el texto de entrada.

public JsonResult GetApplicantProperties(Guid id)

- **Necesita:** un identificador del solicitante.
- **Produce:** un objeto JSon con las propiedades del solicitante.

public JsonResult GetRecipientProperties(Guid id)

- **Necesita:** un identificador del beneficiario.
- **Produce:** un objeto JSon con las propiedades del beneficiario.

public ActionResult DetailRequest(Guid RequestId)

- **Necesita:** el identificador de una solicitud a mostrar.
- **Produce:** muestra los detalles de la solicitud con el identificador de entrada.

3.3.8. StatisticsController

public ActionResult Index()

- **Produce:** la vista de estadísticas.

public ActionResult ByRecipient()

- **Produce:** la vista de estadísticas por beneficiario.

[HttpPost]
public ActionResult ByRecipient(StatisticsByRecipientModel m)

- **Necesita:** una estructura de tipo modelo estadísticas por beneficiario.
- **Produce:** muestra las estadísticas según beneficiarios.

public ActionResult ByInterventions()

- **Produce:** la vista de estadísticas por intervención.

[HttpPost]
public ActionResult ByInterventions(StatisticsByInterventionsModel m)

- **Necesita:** una estructura de tipo modelo estadísticas por intervención.
- **Produce:** muestra las estadísticas según intervenciones.

public ActionResult ByRecord()

- **Produce:** la vista de estadísticas por expedientes.

[HttpPost]
public ActionResult ByRecord(StatisticsByRecordModel m)

- **Necesita:** una estructura de tipo modelo estadísticas por expediente.
- **Produce:** muestra las estadísticas según los expedientes.

[HttpPost]
public ActionResult RecordsBetweenDates(StatisticsRecordsBetweenDatesModel m)

- **Necesita:** una estructura de tipo modelo de expedientes entre fechas.
- **Produce:** muestra las estadísticas según los expedientes entre dos fechas.

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

HERRAMIENTA DE GESTIÓN DE EXPEDIENTES PARA EL CAT

DOCUMENTO Nº VI

MANUAL DE USUARIO

ZEUS PÉREZ GARCÍA
JUNIO 2012

ÁREA DE TELEMÁTICA
TUTOR: XICU XABIEL GARCÍA PAÑEDA

ÍNDICE DEL MANUAL DE USUARIO

ÍNDICE DE CONTENIDO

ÍNDICE DE FÍGURAS DEL MANUAL DE USUARIO

MANUAL DE USUARIO

1. Introducción

La aplicación desarrollada tiene como objetivo la gestión de un servicio de expedientes perteneciente a un centro de atención y ayudas técnicas de servicios sociales. La funciones principales son el registro de solicitudes, solicitantes y expedientes, gestionando los expedientes asociados a las anteriores y permitiendo el registro y gestión documental de las intervenciones relativas a dichos expedientes.

2. Requerimientos de la aplicación

2.1. Software

Para la utilización de la aplicación web de Gestión de Expedientes simplemente es necesario tener instalado un navegador web, Microsoft Internet Explorer 5.0 o superior u otro navegador que tenga prestaciones similares a este para la correcta visualización y utilización de la aplicación.

2.2. Hardware

Para la utilización de la aplicación será necesario disponer de un pc de propósito general que soporte el software necesario para acceder a la aplicación y especificado en el apartado anterior. Además será necesario disponer de una conexión a internet.

3. Acceso a la aplicación

Para acceder a la aplicación será necesario disponer de una dirección web de para el acceso a la aplicación vía internet. Esta dirección dependerá del servidor en el que se encuentre instalado y de las opciones de puesta en funcionamiento de la misma y por lo tanto dicha dirección debe ser proporcionada por el administrador de la aplicación.

Además será necesario disponer de un nombre de usuario y contraseña para el acceso seguro a la aplicación, que también será proporcionado por el administrador de la aplicación y se introducirá en la pantalla de identificación de la aplicación para poder acceder a las funciones de la misma.

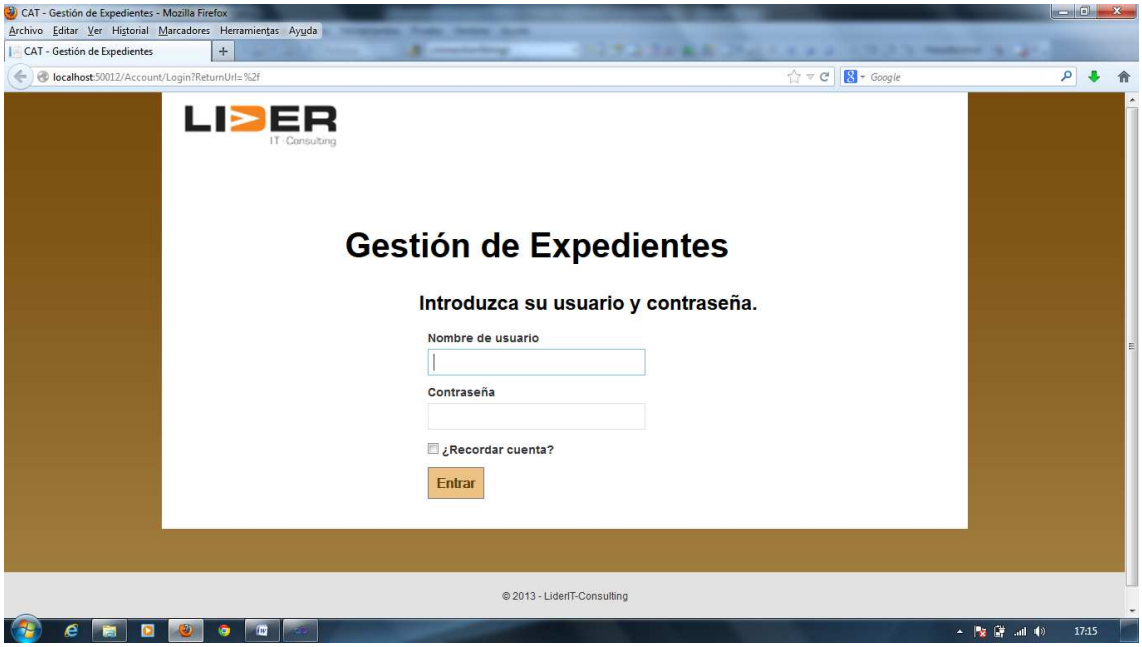


Figura 1. Pantalla de identificación

4. Uso de la aplicación

Una vez se ha realizado la identificación correctamente se podrá acceder a las funciones de la aplicación a través de la pantalla de inicio:

Figura 2. Pantalla de inicio

4.1. Nueva solicitud

Para realizar una nueva solicitud, se accederá al formulario a través del icono de crear solicitud :

Figura 3. Pantalla crear solicitud

Acto seguido se mostrará el formulario de nueva solicitud, con distintas secciones que se pueden desplegar seleccionando las pestañas correspondientes:

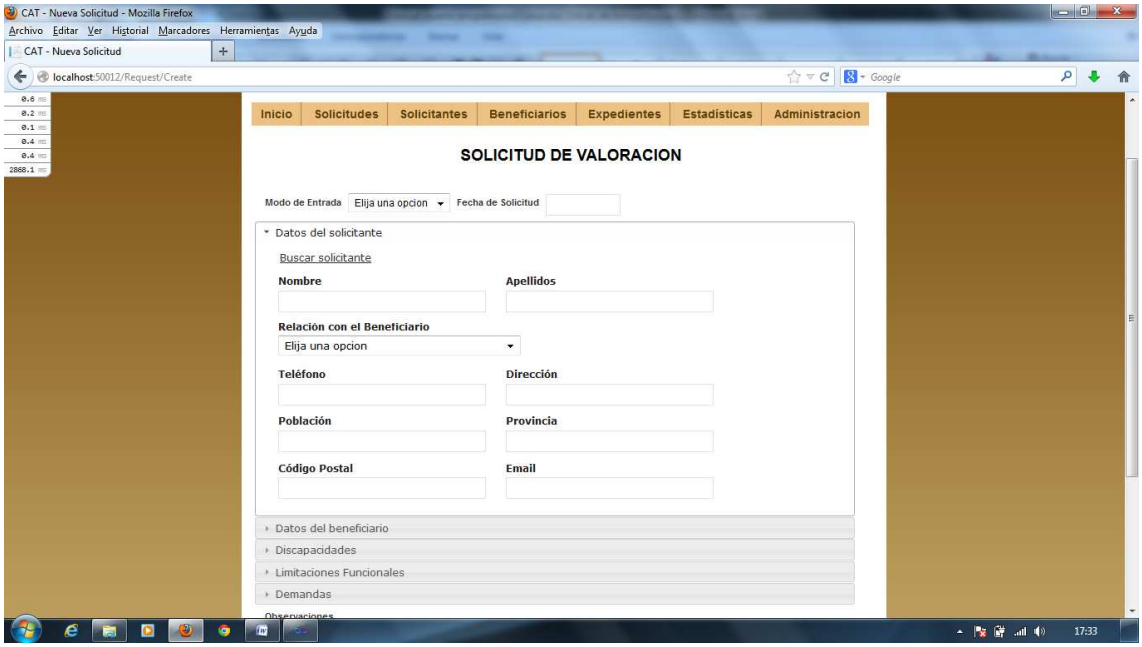


Figura 4. Formulario nueva solicitud

Una vez rellenados todos los campos necesarios se cliqueará en el botón "Crear solicitud" y la solicitud será registrada en el sistema, creándose a su vez un beneficiario, solicitante y expediente en caso de ser necesario.

También se puede crear un beneficiario o solicitante sin crear una solicitud, aunque la opción más aconsejable será crearlos a la vez que se crea la primera solicitud en la que están presentes. Dicha función de crear solicitantes o beneficiarios estará disponible a través de los iconos de crear solicitante o beneficiario en la página de inicio:

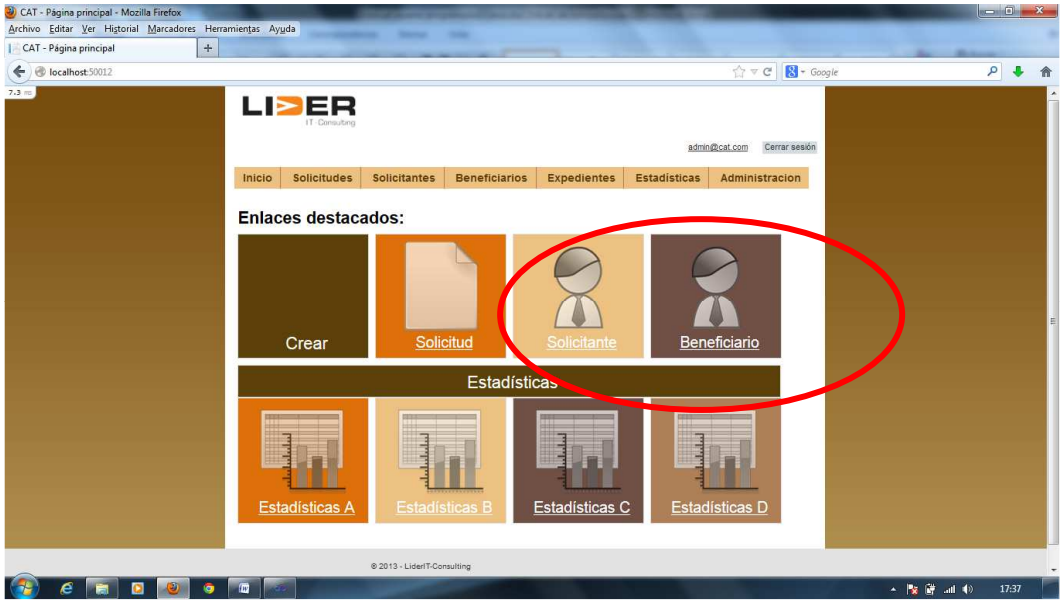


Figura 5. Crear solicitante/beneficiario

4.2. Consulta de información

Para consultar datos existentes en el sistema, se usará la barra de acceso situada en la parte superior de la página de inicio:

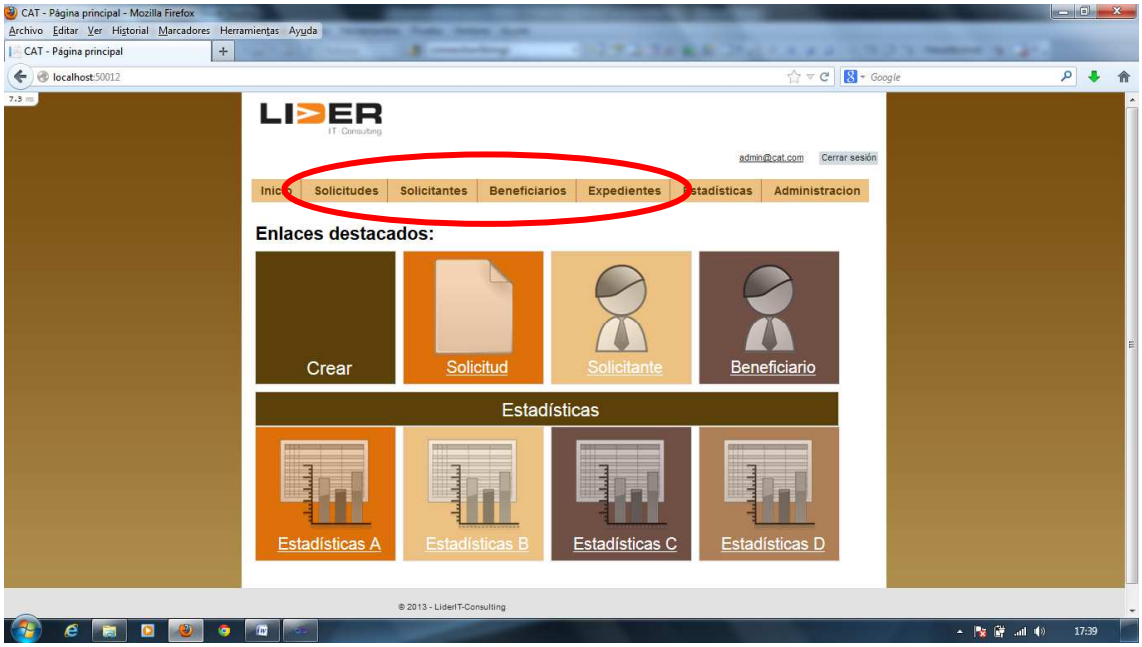


Figura 6. Barra superior

De esta manera se puede acceder a la consulta de solicitudes, expedientes, beneficiarios o solicitantes. Una vez en la pantalla de consulta, se permite introducir información sobre la búsqueda para acceder más fácil a la información deseada:

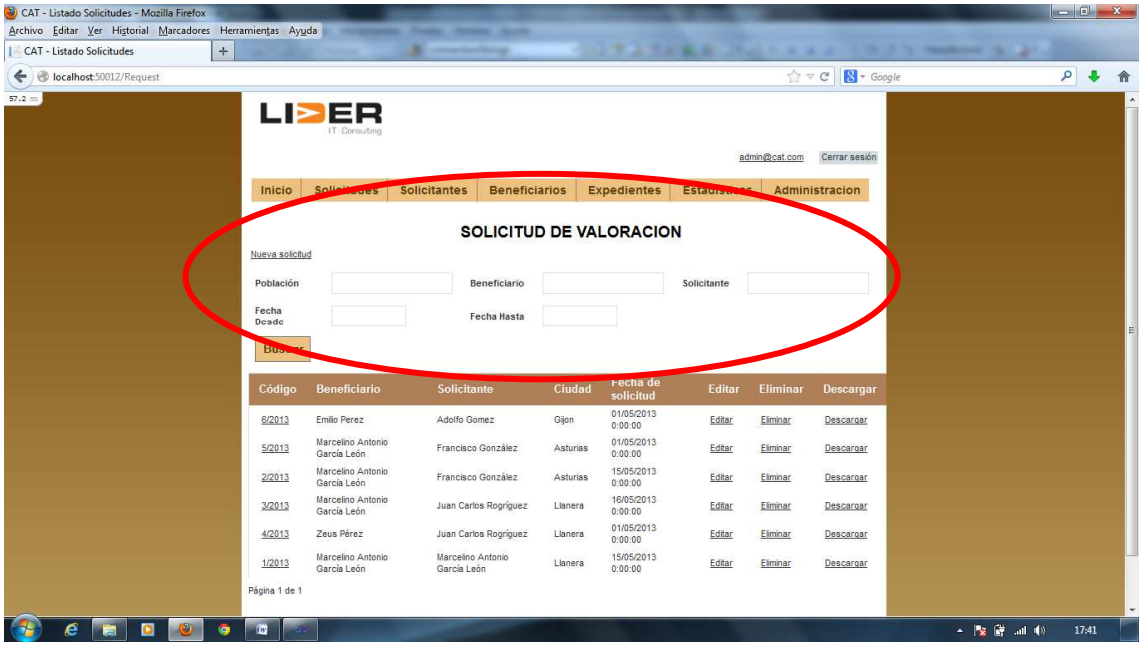


Figura 7. Listado de solicitudes

Una vez introducidos los criterios de búsqueda se pulsará el botón "Buscar" y seguidamente aparecerán los elementos que cumplen los criterios de búsqueda.

Además a través de la pantalla de consulta se pueden modificar ciertos datos. Para ello se accede a la pantalla de edición a través del botón "Editar" y se cambiará en el formulario los datos a modificar:

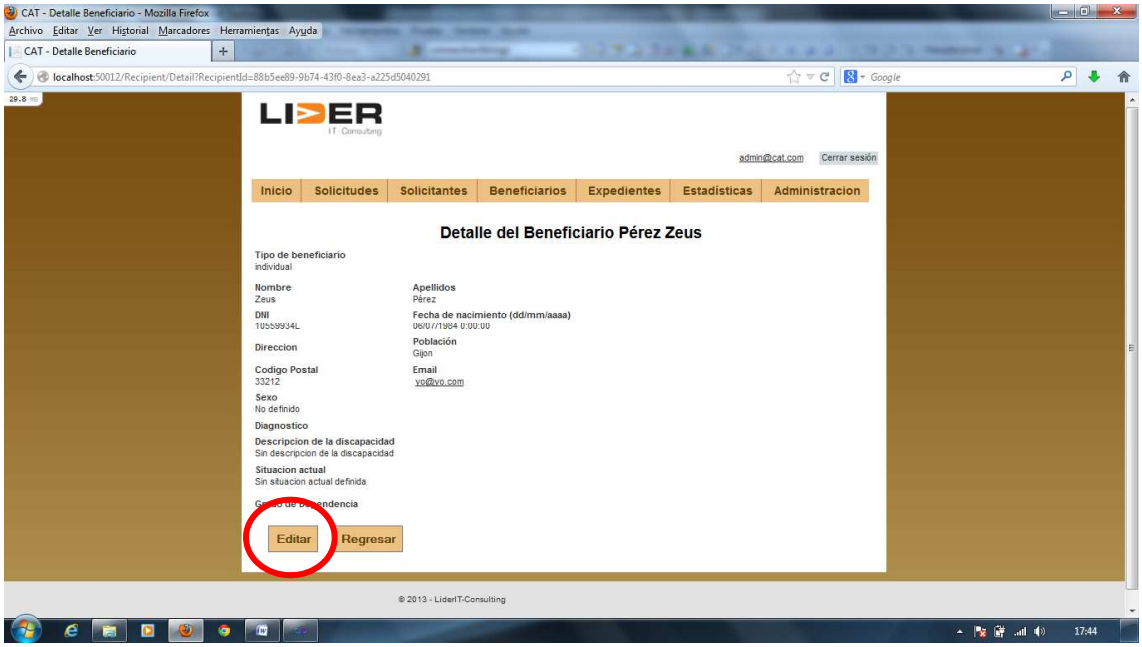


Figura 8. Detalle del beneficiario

4.3. Nueva intervención

Para crear una intervención primeramente hay que acceder al expediente concreto en el cual se desea añadir la intervención y acceder a la pestaña "Intervenciones":

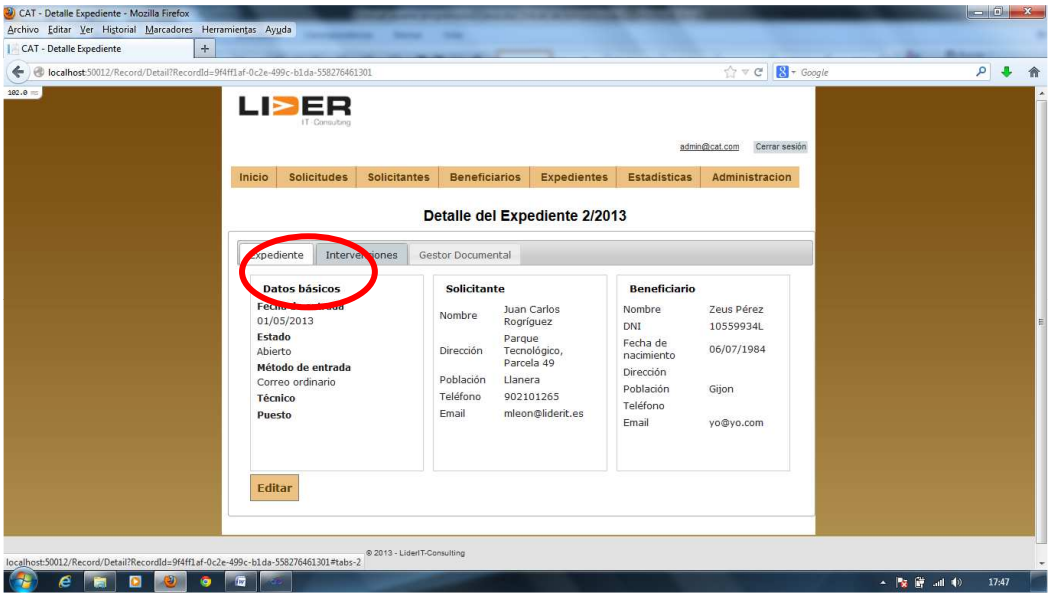


Figura 9. Detalle expediente

Una vez situados en la sección de intervenciones del expediente, se podrá registrar una nueva intervención pulsando en el vínculo con el texto "Añadir". Acto seguido aparecerá el formulario de nueva intervención.

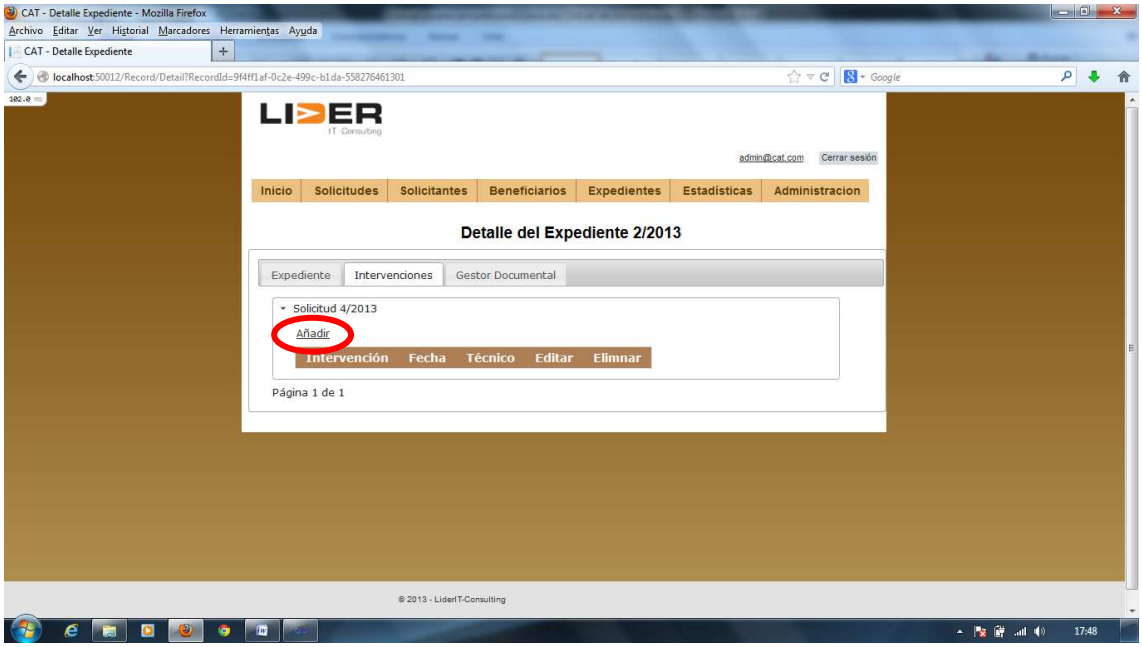


Figura 10. Añadir intervención

4.4. Estadística

Para seleccionar las estadísticas a mostrar se accederá al icono de estadísticas correspondiente, donde aparecerá las opciones de las estadísticas a mostrar:

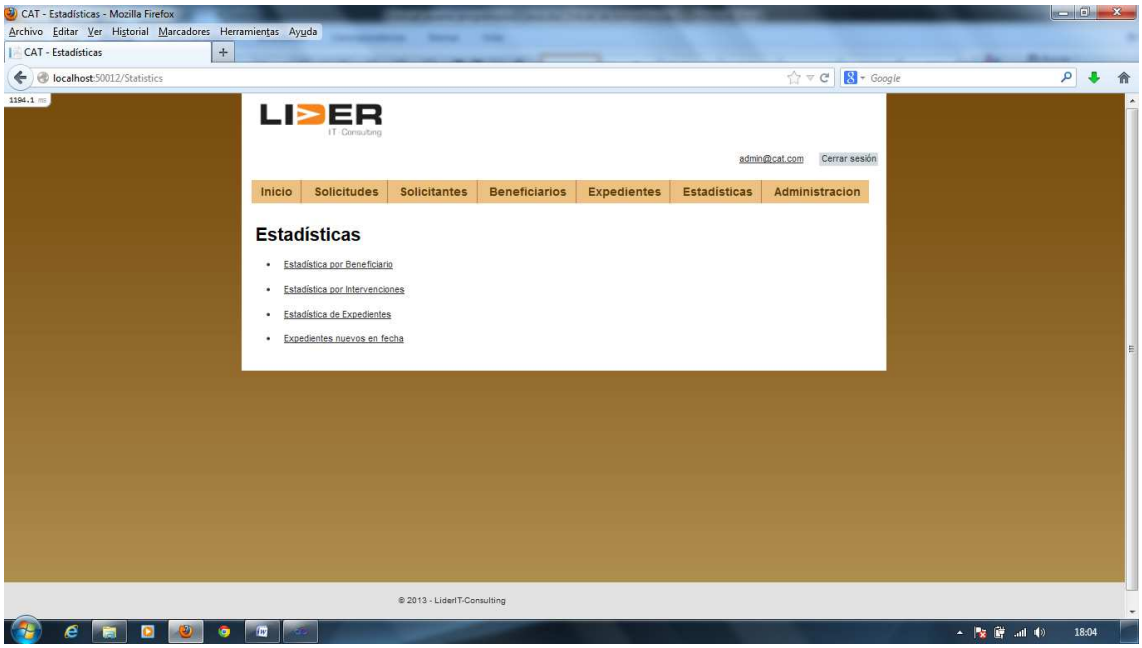


Figura 11. Estadísticas

Pulsando sobre el vínculo deseado se mostrará por pantalla las estadísticas predefinidas para la opción seleccionada.

4.5. Administración

Esta función está destinada para los usuarios administradores de a aplicación. Para acceder a esta función se hará a través de la barra superior:

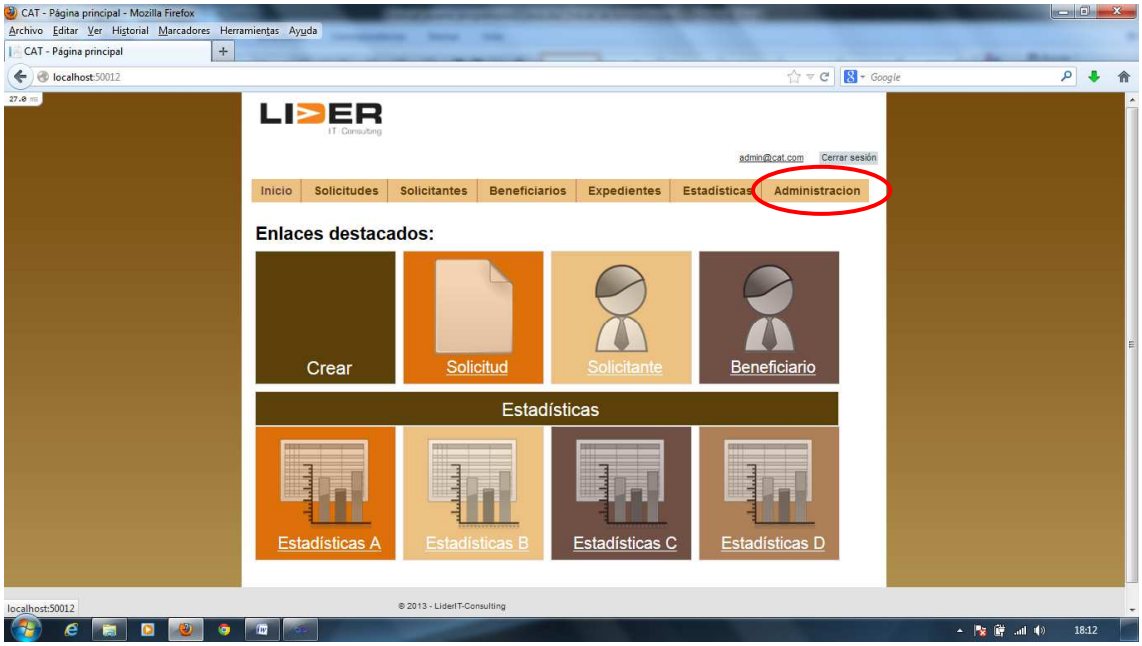


Figura 12. Administración

Posteriormente se mostraran las posibles opciones de administración de la aplicación y una vez seleccionada la opción deseada permitirá la administración de esa opción.

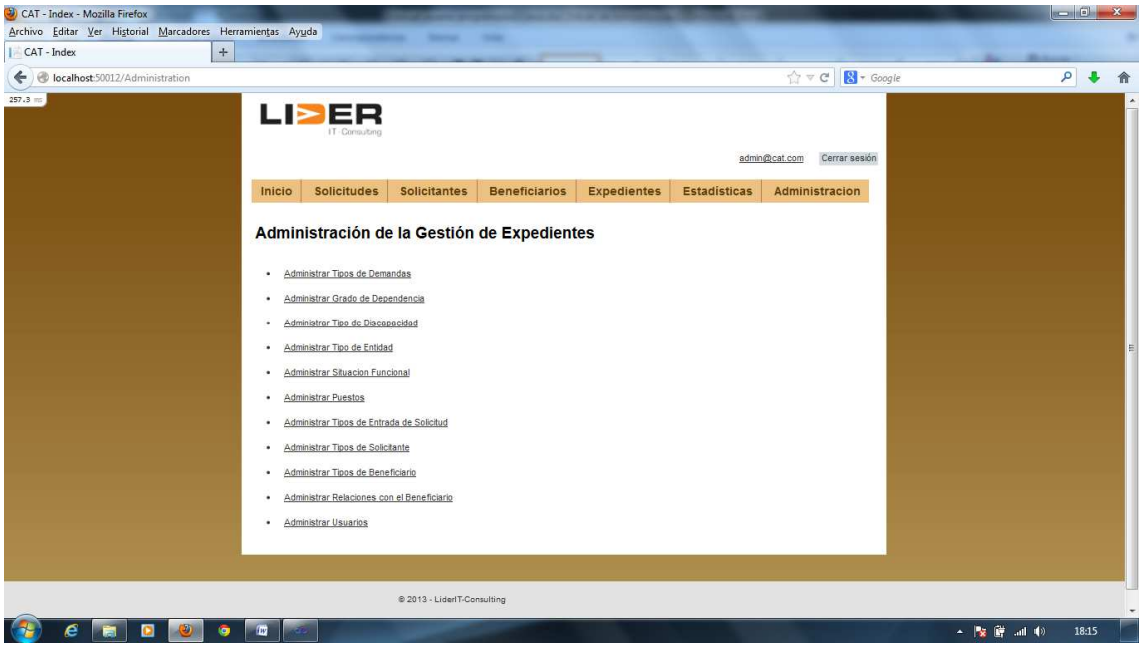


Figura 13. Opciones de administración