Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

ÁREA DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

SISTEMA DE RECOMENDACIONES PARA ACCESOS A ESTADIOS DE FÚTBOL

D. TIESTA COSÍO, Pelayo Ricardo
TUTOR: D. SÁNCHEZ RAMOS, Luciano

FECHA: Junio 2021

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. RESUMEN

Los sistemas de recomendaciones tienen un gran impacto en la sociedad actual en la que vivimos. Es por ello por lo que existen gran cantidad de sistemas enfocados a ofrecer recomendaciones para diferentes acciones cotidianas, tales como ayudar a las personas a elegir la siguiente película que pueden ver, sugerir los artículos que se pueden adquirir en una compra online o incluso recomendar una ruta de tráfico a seguir en función del estado de las carreteras. Por esto, podemos decir que los sistemas de recomendaciones buscan ayudar al usuario, de una forma más o menos lucrativa, en la toma de decisiones en su día a día.

En este proyecto se parte de una idea clara, crear un prototipo de un sistema de recomendaciones para ayudar a las personas que acceden a los estadios de fútbol a llegar a su asiento, teniendo en cuenta para ello las características que estas personas desean que la ruta tenga. Actualmente no existe ningún sistema de recomendaciones que ofrezca estas soluciones a los usuarios, y, por lo tanto, este proyecto servirá de base para futuras implementaciones de un sistema real que los espectadores de los estadios de fútbol podrán utilizar. Para ello se documentarán las características necesarias y las metodologías que dicho sistema debería seguir para su construcción e implementación y se realizará un pequeño programa que mostrará la idea básica del funcionamiento de este prototipo de sistema de recomendaciones.

Palabras clave: Sistemas de recomendaciones, rutas personalizadas, estadios de fútbol, teoría de grafos.

ABSTRACT

Recommendation systems have a big impact on the current society in which we live. That is the reason why there are a large number of systems focused on offering recommendations to different daily actions, such as helping people choose the next movie they can see, showing the recommended products that they can purchase in an online shopping or even recommending a traffic route to follow depending on the state of the roads. For this reason, we can say that recommendation systems try to help the user, in a more or less lucrative way, to make decisions in their day to day.

This project has a clear idea, create a prototype of a recommendation system to help people who access to football stadiums to arrive to their seat, considering the priorities of each one. Currently there is no recommendation system that offers these solutions to users, and, therefore, this project will help for the future implementations of a real system that football stadium spectators will be able to use. For this, the necessary characteristics and the methodologies that this system should follow in its construction and implementation will be documented, and also, a small program will be made to show the basic idea of how this prototype recommendation system works.

Keywords: Recommendation systems, personalized routes, football stadiums, graph theory.

2. MOTIVACIÓN

Con el objetivo de brindar a las personas que acceden a los estadios de fútbol y otros recintos de características similares una ruta óptima en función de sus preferencias y/o necesidades, se ha llevado a cabo el diseño y desarrollo de un sistema de recomendaciones basado en los requisitos seleccionados por cada usuario.

Este sistema pretende facilitar el recorrido desde la entrada al estadio hasta el asiento final de cada espectador. De este modo se podrá mejorar el flujo de las personas que pretenden acceder al estadio, implementando mecanismos tales como el control de masificaciones de personas o el acceso de usuarios con problemas de movilidad reducida u otras necesidades especiales, y que por ejemplo requieran de rutas más largas, aunque más lentas, pero que por consiguiente presenten una dificultad menor (rutas con más tramos planos o rutas con barandillas para facilitar la subida o bajada de escaleras).

3. INTRODUCCIÓN

En la actualidad la forma de acceder a los estadios, y en concreto a los estadios de fútbol, se realiza, por lo general, mediante mecanismos que permiten y facilitan a los organizadores llevar a cabo un acceso controlado y regulado de los espectadores; por ejemplo, mediante la lectura de las entradas con PDAs[1]. A través de estos dispositivos se comprueba la originalidad de las entradas y se garantiza que no se usen de forma reiterada la misma para diferentes espectadores.

Para llevar este control de accesos en los estadios se utilizan fundamentalmente dos[2] tipos de mecanismos, por un lado, podemos encontrar, como ya se ha mencionado anteriormente, el uso de entradas. Estas suelen ser impresas en el momento de la compra y en la mayoría de los casos no son vinculantes, por lo que la persona que la compra puede ceder su entrada a otro individuo. Por otra parte, encontramos el uso de carnets como mecanismos de identificación para el acceso a un estadio. [1] Como se indica los abonados disponen de un carnet personal e intransferible, además de una ubicación fija en el estadio durante toda la campaña o temporada.

3.1. ANTECEDENTES

Con el objetivo de analizar la viabilidad de uso del sistema que se pretende implementar, se han analizado los datos referentes a los accesos a estadios de fútbol españoles de la campaña 2017-2018[3] en las categorías de Primera División y Segunda División [2]. En concreto, nos centraremos en los accesos mediante entradas (público general) y carnet (socios o abonados).

[1] Una PDA o *Personal Digital Assistant* es un dispositivo equivalente a un ordenador de bolsillo, organizador personal o agenda electrónica, que en el caso del control de accesos a estadios de fútbol se utiliza para la lectura del código de barras de entradas o carnets.
[2] Cabe indicar que ciertos clubes pueden ofrecer pases o invitaciones a determinadas personas, pero son números significativamente pequeños en relación con los accesos mediante entrada o carnet.
[3] Datos más actuales sobre la asistencia de espectadores brindados por LaLiga.

Tipo	Liga Santander (Primera División)	Liga 123 (Segunda División)
Espectadores Totales	10.173.142	3.984.604
Socios	7.595.894	2.940.104
Porcentaje Socios	74,67%	73,79%
Entradas	1.736.599	620,455
Porcentaje Entradas	17,07%	15,57%

Tabla 3.1. Accesos Temporada 2017-2018 campos de fútbol España.

A la vista de dichos datos [Tabla 3.1] apreciamos que el porcentaje de acceso de espectadores utilizando un carnet (socios) es del 74,67 % en el caso de Primera División y del 73,79 % en Segunda; por el contrario, en el caso de los accesos mediante entradas obtenemos un porcentaje del 17,07 % y del 15,57 %, respectivamente, para cada categoría[4]. En una primera impresión podemos pensar que el acceso mediante entradas es muy reducido en comparación con los accesos de carnets. Por ello se ha realizado un análisis de los accesos a una serie de campos de fútbol de Primera División y de Segunda [Tabla 3.2 y Tabla 3.3]. Los campos elegidos en el caso de Primera División son: Camp Nou, por ser el estadio con más aforo de España, e Ipurúa, por ser el campo con menor capacidad. Del mismo modo, se han seleccionado los estadios La Romareda y Camp Nou Municipal para Segunda División.

Además, de modo representativo y para tener una visión más cercana se ha analizado el estadio de fútbol El Molinón - Enrique Castro Quini [Tabla 3.4].

Tipo	Camp Nou - Estadio del Futbol Club Barcelona	Ipurúa - Estadio de la Sociedad Deportiva Eibar
Aforo	99.354	7.083
Socios (74,67%)	74.187,63	5.288,88
Entradas (17,07%)	16.959,73	1.209,07

Tabla 3.2. Comparativa accesos Primera División entradas-socios 2017-2018.

[4] Como se puede comprobar con los datos, la suma de estos porcentajes no llega al 100 % debido a que los clubes de fútbol conceden pases e invitaciones a determinadas personas.

Tipo	La Romareda - Estadio del Real Zaragora	Camp Nou Municipal - Estadio del Reus Deportivo
Aforo	34.596	4.500
Socios (74,67%)	25.833	3.360
Entradas (17,07%)	5.906	768

Tabla 3.3. Comparativa accesos Segunda División entradas-socios 2017-2018.

Tipo	El Molinón Enrique Castro "Quini" - Estadio del Real Sporting de Gijón
Aforo	30.000
Socios (74,67%)	22.401
Entradas (17,07%)	5.121

Tabla 3.4. Comparativa accesos a El Molinón entradas-socios 2017-2018.

Tras realizar y analizar las comparativas de las Tablas [Tabla 3.2, Tabla 3.3, Tabla 3.4], podemos visualizar cómo las cifras de los accesos a los estadios de fútbol de espectadores con entradas son significativas. En el caso de los campos más grandes, como es el Camp Nou, los accesos mediante entrada equivalen a 16.959 personas, mientras que en el caso de La Romareda los espectadores que acceden con entrada corresponden a 5.906 personas.

Es por ello por lo que el desarrollo de un sistema de recomendaciones para los accesos a estadios sería de gran utilidad para personas que pasen mediante entradas, ya que estas deben encontrar su asiento sin saber dónde está concretamente ni la ruta óptima para llegar a él. Aunque las personas que acceden mediante carnets también podrían beneficiarse del uso de este sistema.

3.2. TEORÍA DE SISTEMA DE RECOMENDACIÓN

Con el objetivo de disponer de una visión más amplia de los sistemas de recomendación, así como de analizar las diferentes opciones que nuestro sistema podría optar por implementar, se han realizado los siguientes apartados teóricos.

3.2.1. DEFINICIÓN DE LOS SISTEMAS DE RECOMENDACIÓN

[3] Los sistemas de recomendación son herramientas y técnicas de software que ofrecen sugerencias entre una serie de objetos, tanto tangibles como intangibles, los cuales tienden a tener utilizad para el usuario que recibe la recomendación. Estas sugerencias equivalen al proceso de toma de decisiones, como puede ser la selección de un artículo para comprar, un nuevo libro para leer o la siguiente canción a escuchar.

Estos sistemas se centran en recomendar grupos de elementos o "ítems", y a raíz del tipo o estructura de estos se diseñan y crean tanto las interfaces gráficas como las técnicas de recomendaciones usadas. Es por esto por lo que un sistema diseñado para recomendar películas en una plataforma de contenido digital no puede ser utilizado directamente para recomendar canciones, ya que no son el mismo tipo de elementos. Por ello, el sistema deberá rediseñarse o adaptarse para poder recomendar estos "ítems" diferentes.

3.2.2. EVOLUCIÓN DE LOS SISTEMAS DE RECOMENDACIÓN

Con el paso de los años y ante los avances tecnológicos los sistemas de recomendaciones han ido evolucionando y mejorando, desde versiones sencillas que ofrecían recomendaciones al usuario basándose en la identidad de los elementos recomendados, hasta complejos sistemas que predicen exactamente lo que el usuario desea.

Es por estos avances y cambios que la definición de lo que se conoce como un sistema de recomendaciones ha evolucionado. Originalmente los autores Paul Resnick y Hal Varian [4] los definían de la siguiente forma: "*En un sistema típico, las personas proporcionan recomendaciones como entradas, que luego el sistema agrega y dirige a los destinatarios apropiados. En algunos casos, la transformación primaria está en la agregación; en otros, el valor del sistema radica en su capacidad para hacer buenas correspondencias entre quienes recomiendan y quienes buscan recomendaciones*". De este modo, su idea radicaba en que los sistemas de recomendaciones funcionaban como un apoyo entre los usuarios.

Con el paso del tiempo esta definición se ha ido modificando a una visión más amplia: [5] *"Cualquier sistema que produzca recomendaciones individualizadas como salida o que tenga el efecto de guiar al usuario de forma personalizada hacia objetivos interesantes o útiles, en un gran espacio de opciones posibles"*, y en el año 2005 Adomavicius and Tuzhilin definieron de una manera más formalizada esta idea: *"El problema de la recomendación se puede formular de la siguiente manera: Sea **C** el conjunto de todos los usuarios y sea **S** el conjunto de todos los elementos posibles que pueden recomendarse. Sea **u** una función de utilidad que mide la satisfacción del elemento **s** para el usuario **c**, es decir, u: C x S → R, donde **R** es un*

conjunto totalmente ordenado. Entonces, para cada usuario ***c*** $\in$ ***C****, queremos elegir un elemento* ***s'*** $\in$ ***S*** *que maximice la satisfacción del usuario"*.

De estas definiciones y consideraciones se han sacado los siguientes principios básicos por los que se guía cualquier sistema de recomendaciones:

- Un sistema de recomendaciones es personalizado; es decir, las recomendaciones que estos producen están destinadas a optimizar la experiencia de cada uno de los usuarios y no para brindar la misma salida a todas las personas del sistema.
- La finalidad de un sistema de recomendaciones está en ayudar al usuario a seleccionar entre unas opciones conocidas de antemano.

Las personas recurren, en su día a día, a otras que les faciliten o aconsejen sobre la mejor opción a la hora de decidir sobre un tema determinado. En nuestro caso, cuando un espectador accede a un estadio de fútbol o un entorno parecido solicita información a una persona con una mayor experiencia, como puede ser otro espectador o un acomodador, con la finalidad de obtener la mejor ruta para llegar a su asiento.

Los sistemas de recomendaciones actuales pretenden simular el comportamiento de estas personas que aconsejan o deciden por el recomendado la mejor opción a tomar. Pero el punto fuerte que los diferencia frente a un sistema clásico de recomendaciones viene dado por la posibilidad de disponer de un amplio abanico de los intereses o preferencias del usuario, o de todas las posibles opciones a tomar, facilitando con ello realizar simulaciones y valorar la mejor solución a la recomendación.

En la actualidad, el uso de los sistemas de recomendación tiene un gran impacto en nuestras vidas, es por ello por lo que grandes empresas de servicios *Online*, tales como *Amazon*, *AliExpress* o *Netflix*, entre otros, centran la mayor parte de su trabajo en diseñar sistemas para realizar recomendaciones personalizadas a cada usuario y en obtener datos de sus gustos y preferencias. Y es que uno de los puntos fuertes de los sistemas de recomendaciones y de estas empresas recae en las bases de datos donde se recogen los gustos, las valoraciones y las decisiones de los usuarios. Al disponer así de gran cantidad de información y de distintos perfiles de usuarios el ajuste de las recomendaciones será mejor y, por lo tanto, se podrá ofrecer al usuario una sugerencia de qué artículo comprar y que seguramente desee o qué película le puede resultar interesante de ver.

3.2.3. CLASIFICACIÓN DE LOS SISTEMAS DE RECOMENDACIÓN

Los sistemas de recomendación se pueden clasificar, en función de los datos que estos utilicen para realizar dichas recomendaciones, en dos grandes grupos. Por un lado, encontramos los **modelos de filtrado colaborativo**, los cuales, en función de las similitudes de gustos de los individuos, realizan filtrados para ofrecer una recomendación a cada usuario. El segundo gran grupo que podemos encontrar está formado por los **modelos basados en contenidos**, en este caso las recomendaciones se realizan teniendo en cuenta ciertos rasgos característicos o atributos del conjunto de datos a recomendar.

[6] Además de estos dos modelos principales, podemos encontrar un tercer grupo denominado **sistema de recomendación híbrido**. Este combina tanto el modelo de filtrado colaborativo como el modelo basado en contenidos con el objetivo de obtener las mejores características de cada uno y de esta forma realizar mejores recomendaciones.

3.2.3.1. SISTEMAS BASADOS EN FILTRADO COLABORATIVO

Los modelos de filtrado colaborativo, o también conocidos como modelos de recomendaciones colaborativas, tratan de evaluar y filtrar los elementos que se recomendarán al usuario en función de las opiniones de otros usuarios del sistema.

[7] Para entender el funcionamiento de este modelo de recomendaciones nos podemos fijar en cómo una persona seleccionaba una película o un libro. Si una persona A se encontraba en la oficina de su trabajo pensando en qué película vería ese fin de semana, lo más sencillo para él era recurrir a la recomendación de un compañero B que viese películas y que ambos tuviesen unas preferencias parecidas. Este compañero B podría indicarle el nombre de algunas películas que intuyera que le podrían interesar (basándose en sus gustos comunes). Además, en el momento en que A recibiera una lista de posibles películas para ver de su compañero este podría comentar con otras personas que las hubiesen visto cuál considerarían que debería ver.

El problema principal de este sistema de recomendaciones “de boca en boca” está en que, en nuestro ejemplo, la persona A debería conocer a un gran número de compañeros en su oficina que hayan visto la película, que tengan gustos similares y que disponga de tiempo suficiente para ir preguntando a todos, para finalmente sacar conclusiones sobre la mejor recomendación.

Pero ¿y si lo valoramos en un sistema de recomendaciones utilizando una página web? De este modo obtendremos un entorno donde los usuarios, una vez vista la película, la puntúan y, además, se podrá indagar con un buen cuestionario sobre sus géneros

favoritos, sus actores preferidos, la duración que les gusta, etc., y conseguir así sus puntuaciones para una gran cantidad de atributos de las películas. De esta manera, cuando un usuario desea que se le recomiende una película solamente tendrá que entrar en la página la web, indicar sus gustos y preferencias y el sistema, mediante un filtrado colaborativo, buscará en función de los datos introducidos por el nuevo usuario y los ya registrados en la web y le facilitará información sobre una serie de recomendaciones. Con la ventaja, frente al sistema tradicional, de que una película puede tener miles o millones de valoraciones, algo que sería imposible de realizar preguntando uno a uno, como lo susodicho en el ejemplo anterior.

Este tipo de sistemas no solo nos permiten ofrecer recomendaciones equiparando los gustos de los usuarios, sino que también podemos comparar dos objetos, en cuyo caso estaríamos hablando de un sistema "item-based" o "item-item". Este tipo de recomendadores responden a la lógica de si te gusta este objeto te gustará este otro. Para ello se compararán los atributos de los dos objetos y si tienen un alto grado de similitud pueden ser interesantes para el usuario.

3.2.3.2. SISTEMAS BASADOS EN CONTENIDOS

Los modelos basados en filtrado según su contenido utilizan las características de los elementos con la finalidad de recomendar otros similares, fundamentándose en datos extraídos del usuario, como pueden ser acciones o selecciones previamente realizadas, así como comentarios sobre el tipo de elementos deseados.

Con el fin de poder entender el concepto de filtrado basado en contenidos se ha ilustrado, a continuación, una pequeña simulación de este modelo aplicado a la plataforma de distribución digital de aplicaciones móviles *Google Play Store* [8].

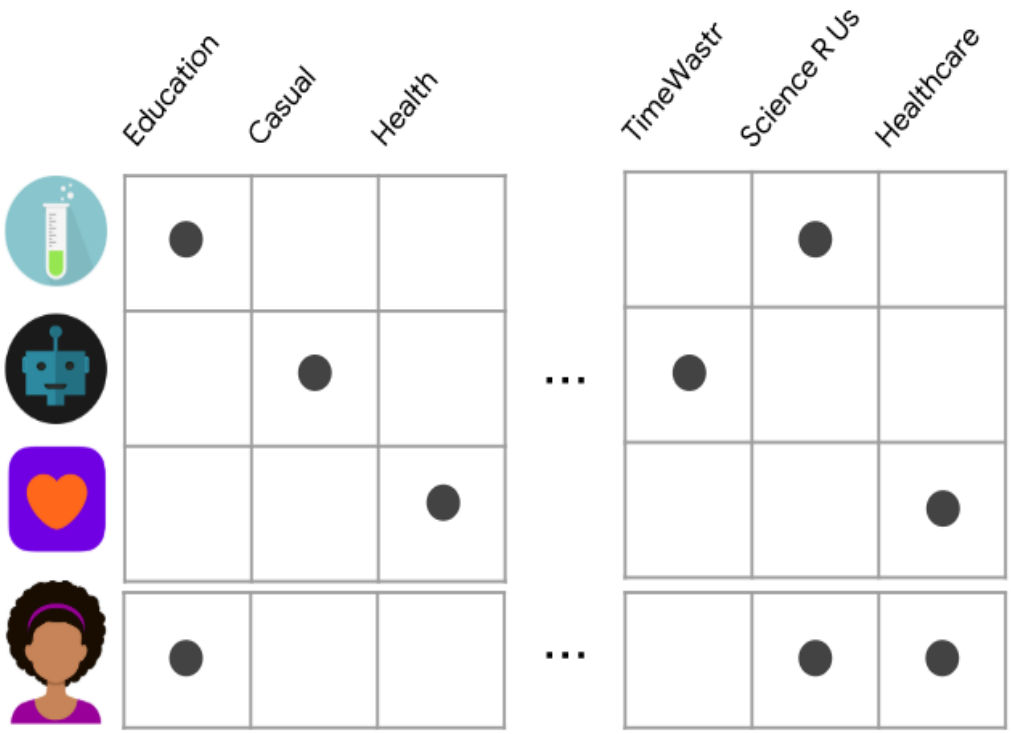

Ilustración 3.1. Matriz de características para un modelo basado en contenido.

La matriz de la [Ilustración 3.1] clasifica y relaciona cada aplicación (representada en filas) con una serie de características (representadas en columnas) previamente otorgadas por *Google Play*; en este caso, las aplicaciones pueden clasificarse en categorías como son: "*Education*", "*Casual*" y "*Health*", además de en otros factores como pueden ser el creador o autor de la aplicación: "TimeWastr"o "Science R Us", y de otros atributos como por ejemplo que cuida de la salud o "Healthcare".

Supongamos así mismo que la matriz es una matriz binaria, por lo que un punto equivalente a un 1 indicará que la aplicación cumple o dispone de esa característica.

Por otro lado, se ha clasificado también al usuario. Para realizar esto existen diferentes métodos, uno de ellos sería que al crear el perfil de este se ha marcado que le gustan las aplicaciones de educación, por lo que se aplicará un valor de "1" a las categorías de "*Education"*. Además, el usuario en cierto momento instaló una aplicación del creador Science R Us, con lo que se sabe que también está interesado en ese tipo de aplicaciones.

Una vez extraída la información del usuario y clasificadas las aplicaciones se debe configurar el sistema para calificar cada elemento candidato basándose en una métrica de similitud, como puede ser el producto escalar.

3.2.3.2.1. CÁLCULO DEL PRODUCTO ESCALAR

Considérese el caso en el que el usuario ***x*** y la aplicación ***y*** son ambos vectores binarios. Dado que si en $\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i$ una de las características *x* o *y* es verdadera, el sumatorio será igual a "1", o lo que es lo mismo $\langle x, y \rangle$ es el número de características que están activas en ambos vectores de forma simultánea. Además, en el caso de coincidir las dos características estaríamos ante una mayor similitud, como sucedería en el caso de la primera aplicación, ya que tanto el usuario como esta disponen de la característica "*Education*" y son del editor de aplicación "*Science R Us"* [8].

3.2.3.2.2. DIFERENCIAS CON EL MODELO BASADO EN FILTROS COLABORATIVOS

Los sistemas de recomendaciones basados en filtros colaborativos utilizan los comentarios, valoraciones u opiniones de los usuarios del sistema sobre una serie de elementos con la finalidad de obtener la recomendación. Pero ¿qué sucede con los elementos que no tienen ninguna valoración?, ¿cómo puede destacar un nuevo artículo a la venta en *Amazon?*, o ¿qué se debe recomendar a un nuevo cliente o usuario del sistema que entra por primera vez en *Amazon?* y al que se deben mostrar algunos productos de su interés. Estos problemas son conocidos como "*cold start"* o arranque en frío de un sistema de recomendaciones y, como se ha planteado anteriormente, es una situación muy típica de encontrarse en el sistema cuando el usuario se registra por primera vez, cuando se agrega un posible elemento al sistema de recomendaciones o incluso al crear el propio sistema, ya que no se tiene ningún dato sobre los artículos y sus relaciones ni tampoco perfiles de usuarios para poder llevarlas a cabo.

Este problema no está presente en los modelos basados en contenidos ya que los elementos se pueden completar con una serie de atributos, como pueden ser el género musical de una canción, su duración, su cantante o una serie de variables que ayuden a recomendar dicha canción al usuario.

3.2.3.3. SISTEMAS HÍBRIDOS

Debido a la gran cantidad de sistemas de recomendaciones que se desarrollan hoy en día y a la necesidad que surge tanto de corregir los defectos de los dos modelos anteriormente

mencionados como de explotar los puntos fuertes de cada uno, aparece un tercer grupo denominado sistema de recomendaciones híbridos.

La combinación de los modelos basados en contenidos y filtrado colaborativo, junto a la utilización de "*Deep Learning*", nos permite crear nuevos modelos, configurándolos, por ejemplo, para mezclar las salidas de un sistema de recomendaciones basados en contenidos, como podría ser un conjunto de elementos de una tienda de ropa [9], con un filtrado colaborativo de estas prendas concretas y de este modo obtener una mejor recomendación.

3.2.3.4. ELECCIÓN DE LA MEJOR SOLUCIÓN

Nuestro sistema contará con un modelo de recomendaciones basado en contenidos, ya que como se detallará a continuación esta es, desde nuestro criterio, la mejor solución a la hora de brindar recomendaciones personalizadas para cada espectador del evento.

En primer lugar, debemos analizar cuál es nuestro elemento a recomendar. Cuando un usuario accede a un campo de fútbol accederá a nuestro sistema para obtener una recomendación de la ruta idónea a sus necesidades. Es por ello que en este punto tenemos un perfil del usuario y una serie de variables a tener en cuenta, como podrían ser la edad, las preferencias de recorridos (escaleras con barandillas, suelos planos, tramos no congestionados), la movilidad del espectador, la hora de llegada, etc.

Debemos darnos cuenta de que un asiento no es único para todo el estadio y, dado que forma parte de la recomendación, hace que cada salida producida por el sistema sea única para cada espectador y para cada día de partido. Es por ello por lo que las recomendaciones para acceder al asiento ***a*** el día ***x*** para el espectador ***e*** son únicos, ya que otro espectador, frente al mismo asiento, pero otro día, puede tener otras preferencias totalmente diferentes; incluso podría no darnos su valoración sobre el recorrido, por lo que no tendríamos puntuaciones para realizar un filtrado colaborativo. Además se debe tener en cuenta que, al ser un asiento único en todo el estadio, pues cada uno presenta unas condiciones de acceso diferentes, para poder hacer un modelo basado en filtrado colaborativo deberíamos realizar el aprendizaje de este con una duración temporal muy elevada, ya que deberíamos recomendar para cada perfil todas las posibles rutas a un único asiento, hasta obtener de este modo un conjunto de valoraciones con las posibles recomendaciones de recorridos.

Otro punto a considerar es que el sistema varía de una semana a otra. En primer lugar, los asientos contiguos pueden cambiar y cada fin de semana la recomendación será diferente. Además los tramos, como puede ser una escalera, pueden llenarse o estar cerrados por obras y el sistema deberá recomendar otro recorrido haciendo totalmente diferente una recomendación de otra.

Por todo esto, se ha optado por un sistema de recomendaciones basado en contenidos frente a un modelo de filtrado colaborativo.

3.2.4. EVALUACIÓN DE LOS SISTEMAS DE RECOMENDACIONES

[10] Del mismo modo que sucede con cualquier algoritmo de aprendizaje automático o *Machine Learning*, se necesita poder evaluar el rendimiento de los sistemas de recomendaciones con el objetivo de decidir si el ajuste a la situación que se está tratando es correcto. Existen dos conjuntos de métodos para evaluar dichos sistemas: evaluaciones basadas en métricas bien definidas y evaluaciones basadas, principalmente, en el juicio humano y la estimación de la satisfacción producida.

3.2.4.1. EVALUACIÓN BASADA EN MÉTRICAS

Este tipo de evaluaciones son utilizadas para sistemas de recomendaciones basados en modelos que generan valores numéricos, tales como predicciones de puntuaciones o probabilidades de emparejamiento. En estos casos la calidad de la recomendación se puede comprobar con técnicas simples, como puede ser la medición del error producido, por ejemplo, mediante el error cuadrático medio: $ECM = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$, donde $\hat{Y}$ es el vector de n predicciones e Y es el vector de valores verdaderos.

Pero *¿qué sucede si las recomendaciones no devuelven datos numéricos?* En ese caso se debe "binarizar" las soluciones, para ello se tomará un umbral de referencia y los valores por encima de este serán positivos mientras que los que se ubiquen por debajo serán negativos. Tras realizar esta "binarización" se podrá evaluar la calidad de las salidas del mismo modo que se realiza para los sistemas de recomendaciones que generen valores numéricos.

Además, existe un tipo de sistemas que generan listas de recomendaciones, del tipo "*item-item"* o *"user-user"*. Para estos casos no es posible aplicar una evaluación basada en métricas de forma directa sobre las salidas, sino que se debe obtener la valoración de esta

por parte del usuario; es decir, la persona que recibe la recomendación debe puntuar cuan acertada ha sido y este valor numérico será el analizado.

3.2.4.2. EVALUACIÓN BASADA EN EL JUICIO HUMANO

A la hora de diseñar un sistema de recomendaciones puede ser recomendable obtener, por un lado, un modelo de recomendaciones que se sabe que será bueno y, por otro, las propiedades de "explicabilidad" de estas.

La "explicabilidad" es un punto clave a la hora tener éxito con un algoritmo de recomendaciones. Esto se debe a que si un usuario del sistema no entiende por qué se le recomienda un artículo o un objeto específico tenderá a perder la confianza en estas recomendaciones y empezará a omitirlas o dudar de su valía. Para poder realizar un sistema con recomendaciones explicables se debe agregar al artículo o la lista de artículos una oración que explique por qué se están recomendando; esto es algo que se puede observar al comprar en una tienda de e-commerce o comercio electrónico en la que se usan frases, junto a las recomendaciones, del estilo "si te ha gustado este artículo que tienes en la cesta, te pueden interesar estos". Con estas frases se aumenta la "explicabilidad" del elemento recomendado ya que se está mostrando por qué se recomendó ese producto y no otro. Es en este instante cuando se puede evaluar el modelo de recomendaciones basándose en el juicio humano.

3.3. TEORÍA DE GRAFOS

La Teoría de Grafos nos permite crear modelos de abstracción con la finalidad de poder representar los diferentes elementos del sistema con los que el usuario interactuará, logrando simplificar el diseño de la red de caminos mediante la creación de nodos y enlaces que representarán los diferentes pasillos, asientos y uniones del sistema.

De este modo, mediante la representación gráfica y el uso de la Teoría de Grafos, podremos obtener la ruta óptima para el usuario e informarle de las demás opciones posibles que se podrían tomar para llegar al asiento.

3.3.1. DEFINICIÓN DE GRAFO

Un grafo es un par, ordenado o no ordenado, $G = (V, E)$ de conjuntos que satisfacen que $E \subseteq [V]$; es decir, *E* es un subconjunto de *V*, siendo *V* los vértices o *nodos* y *E* los arcos o enlaces que componen el grafo *G*.

De esta forma, la representación gráfica habitual de los grafos consiste en dibujar un punto para cada nodo, uniéndolos mediante una línea que representa el enlace, tal y como se puede apreciar en la [Figura 3.1].

Así, el grafo está formado por 7 nodos o vértices, por lo tanto, podríamos decir que $V = \{1,2,3,4,5,6,7\}$, y el conjunto de enlaces quedaría definido como $E = \{ (1,2), (1,5), (2,5), (3,4), (5,7) \}$.

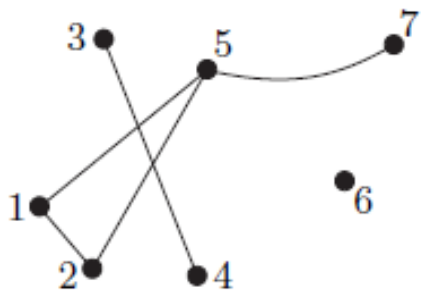

Figura 3.1. Representación gráfica de un grafo.

3.3.2. CLASIFICACIÓN DE LOS GRAFOS

Existe una amplia variedad de tipos o clasificaciones para los grafos en función de las virtudes o características de cada uno de ellos. En consecuencia, se ha realizado una descripción de una serie de categorías de las que nuestro sistema podría disponer y que es conveniente entender.

3.3.2.1. GRAFOS DIRIGIDOS

Los grafos dirigidos o dígrafos son un tipo en los que los enlaces que unen dos nodos solamente presentan una dirección.

De este modo, como podemos apreciar en la [Figura 3.2], el grafo formado por los vértices $\{1,2\}$ presenta una unión desde el nodo 1 hasta el nodo 2; es decir, el grafo está formado

por un enlace $(1,2)$ y es entendible gráficamente gracias a la utilización de la flecha que nos indica el sentido de la relación.

Además, debemos destacar que este tipo de grafos nos permite indicar el sentido del enlace. Por ello, si deseamos representar un pasillo por el que solo se puede transitar en una dirección, para un supuesto en el que deseamos controlar el flujo de la gente al entrar al estadio, podremos representarlo como un enlace $(1,2)$ y se entenderá que existe un camino desde el nodo 1 hasta el nodo 2, pero no podremos llegar al nodo 1 desde el otro nodo debido a que $(1,2) \neq (2,1)$.

Figura 3.2. Representación gráfica de un grafo dirigido.

3.3.2.2. GRAFOS NO DIRIGIDOS

Los grafos no dirigidos son aquellos en los cuales las aristas que los forman son bidireccionales; en otras palabras, el enlace que une el nodo 1 y el nodo 2 se puede recorrer en ambos sentidos.

Como se refleja en la [Figura 3.3] este tipo de grafos se pueden identificar por estar los nodos unidos por líneas, sin flechas que indiquen la dirección. Igualmente, este tipo de grafos cumplen la relación $(1,2) = (2,1)$, por lo que si existe un camino desde el nodo 1 hasta el nodo 2 también existirá el recorrido opuesto.

Figura 3.3. Representación gráfica de un grafo no dirigido.

3.3.2.3. GRAFOS MIXTOS

Los grafos mixtos se pueden definir como el tipo de grafos donde existen enlaces tanto dirigidos como no dirigidos. Estos serán de gran utilidad en nuestro sistema ya que podríamos representar pasillos del estadio y asignarles tanto un sentido como una dirección. De este modo, un pasillo representado con una flecha se entenderá como un camino de una única dirección y, por otro lado, un enlace con una línea sin flechas representará un camino bidireccional.

Esto nos permitirá, entre otras cosas, crear recorridos de una dirección, como pueden ser escaleras de subida únicamente, y con ellos implementar soluciones a las aglomeraciones de personas.[5]

3.3.2.4. GRAFOS PONDERADOS

Los grafos ponderados, pesados o con costos son un tipo de grafos donde las aristas o enlaces tienen una etiqueta. Estas etiquetas pueden reflejar un nombre, un costo o un valor representando cualquier tipo de dato.

Estos grafos nos permiten modelar problemas reales, como puede ser calcular un recorrido en función del interés o la mejor aportación de los enlaces que recorreremos. De esta forma, supongamos que un transportista debe entregar tres paquetes en tres ciudades diferentes que están conectadas entre sí. El gráfico representará en los nodos las ciudades, y los enlaces llevarán asociada la distancia entre las ciudades [Figura 3.4]. De este modo, mediante los grafos ponderados podremos minimizar la distancia recorrida y ofrecer al transportista la mejor ruta en función de la distancia.

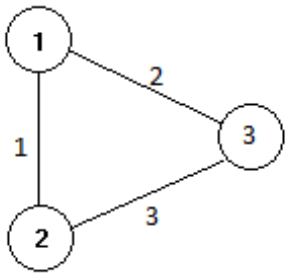

Figura 3.4. Gráfico ponderado de tres ciudades.

[5] Pese a ser una muy buena solución, no se podrá implementar en el código por la librería de grafos que se usará.

3.3.2.5. GRAFOS CONEXOS O INCONEXOS

Un grafo *G* se considera conexo si, para un par de nodos n_1 y n_2, existe un camino posible desde el nodo n_1 hasta el nodo n_2 [Figura 3.5]. Asimismo, un grafo es fuertemente conexo si para todos los nodos de este existe un camino de ida y de regreso que conecte cada uno de los nodos con todos los demás [Figura 3.6].

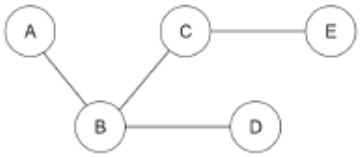

Figura 3.5. Grafo conexo.

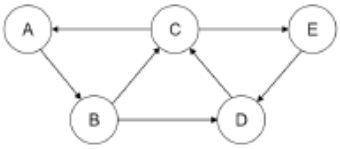

Figura 3.6. Grafo fuertemente conexo.

De la misma forma, un grafo es inconexo si para un nodo n_1 no existe un camino que conecte todos los pares de nodos y, por consiguiente, no habrá un camino que transcurra por todos los nodos del grafo [Figura 3.7].

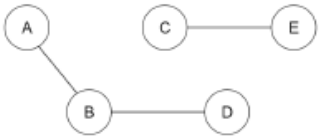

Figura 3.7. Grafo inconexo.

3.3.2.6. ÁRBOL

Un árbol consiste en un grafo en el que todos los nodos están conectados por exactamente un camino, y así mismo el grafo que forma el árbol debe ser conexo, no dirigido, conectado a todos los nodos y sin ciclos [Figura 3.8].

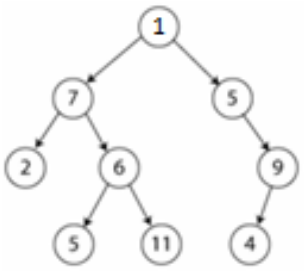

Figura 3.8. Árbol.

3.3.3. REPRESENTACIÓN DE GRAFOS

Existen varias formas de representar un grafo de manera computacional con el objetivo de poder realizar operaciones sobre ellos. Por esto se han descrito a continuación tres formas distintas de representar un grafo, cada una con sus ventajas y sus desventajas.

3.3.3.1. LISTAS DE ARISTAS

Una de las formas de representación de grafos más sencillas consiste en representar un grafo en un *array* o lista de aristas. Para realizar esto cada elemento de la lista estará compuesto por un vector con dos componentes que representarán los vértices o nodos del enlace que se pretende reflejar.

En el caso de que se trabaje con grafos con peso se añadirá otra componente al vector de aristas y por lo tanto cada vector de la lista tendrá los dos nodos y el enlace. Verbigracia, la representación mediante una lista de aristas del grafo de la [Figura 3.1], sería: { [1,2], [1,5], [2,5], [3,4], [5,7] } .

3.3.3.2. MATRIZ DE ADYACENCIA

Dado un grafo con un numero V de vértices la matriz de adyacencia se define como una matriz de |V| x |V|, de ceros y unos, donde el valor para la posición $[i, j]$ de la matriz será uno si, y solamente si, la arista (i, j) está en el grafo.

Mediante la matriz de adyacencia también podemos indicar el peso de las aristas. En ese caso reservaremos un valor como puede ser null para indicar que esa arista no existe (de este modo null se entenderá como un cero) y un valor distinto representará que existe el enlace y además corresponderá a su peso.

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Figura 3.9. Matriz de adyacencia del grafo de la Figura 3.1.

En la [Figura 3.9] podemos ver la representación mediante la matriz de adyacencia del grafo, y de dicha matriz podemos deducir lo siguiente. Si nos fijamos en la diagonal de la matriz podemos ver que todo son ceros, lo que nos indica que no tenemos ningún ciclo para ningún nodo. Además, vemos que el nodo 6 es un nodo aislado puesto que no se comunica con ningún otro nodo.

3.3.3.3. LISTA DE ADYACENCIA

Una lista de adyacencia consiste en la combinación de las matrices de adyacencia con las listas de aristas, lo que genera, por cada vértice *i*, un vector de vértices adyacentes a este. Cada posición del vector contendrá un puntero haciendo referencia a cada nodo adyacente.

Este tipo de implementaciones permite optimizar el uso de la memoria necesaria para almacenar el grafo, consiguiendo así búsquedas más rápidas y, por lo tanto, su implementación en sistemas reales acelerará la ejecución.

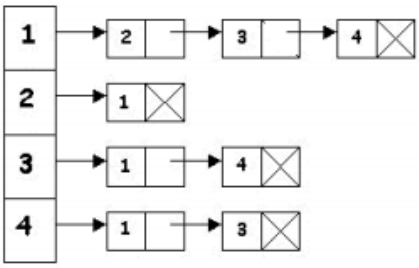

Figura 3.10. Lista de adyacencia del grafo de la Figura 3.11.

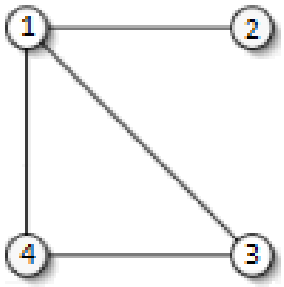

Figura 3.11. Grafo de la lista de adyacencia.

3.3.4. SOLUCIÓN PROPUESTA

Dada la complejidad de los elementos que forman el estadio de fútbol, se debe realizar una abstracción que permita modelar cualquier estadio y, de este modo, poder realizar las diferentes recomendaciones de una manera unificada para cada estadio, cada espectador y cada asiento.

Lo primero que debemos observar es la asignación de los enlaces del grafo en el estadio real. Para ello, se ha considerado que cada pasillo o escalera del sistema corresponderá a un enlace en el grafo. De esta forma podremos asignar a los enlaces un peso o atributos que nos permitan realizar posteriormente las recomendaciones, tales como: longitud, amplitud, cantidad de gente transitándolo, etc.

En segundo lugar, definiremos los nodos del grafo. Como nodos iniciales tomaremos cada una de las puertas de acceso al estadio[6] por las que el espectador puede acceder. Siguiendo

[6] Puerta de acceso: En nuestro caso un espectador únicamente accede por una puerta.

la misma metodología, los nodos finales corresponderán a los asientos donde se sentará el espectador.

El último tipo de nodos corresponde a los formados por las uniones de pasillos o escaleras. Este tipo de nodos nos permitirá "jugar" con las recomendaciones, ya que nos facilita la creación de caminos más cortos y, por lo tanto, modificar las rutas y las recomendaciones de la manera que mejor consideremos. Además, este tipo de nodos al crear enlaces de menor tamaño posibilitará la implementación de ciertas funcionalidades, como el control de aglomeraciones o el cierre de enlaces entre ellos.

De esta forma nuestros grafos del estadio serán representados mediante grafos dirigidos, ponderados y conexos.

El grafo debe de ser dirigido debido a que mediante este tipo de enlaces podremos asignar un sentido al pasillo y así, por ejemplo, controlar el flujo de los espectadores cerrando pasillos en un sentido determinado.

Por otro lado, el grafo será ponderado dado que cada enlace tendrá unos atributos que nos permitirán mediante nuestro mecanismo de recomendaciones elegir cuál es el enlace óptimo a las exigencias o necesidades del espectador.

Por último, que nuestros grafos sean conexos nos indica que podemos acceder a cualquier asiento; aplicado a nuestro sistema supone que no a cualquier asiento del estadio, sino a cualquier asiento de la grada a la que entra el espectador. Una grada está formada por los distintos sectores donde se encuentran los asientos, y de este modo en el estadio El Molinón, que se representa en la [Ilustración 5.7], podemos identificar como a la Grada Norte se accede desde las puertas 4, 5 y 6 y desde estas podremos llegar hasta cualquier asiento de la grada, por lo que estamos ante grafos conexos.

4. OBJETIVOS

4.1. OBJETIVOS GENERALES

Con el desarrollo de este proyecto se pretende definir la estructura de un sistema basado en recomendaciones y grafos que permita a las personas, en función de sus preferencias, encontrar la ruta que mejor se ajuste a sus requisitos o necesidades. Este sistema, por lo tanto, devolverá un itinerario personalizado para cada individuo en función de una serie de atributos que previamente los usuarios han indicado.

4.2. OBJETIVOS ESPECÍFICOS

Este sistema pretende mejorar el acceso a los estadios de fútbol de los usuarios. Por ejemplo, si un usuario necesita acceder a su asiento por una ruta que disponga de barandillas (esto puede ser debido a problemas de movilidad) el sistema debe recomendar, si existe, una ruta que se ajuste a sus necesidades y que devuelva la ruta óptima que cumpla estas condiciones; si no existe ninguna ruta el sistema deberá obtener otra que satisfaga lo máximo posible las necesidades del espectador o informar de que no se encuentra ningún recorrido para recomendar.

Otro ejemplo del uso de este sistema estaría enfocado a personas que precisen acceder de la forma más rápida a su asiento (personas que lleguen tarde), en cuyo caso se recomendará la ruta más rápida sin necesidad de que esta disponga de elementos adicionales (como en el caso anterior) para el acceso.

Una de las aplicaciones más importantes que se podría implementar mediante un sistema de recomendaciones de este tipo sería el control de acceso de forma masificada. De esta manera el sistema podría controlar las aglomeraciones de gente y modificar los flujos de tráfico de personas distribuyéndolo por diferentes rutas, lo que podría ser de gran utilidad para evitar problemas relacionados con

la COVID-19[7], que pretende eludir o minimizar las agrupaciones de personas por los mismos espacios.

[7] La COVID-19 es la enfermedad causada por el nuevo coronavirus conocido como SARS-CoV-2. La OMS tuvo noticia por primera vez de la existencia de este nuevo virus el 31 de diciembre de 2019, al ser informada de un grupo de casos de «neumonía vírica» que se habían declarado en Wuhan (República Popular China). [15]

5. METODOLOGÍA

5.1. TECNOLOGÍAS UTILIZADAS

A continuación, mencionamos las tecnologías que se emplearán para la realización de este proyecto.

5.1.1. LENGUAJE

Para el desarrollo de este proyecto se ha decidido la utilización de Python versión 3.8.5[8] debido a que estamos ante un producto *software* prototipo cuyo objetivo es servir de guía para futuras implementaciones de los algoritmos desarrollados, y con el que lograremos que se realice un código eficaz y con una ejecución veloz y fluida.

Se considera que Python es el lenguaje óptimo para desarrollar una primera versión de los algoritmos y del sistema que en un futuro podrá implementarse en otros lenguajes y con otras tecnologías, con la finalidad de mejorar la fluidez y los tiempos de ejecución de las recomendaciones.

Además, cabe destacar que el uso de Python nos facilita la utilización de librerías para el trabajo con grafos y modelos de recomendaciones.

5.1.2. LIBRERÍAS UTILIZADAS

Con el objetivo de facilitar la programación del código, así como de incluir funcionalidades extras ya desarrolladas y que permitan obtener un programa de mayor calidad, se han utilizado principalmente[9] las siguientes librerías o módulos Python:

[8] El código ha sido desarrollado y probado en Python 3 para Windows 10, mediante el IDE Spyder 4.2.5.
[9] El código puede contener más librerías, pero por su poca repercusión en él o su baja influencia se han decidido no incluir en este apartado.

5.1.2.1. MATPLOTLIB

Matplotlib es una librería completa que permite crear visualizaciones 2D estáticas, animadas e interactivas en Python, como se aprecia en la [Ilustración 5.1].

El uso principal en este proyecto es representar de forma gráfica los nodos que componen la grada del estadio. De esta forma es posible entender de manera más intuitiva las rutas recomendadas, así como facilitar la depuración tanto del modelo de la grada, como de las soluciones ofrecidas al usuario.

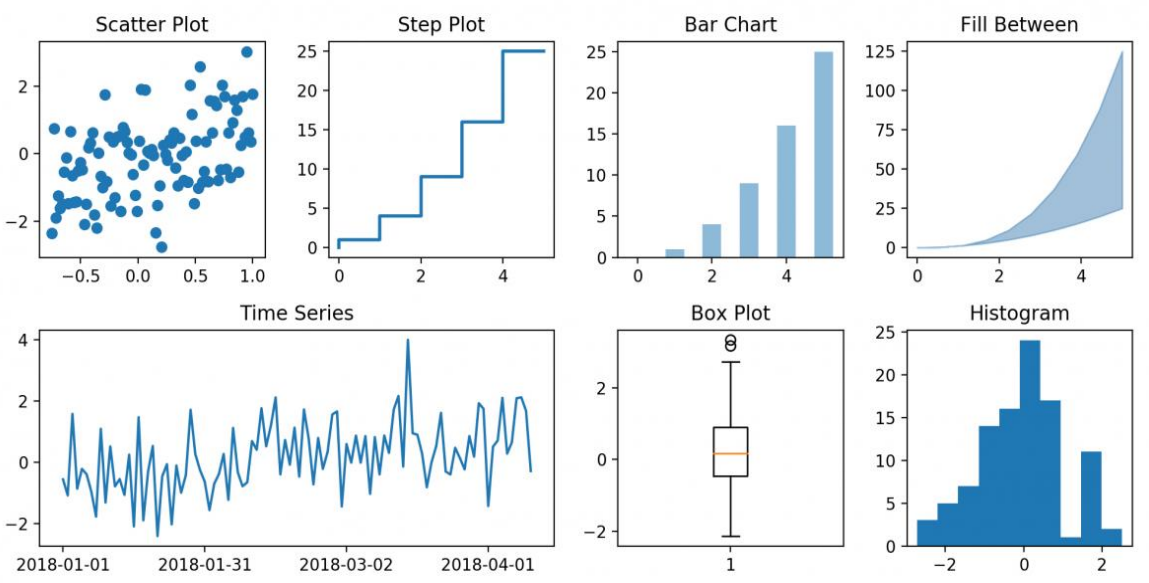

Ilustración 5.1. Ejemplo de Matplotlib.

5.1.2.2. NETWORKX

NetworkX es una librería sencilla de implementar, pero muy completa, para la creación, manipulación y estudio de grafos y redes como los que se muestran en la [Ilustración 5.2].

El uso principal en este proyecto es la creación de grafos compuestos por enlaces que simulen los diferentes pasillos del estadio, así como de las uniones de estos. De este modo se podrá visualizar el grafo de forma gráfica, además de realizar sobre él modificaciones, lecturas y aplicaciones de algoritmos ya definidos para la obtención de diferentes rutas.

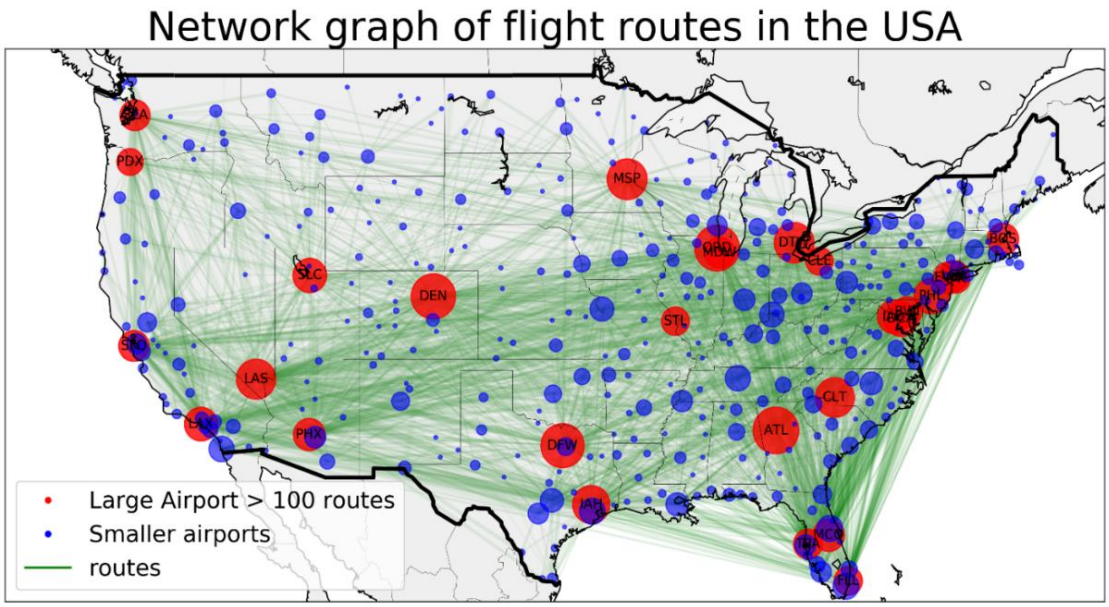

Ilustración 5.2. Ejemplo de representación de rutas de vuelos en USA con NetworkX.

5.1.2.3. TKINTER

Tkinter es un módulo Python que permite crear y mostrar interfaces gráficas al usuario, y de esta forma, establecer una comunicación entre las variables del código y el usuario.

El uso principal en este proyecto será la creación de un formulario para leer las preferencias de la ruta del espectador.

5.1.2.4. NUMPY

Numpy es una biblioteca que facilita y acelera el trabajo con gran volumen de datos vectoriales y matrices en Python.

5.1.3. CONTROL DE VERSIONES

Con la finalidad de mantener y desarrollar un código eficaz y limpio se ha implementado el control de versiones mediante Git. En concreto, se usarán los servicios de la empresa GitHub, de este modo se podrá conservar el código almacenado en la nube e ir actualizándolo con las diferentes versiones y cambios que se realicen en este.

5.1.4. DOCUMENTACION DEL CÓDIGO

Con el objetivo de que el código realizado sea fácil de entender para futuros desarrollos o implementaciones, se ha decido crear una documentación del uso del programa, las librerías utilizadas y los métodos creados en los distintos módulos del proyecto.

Para elaborar esta documentación se ha empleado la herramienta de código abierto *Read the Docs*, la cual nos brinda una página web en la que se puede hospedar la documentación del código y del proyecto.

Read the Docs se integra con GitHub de forma que se comparte el repositorio Git entre una plataforma y otra. De esta manera *Read the Docs* es capaz de leer los comentarios en formato Docstrings de las diferentes funciones de los módulos Python y generar con ellos la documentación en formato *.html*, para posteriormente crear una página web mostrando dicha documentación.

5.2. ALGORITMOS

A continuación, se van a describir los algoritmos que se tendrán en cuenta para ofrecer las distintas funcionalidades del sistema.

5.2.1. RUTA MÁS CORTA

El algoritmo de búsqueda de la ruta más corta consiste en hallar el camino más breve entre dos vértices, en nuestro caso estos serán la entrada al estadio (nodo inicial) y el asiento del espectador (nodo final).

Para poder encontrar el camino más corto del grafo entre dos nodos debemos tener definidos previamente en este el tamaño de los enlaces, de este modo podremos aplicar el algoritmo correspondiente para realizar el cálculo y obtener la ruta.

Existen diversos algoritmos que nos permiten encontrar esta solución para un grafo dado.

5.2.1.1. ALGORITMO DE DIJKSTRA

[12] Este algoritmo, también conocido como algoritmo de caminos mínimos, fue propuesto por Edsger Dijkstra en el año 1956. Permite resolver el problema de la ruta más corta, desde un nodo inicial hasta otro final, para cualquier grafo dado siempre que cumpla las condiciones de grafo dirigido, ponderado y con pesos no negativos.

Las aplicaciones principales de este algoritmo están enfocadas, por ejemplo, a la hora de encontrar las rutas óptimas en una empresa de transporte, donde desde un punto inicial se obtienen las rutas más cortas a los domicilios finales.

Dado del grafo de la [Figura 5.1], el cual refleja en los enlaces, como peso, la distancia de ellos, se pretende encontrar el camino más corto partiendo del nodo 1 hasta el nodo 5. Para ello, en primera instancia podríamos considerar que el camino formado por los nodos: 1 > 2 > 5, con un peso de 10 unidades de distancia, sería el camino más corto, puesto que el otro camino está formado por un enlace más (tres enlaces en total).

Pero, no se obtiene la misma solución aplicando el algoritmo de Dijkstra puesto que este, al tener en cuenta el coste de estos enlaces y no el número de ellos, nos devolvería que el camino más corto es el formado por los nodos: 1 > 3 > 4 > 5, con un coste total de 8 unidades de distancia, 2 menos que la primera solución basada en el número de enlaces.

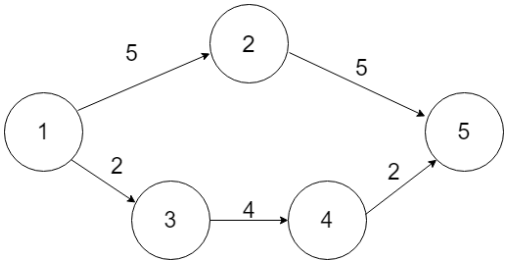

Figura 5.1. Grafo dirigido y ponderado Dijkstra.

5.2.1.2. ALGORITMO DE FLOYD-WARSHALL

[13] El algoritmo de Floyd-Warshall fue publicado en el año 1962 por Robert Floyd, y se basaba en otro algoritmo que en el año 1959 había sido citado por Bernard Roy.

Este algoritmo, al igual que el resto de los tratados en esta sección, busca el camino más corto para todos los pares de nodos o vértices de un grafo dado; esto lo realiza mediante el uso de metodologías de Programación Dinámica[10]. Para ello, partiendo de un grafo dirigido, ponderado, sin ciclos negativos y su correspondiente matriz de pesos o distancias de los enlaces, el algoritmo realiza iteraciones desde un nodo inicial *i* hasta un nodo final *j*, analizando los pasos por cada uno de los nodos intermedios. Una vez que se han probado todos los nodos del grafo, se obtiene una matriz con la menor distancia entre todo par de nodos.

5.2.1.3. ALGORITMO DE JOHNSON

El algoritmo de Johnson se usa como método para encontrar las rutas más cortas entre un par de nodos de un grafo dirigido, ponderado y en el que los pesos sí pueden ser negativos, a diferencia de los algoritmos de Dijkstra y de Floyd-Warshall. Este utiliza, como subrutinas, el algoritmo de Dijkstra, previamente visto, y el algoritmo Bellman-Ford[11]; y es más eficaz en grafos dispersos debido a que la complejidad temporal de este depende del número de aristas en el grafo.

5.2.1.4. ALGORITMO ELEGIDO PARA EL SISTEMA DE RECOMENDACIONES

Se ha considerado que el algoritmo óptimo para este sistema de recomendaciones, en este caso en el que se está buscando la ruta más corta, es el algoritmo de Dijkstra.

Este algoritmo ha sido considerado como óptimo pues, para el tipo de grafos con los que trabajaremos, es decir, grafos con vértices no negativos, es el que mejor se ajusta. En el supuesto de tener vértices negativos deberíamos optar por el algoritmo de Johnson o Bellman-Ford.

Además, la librería NetworkX ofrece un amplio abanico de funciones para obtener rutas más cortas con el algoritmo de Dijkstra, como se muestra en la [Tabla 5.1]. En este proyecto se usará el método bidirectional_dijkstra (G, source, target), el cual obtiene la

[10] La Programación Dinámica consiste es un método empleado con la finalidad de reducir el tiempo de ejecución de un algoritmo. Para ello se utilizan subproblemas superpuestos y subestructuras óptimas.
[11] El algoritmo de Bellman-Ford permite volver a ponderar el grafo de entrada, eliminar nodos negativos y detectar ciclos negativos.

longitud y los nodos que componen la ruta más corta desde un nodo origen o "source" hasta un nodo final o "target". Esta función toma por defecto el valor del campo "weight" de los enlaces para realizar los cálculos, por lo que para la búsqueda de la ruta más corta se tomará el valor longitud como peso o "weight" del enlace.

dijkstra_predecessor_and_distance(G, source)
dijkstra_path(G, source, target[, weight])
dijkstra_path_length(G, source, target[, weight])
single_source_dijkstra(G, source[, target, …])
single_source_dijkstra_path(G, source[, …])
single_source_dijkstra_path_length(G, source)
multi_source_dijkstra(G, sources[, target, …])
multi_source_dijkstra_path(G, sources[, …])
multi_source_dijkstra_path_length(G, sources)
all_pairs_dijkstra(G[, cutoff, weight])
all_pairs_dijkstra_path(G[, cutoff, weight])
all_pairs_dijkstra_path_length(G[, cutoff, …])
bidirectional_dijkstra(G, source, target[, …])

Tabla 5.1. Métodos para el algoritmo de Dijkstra de la librería NetworkX.

5.2.2. RUTA MÁS RÁPIDA

Aunque en una primera impresión se podría considerar que la ruta más corta es además la ruta más rápida, no siempre tiene por qué serlo, ya que en nuestro sistema, la acción de subir escaleras comparada con recorrer caminos planos implica un mayor gasto de tiempo.

Por ello, si partimos de que disponemos de varias rutas desde un nodo inicial hasta uno final, deberemos de considerar cuánto se tarda en recorrer cada parte de la ruta en función de los factores de tiempo que se aplican a cada enlace del recorrido. Por ejemplo, si una de ellas dispone de escaleras relativamente cortas, pero con unas inclinaciones elevadas en comparación a otras de mayor longitud, pero con unas elevaciones menores, puede ser que el tiempo de recorrer las escaleras más pronunciadas sea mayor y, por tanto, el sistema debe recomendar una ruta más larga, pero de mayor velocidad.

Además, existe un caso especial. Podríamos considerar que el tiempo empleado en recorrer una ruta también depende del estado de ocupación de los enlaces y por lo tanto el sistema debe tener en cuenta el número de personas que se encuentran en estos, valorando el tiempo necesario para transitar por los segmentos más concurridos. Pero, dado que este problema representa una de las características especiales del sistema[12], se ha decidido tratar de forma individual en el algoritmo *"*RUTA CON CONTROLES DE AGLOMERACIÓN*".*

Este algoritmo, por lo tanto, será una adaptación del problema de búsqueda de la ruta más corta, pero adaptando las distancias a la velocidad de recorrido. Es decir, si por ejemplo tenemos unas escaleras con una determinada inclinación, el peso del enlace será la distancia por una constante de velocidad, lo que nos permitirá comparar unos enlaces con otros.

5.2.2.1. CÁLCULO DEL PESO DE LOS ENLACES

Para realizar las matrices de representación de los tiempos o dificultades de los enlaces del grafo, con la finalidad de poder aplicar posteriormente el algoritmo del camino más corto, se han descrito los pasos necesarios para poder obtener los elementos de la matriz.

Partiendo del análisis realizado por el "*Department of Fire Protection Engineering*" de Corea del Sur [14], en el que se cita *"The average descent speed for the male and female population was 0.83 m/s and 0.74 m/s, repectively, while the average ascent speed was 0.66 m/s and 0.48m/s."*, se han obtenido, de forma general, los valores medios entre hombres y mujeres de velocidad de ascenso de escaleras de 0.79 m/s y de descenso 0.57 m/s.

Es importante señalar que como se trata en el artículo previamente referenciado, estos tiempos cambian en función de las diferentes variables, como son los grados de inclinación de las escaleras, la edad, el sexo, la altura del usuario, las condiciones climáticas, etc. Por ello, y debido a la complejidad de los cálculos a la que se sometería este trabajo, se ha decidido tomar estas variables de velocidad para un primer prototipo de la aplicación y dejando una vía de evolución. Es decir, en un uso real del sistema de recomendaciones se deberá hacer un estudio previo del estadio con el objetivo de poder crear un modelo real que se adapte al tipo de escaleras, a las condiciones meteorológicas cambiantes, a las características de las personas que utilizan las mismas, etc., obteniendo así un modelado

[12] El control de accesos en función de la aglomeración de personas, para problemas relacionados con la COVID-19, es uno de los objetivos específicos de este sistema.

más realista y unas variables más ajustadas, pero esto complicaría excesivamente el desarrollo de este prototipo.

Por ello, para nuestro sistema representaremos las matrices de tiempos de los enlaces en función de la fórmula $t_{enlace} = \frac{d_{enlace}}{v}$, donde t_{enlace} representa cada uno de los elementos de la matriz o pesos de los enlaces del grafo en segundos, d_{enlace} equivale a la distancia del enlace en metros y v será la variable previamente obtenida de 0.79 m/s para subidas de escaleras, 0.57 m/s para bajadas de escaleras y 1.56 m/s para caminos planos, las cuales se pueden variar en función de las condiciones de estas.

5.2.3. RUTA CON CONTROLES DE AGLOMERACIÓN

Uno de los algoritmos con más utilidad del sistema de recomendaciones consiste en el control del número de personas que circulan por un determinado tramo o pasillo del estadio. De esta forma, dispondremos de la posibilidad de recomendar las rutas con menor número de personas o equilibrar la cantidad de espectadores circulando por cada uno de los enlaces.

Este algoritmo presenta una gran importancia, pues en la fecha en la que se está realizando este documento el mundo está sumergido en una pandemia[13] provocada por el virus SARS-CoV-2 o coronavirus que afecta a la celebración de todos los eventos masivos de personas, como son partidos de fútbol, baloncesto, tenis, carreras de automoción o incluso cines u obras de teatro, todos ellos han quedado influenciados por este virus y se ha tenido que eliminar o reducir el acceso de espectadores a las diferentes gradas de estos eventos. Por ello, se pretende con este algoritmo crear una posible implementación de un mecanismo de control del tránsito de personas, sirviendo para futuras implementaciones donde se necesite regular el aforo u organizar los accesos del público, sin acumulaciones y evitando así la posible transmisión del virus por la presencia de gran cantidad de personas en espacios reducidos, como son los pasillos de un estadio.

Una de las analogías de esta idea es el control de tráfico utilizado por sistemas como Google Maps, Waze o Apple Maps, entre otros. Estos sistemas analizan en tiempo real el estado de las carreteras utilizando datos de los usuarios que circulan o han circulado por ellas; en otras

[13] El 11 de marzo de 2020, la Organización Mundial de la Salud declaró como pandemia mundial a la pandemia originada en la ciudad China de Wuhan en diciembre de 2019, provocada por el virus SARS-CoV-2 y que llevó a evitar o reducir los encuentros de personas de forma masificada debido al gran índice de transmisión de este germen por vías aéreas.

palabras, se usan tanto datos en tiempo real como registros históricos. De esta forma se puede obtener un modelo del plano de las carreteras con su ocupación y tiempo empleado en recorrer un determinado enlace, y a través de esto poder deducir y recomendar al usuario la ruta óptima en función del tiempo del recorrido, y no solamente derivado de la distancia de la ruta, sino también de la ocupación de las vías.

5.2.3.1. ANÁLISIS EN TIEMPO REAL

Una de las posibles soluciones para obtener los datos que dibujen un modelo de la ocupación de las vías del estadio consiste en el análisis de las personas mediante mecanismos de contadores en tiempo real. Esta solución se podría implementar a través de diferentes mecanismos.

5.2.3.1.1. MEDICIONES MEDIANTE SENSORES

La medición de personas mediante sensores consiste, como su propio nombre indica, en realizar mediciones a través de estos en el inicio y final de cada enlace o pasillo. De esta forma al activarse el sensor de entrada el sistema aumentará el contador de ese pasillo y en el lado opuesto, correspondiente a la salida del enlace, se restará uno al activarse el sensor. De este modo se pueden ir obteniendo los datos de la cantidad de personas en cada enlace del grafo.

Este tipo de mediciones es similar al utilizado en los sistemas de control de la ocupación de los *parkings* de coches, el cual mostramos representado en la [Figura 5.2], y como ya describimos el proceso consiste en que al pasar un coche por la puerta 1 de entrada *("Gate 1 Entry")* se aumenta la ocupación del *parking* y solo se reduce al salir un coche por la puerta 1 de salida *("Gate 1 Exit").*

Figura 5.2. Contador de coches por sensores.

5.2.3.1.2. MEDICIÓN POR RECONOCIMIENTO FACIAL

La medición de la ocupación mediante reconocimiento facial consiste en, partiendo de una imagen en tiempo real, obtener el número de personas que se observan en esta mediante mecanismos de Inteligencia Artificial.

En la actualidad existe una amplia cantidad de soluciones a este planteamiento, [17] el proyecto RetailNet ha diseñado un enfoque de aprendizaje profundo que tiene como objetivo realizar una estimación de la cantidad de personas en tiendas minoristas en tiempo real, además de la detección de puntos calientes o concurridos. Este sistema parte de una cámara RGB, como puede ser una cámara de video vigilancia, y mediante el procesado de estas imágenes en tiempo real se obtiene el conteo de personas.

Este tipo de implementaciones nos permitirían mediante dos cámaras, una al inicio y otra al final de cada enlace, obtener el conteo de personas para cada pasillo y de esta forma lograr un modelo en tiempo real de la ocupación en todas las parte del estadio.

5.2.3.1.3. MEDICIÓN POR GEOLOCALIZACIÓN

El uso de la geolocalización aplicada al control de aglomeraciones consistiría en, a través de dispositivos con GPS[14], tales como teléfonos móviles, tabletas, relojes inteligentes, etc., obtener la posición geográfica en tiempo real de cada uno de los portadores de estos aparatos. De esta forma sería posible dibujar el modelo del estadio con la ocupación de cada pasillo y, por consiguiente, realizar las recomendaciones de rutas según la cantidad de personas.

Un ejemplo de este tipo de aplicación es la herramienta creada por Google Maps y denominada *Google Traffic*. Este sistema, basado en el posicionamiento geográfico en tiempo real de los usuarios que circulan por vías públicas, permite crear "mallas" de colores superpuestas sobre las calles de las ciudades. En la [Ilustración 5.3] se puede apreciar el tráfico en tiempo real para una serie de calles de Nueva York; como la leyenda indica en el margen inferior, existen 4 colores que representan el tráfico de más rápido a más lento, esto está directamente relacionado con la cantidad de coches que están circulando por cada calle de la ciudad.

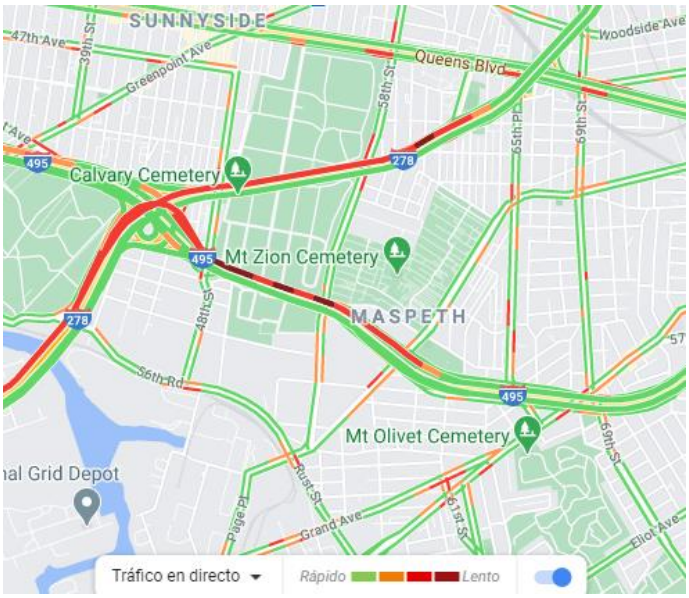

Ilustración 5.3. Tráfico en tiempo real de Google Maps en Nueva York.

[14] El GPS o *Global Positioning System* es un sistema basado en tres satélites que permite localizar geográficamente objetos en la Tierra.

[16] Como Dave Barth indica en la entrada de su blog, Google Maps obtiene datos en tiempo real de los usuarios que circulan por las calles. De esta forma, basándose en las posiciones geográficas en cada momento de los individuos puede calcular el tiempo empleado en desplazarse entre dos puntos (tales como el inicio y final de una calle). Además, Google Maps necesita conocer el tiempo normal empleado en circular por una calle. Por ello estos datos son almacenados y tratados con la finalidad de determinar cuánto tiempo se emplea de normal en recorrer un segmento del mapa. Así, comparando las variables de tiempo real (en el momento) y tiempo histórico se obtiene la clasificación del estado o fluidez de la calle.

5.2.3.2. ANÁLISIS DE DATOS HISTÓRICOS

El análisis de datos histórico es la principal alternativa a la obtención de la cantidad de usuarios en cada enlace mediante datos en tiempo real. Este tipo de análisis consiste, basándose en comportamientos pasados del acceso de personas a los estadios, en generar modelos que permitan predecir comportamientos futuros y de esta forma poder adivinar cómo serán los accesos a los estadios de todos los usuarios por los diferentes pasillos del sistema. De este modo se podrían realizar recomendaciones, distribuir de forma equilibrada a los espectadores y controlar las aglomeraciones en cada punto del estadio.

Si extrapolamos a las personas que acceden a los estadios de fútbol, podemos sacar la conclusión de que siempre tienden a acceder por la misma ruta o recorrido hasta llegar a su asiento, lo cual no quiere decir que esta ruta sea la óptima, ni en una situación normal ni en un estado con aglomeraciones, donde sería necesario controlar el tránsito de estos espectadores, pero sí podríamos deducir que el comportamiento repetitivo en el tiempo podría ser de utilidad para crear un modelo histórico de accesos, puesto que como se indicó anteriormente los accesos tienden a realizarse de forma similar en el tiempo.

Este tipo de datos que se pretenden analizar no están siendo capturados actualmente en gran parte de los estadios; es por ello que al igual que para los sistemas en tiempo real se deben crear mecanismos que nos permitan obtener estas mediciones con el objetivo de analizarlas posteriormente y tratarlas para obtener conclusiones.

El diseño e implementación de un posible sistema de captación de datos tanto en tiempo real como de datos histórico se sale de los límites de este Trabajo Fin de Grado; por

ello, se ha enfocado este apartado con la idea de servir como base a posibles implementaciones en un sistema real.

Por otro lado, sí se ha realizado una simulación de los accesos de las personas al estadio y, por consiguiente, de la ocupación de los enlaces. De este modo se pueden obtener recomendaciones basándonos en la ocupación del estadio en tiempo real.

Esta simulación parte de un estadio vacío y se simula la entrada de personas a este durante un periodo de tiempo[15]. Una vez transcurrido este tiempo se obtiene la ruta con menor ocupación desde el nodo inicial o puerta de entrada hasta el asiento, pero solo se toma el primer enlace de dicha ruta obtenida. De este modo durante el recorrido del pasillo o escaleras por parte del espectador se sigue llenando el estadio y, por lo tanto, las ocupaciones de los enlaces continúan variando. Cuando el espectador termina de recorrer el tramo se obtiene una nueva ruta recomendada con base en la ocupación de ese instante, y se repite el proceso de forma recursiva para los siguientes enlaces hasta llegar al nodo final.

La [Ilustración 5.4] muestra el Pseudocódigo del algoritmo usado para simular el proceso de recorrido del espectador en función de la ocupación.

[15] Para simular las personas que entran se toma un valor de 0 o 1 aleatorio. Si es 0 se restarán entre 0 y 10 personas (siempre que no baje de 1 la ocupación de un enlace); por otro lado, un valor de 1 sumará entre 0 y 10 espectadores al enlace.

```
Algorithm 1: Simulación de la ruta según la ocupación del estadio.
nodoActual = nodoInicial
while tiempo de rellenado do
  | Rellenar la ocupación de forma aleatoria
end
while nodoActual != nodoFinal do
  | //Se obtiene el siguiente nodo desde el nodo Actual hasta
  |  el nodo Final según la ocupación actual
  | siguienteNodo = Dijkstra(nodoActual, nodoFinal)[1]
  | while tiempo de caminar enlace do
  |  | Se simula el caminar y se rellena la ocupación de forma aleatoria
  | end
  | //Se borra el enlace para evitar ciclos
  | Borrar enlace recorrido
  | nodoActual = siguienteNodo
end
//En este punto se llego al nodo Final.
```

Ilustración 5.4. Pseudocódigo de la simulación de las rutas según la ocupación.

5.2.4. RUTA SEGÚN REQUISITOS PERSONALES

El algoritmo de selección de la ruta en función de los requisitos personales del espectador tiene como objetivo brindar la ruta obtenida por el sistema con base en las preferencias o necesidades de cada persona que acceda al estadio. Se partirá de un perfil de usuario en el que se almacenan una serie de preferencias.

Para ello el sistema mostrará al usuario un panel con una lista de características[16] a puntuar del 0 al 5[17]:

- Pasillos con barandillas.
- Pasillos en buen estado.
- Pasillos amplios.
- Pasillos ventilados.
- Seco en caso de lluvia.
- Pasillos bien iluminados.

[16] Estas características pueden ampliarse en el sistema, pero en este trabajo se han considerado las mencionadas.
[17] Donde 0 equivale a menos importancia y 5 a importancia máxima.

De este modo, para obtener los pesos de los enlaces del grafo que cumplen con los atributos que el usuario desea y generar con ellos la ruta a recomendar se emplea la siguiente fórmula[18]: $F_i = 1 - \frac{k_i}{6}$, donde para cada atributo *i* se calcula su factor de peso en el enlace F_i. k_i representa la puntuación de 0 a 5 que el usuario le ha dado al atributo. Si un enlace no cumple un atributo, por ejemplo, no tiene escaleras, el factor F_i será 1 y por lo tanto no influirá en el peso del enlace.

Estos factores F_i serán multiplicados junto a la longitud del enlace y de esta forma se obtendrá el peso del enlace: $Peso_{enlace} = longitud_{enlace} * \prod_{i=1}^{6} F_i$

De esta forma si un atributo tiene una puntuación, por ejemplo, de 5 obtendrá un factor F_i reducido, lo que implicará que al multiplicarlo por la longitud del enlace se minimizará este peso, convirtiéndolo así en un enlace con mayor posibilidad de ser recomendado dado a que es interesante para el usuario. Ocurriendo lo contrario en el caso de una puntuación de 0, lo cual indicaría que no es atrayente para el usuario.

Para explicar este cálculo se ha realizado el siguiente ejemplo sobre el grafo de la [Figura 5.3].

Supongamos que la ruta a recomendar es la que transcurre entre los nodos 1 y 3. En el caso de la ruta más corta será la ruta de longitud 17 formada por el enlace 1-3. Pero ¿Y si el usuario desea que su recomendación tenga escaleras con barandillas? En un primer lugar podríamos considerar borrar los enlaces con nodos sin barandillas y hacer la recomendación, pero en este caso no existiría una ruta del nodo 1 al 3 con escaleras con barandillas. Por ello se realiza el cálculo del peso del enlace en función de los atributos, en este caso del atributo "Barandillas", además del valor longitud, por lo que, si por ejemplo el usuario le diera una importancia de 1 a que tengan barandillas y existe una ruta mucho más corta, pero sin barandillas, se recomendará la de menor longitud.

[18] Se utiliza 6 como divisor en la función F_i ya que los valores del 0 al 5 son 6 componentes y, además, en el caso de que el usuario eligiese una importancia de 5 y se dividiera entre 5 nos daría un $k_i = 0$, lo cual al multiplicarlo junto a la longitud del enlace daría peso 0. Algo que el sistema no permite a los nodos.

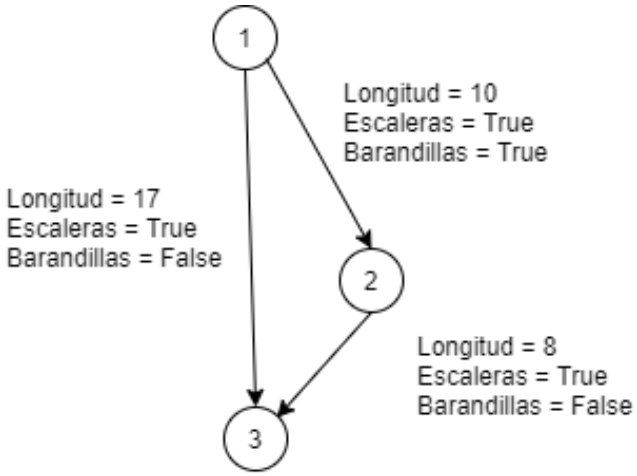

Figura 5.3. Grafo con atributos.

En el supuesto de que el usuario considerase que la importancia de la ruta tenga escaleras con barandillas fuese 4 sobre 5, tendríamos un factor para el enlace 1-2; $F_{1-2\ (Escaleras\ con\ barandillas)} = 1 - \frac{k_{1-2\ (Escaleras\ con\ barandillas)}}{6} = 1 - \frac{4}{6} = 0.\hat{3}$ y el peso del enlace 1-2, el único enlace que cumple tener escaleras con barandillas, sería de $10\ x\ 0.\hat{3} \cong 3$.

De este modo el grafo resultante sería el mostrado en la [Figura 5.4], y la ruta recomendada en este caso sería la formada por los enlaces 1-2 y 2-3, con un peso en conjunto de 11, frente a los 17 que presenta el enlace recomendado por el algoritmo de ruta más corta.

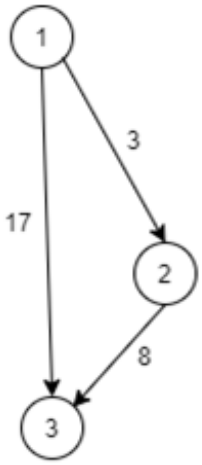

Figura 5.4. Grafo con pesos modificados en función de los atributos.

Este algoritmo, por lo tanto, deberá en primer lugar obtener los datos de preferencias del espectador y los atributos de cada posible enlace a recomendar dentro del estadio. Posteriormente se obtendrán los factores para cada enlace en función de estas preferencias y se recalcularán los pesos del grafo.

El grafo del estadio será almacenado en un fichero de tipo JSON, el cual contiene a su vez los atributos de cada enlace; de este modo será fácil para el sistema obtener los valores de un enlace ya que se encontrarán almacenados de forma organizada gracias a la estructura clave/valor de estos ficheros.

Una posible alternativa al uso de ficheros JSON para el almacenamiento del grafo y de los atributos del sistema consiste en la implementación de bases de datos no relacionales o NoSQL. Estas bases de datos están indicadas para sistemas que necesiten gran volumen de datos, con accesos con baja latencia y datos flexibles. En un estadio real el número de pasillos o enlaces que este tiene es elevado, por lo que sería necesario almacenar gran cantidad de datos; por otro lado, el sistema consulta un gran número de enlaces para hacer las recomendaciones, por lo que una baja latencia a la hora de obtener los datos puede reducir el tiempo de ejecución. Por ello el uso de bases de datos no relacionales, tales como MongoDB, Cassandra o Redis, podría ser una gran alternativa para una implementación de un sistema basado en un estadio real.

5.2.5. SELECCIÓN DEL NODO FINAL

Un aspecto muy importante para tener en cuenta en el sistema consiste en la correcta definición del nodo final.

En una visión global del sistema podemos identificar el nodo final como el asiento de cada usuario, pero a la hora de programar el sistema de recomendaciones que nos devolverá la ruta para el espectador debemos definir un nodo final previo. Es decir, definiremos un nodo en el grafo que sea la proyección horizontal del asiento en el pasillo.

Para ello, antes de la generación de la ruta desde la puerta de entrada hasta el asiento se creará un nodo, que será nuestro nodo “final” de la recomendación o destino, pero que

definiremos como *nodo* prefinal, ya que faltaría recorrer desde este hasta el asiento, lo cual se representará como una línea o camino horizontal[19] entre ambos.

A continuación, se mostrará un ejemplo en el cual se reflejará la obtención de *nodo prefinal* previamente descrito.

Como se aprecia en la [Ilustración 5.5], nos encontramos con un asiento (marcado con un triángulo amarillo) que corresponde al asiento final de un espectador del sistema. Por lo tanto, el sistema recomendará un camino o ruta desde el nodo inicial (puerta de entrada al estadio) hasta este asiento. Fijándonos en el resto de asientos del sector podemos definir este como el asiento ubicado en la fila 11, columna 6, es decir, se encuentra en la mitad izquierda del sector. Por lo tanto, este asiento deberá ser proyectado al enlace izquierdo, formado por los vértices 1 y 2, y formará dos nuevos enlaces 1-NP y 2-NP, donde NP correponde al nodo prefinal, tal y como se muestra en la [Ilustración 5.6].

Por lo tanto, para elegir este nodo final se proyectará el asiento hacia el lado izquierdo o derecho más proximo, y en caso de no existir dicho enlace se proyectará hacia el lado que si exista.

[19] Debe ser una unión horizontal puesto que en los estadios los asientos se organizan en filas horizontales y, por lo tanto, para acceder a un asiento el espectador llega hasta la fila donde se ubica este y camina horizontalmente hasta su asiento.

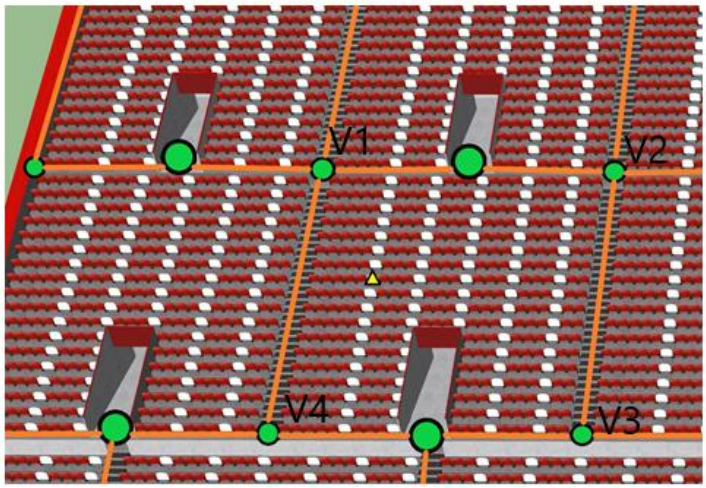

Ilustración 5.5. Pasos para la selección del nodo prefinal.

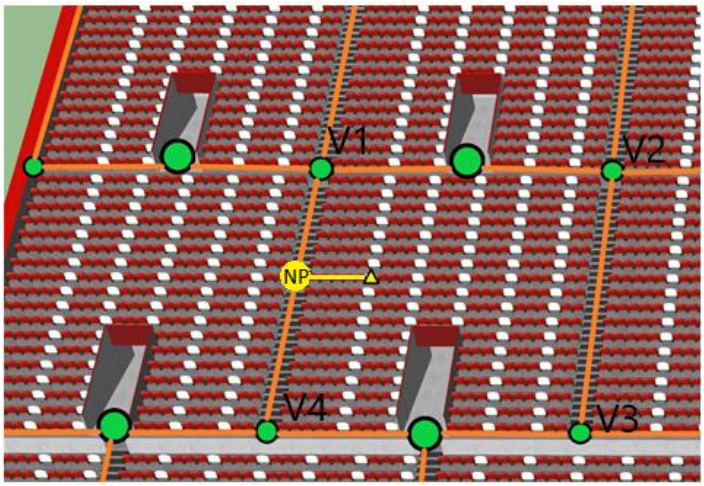

Ilustración 5.6. Nodo prefinal (NP) insertado.

5.3. ENTIDADES

Para la realización de este sistema de recomendaciones se ha optado por tratar cada componente o entidad del sistema como un objeto con una serie de atributos o características, lo que nos permitirá tanto programar como realizar los cálculos del sistema de manera más rápida y sencilla.

Por ello se han definido las entidades que se describen a continuación.

5.3.1. ESPECTADOR

Los espectadores son uno de las componentes principales del sistema y presentan una serie de características y comportamientos entre todos ellos.[20]

Se ha creado esta clase con la intención de representar a las distintas personas que acceden al estadio para ver un partido.

Se han definido los siguientes atributos para todos los espectadores:

- **ID**: Un identificador único para cada espectador que accede al estadio; de este modo se podrá identificar a cada espectador dentro del estadio y a través de este obtener el resto de los atributos. Este ID podría ser, en el caso de las personas con entradas, el número identificativo de esta, ya que este valor es único para cada partido y entrada. También podría ser el número de socio para los abonados con carnet.

- **Asiento**: La finalidad de todos los espectadores es encontrar su asiento; por ello se debe precisar cuál es su asiento, siendo único para todo el estadio, lo que implica que únicamente se defina un asiento para un espectador.

- **Puerta de acceso**: Debido a los diferentes protocolos de acceso a los estadios de fútbol por cada entidad deportiva la entrada se puede realizar por una o varias puertas. Por ello es primordial disponer de una variable que especifique por cuál o cuáles de las puertas del estadio puede entrar el espectador. Esta variable mostrará una gran influencia en la selección de la ruta óptima, dado que corresponderá al nodo inicial de esta.

5.3.2. ESTADIO

Un estadio de fútbol está compuesto por diferentes componentes, parte de los cuales describiremos a continuación, y que son los que permiten distribuir y organizar los asientos para los espectadores. El estadio que vamos a describir es el estadio El Molinón – Enrique Castro “Quini”, de Gijón, Asturias.

[20] En un futuro se podría adaptar el sistema de forma que los espectadores puedan interactuar entre ellos en tiempo real, permitiendo así tanto que modifiquen como que influyan en la selección de la ruta óptima de otro espectador.

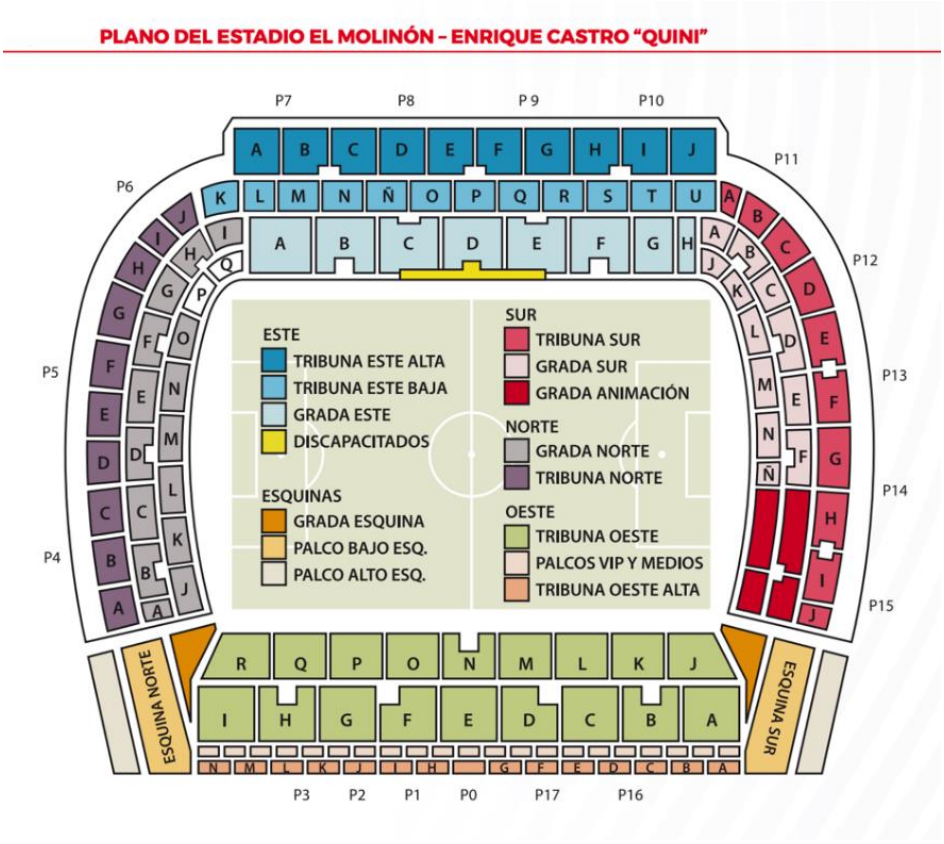

Ilustración 5.7. Plano del estadio El Molinón - Enrique Castro "Quini".

5.3.2.1. GRADA

En relación con la orientación a la grada se le atribuye un nombre a esta. De este modo, como se puede comprobar en la [Ilustración 5.7], dispondremos de 4 gradas[21]: Grada Norte, Grada Sur, Grada Este y Grada Oeste[22].

De esta forma, al recibir el sistema un espectador buscando su asiento podremos ubicarlo en una grada en función de la puerta de acceso asignada al espectador, y dado que cada

[21] La [Ilustración 5.7] también muestra las gradas llamadas esquinas, pero para esta explicación se considera que las esquinas están juntadas a la Grada Oeste.
[22] El concepto de grada en la [Ilustración 5.7] corresponde a la parte baja de la grada, pero se ha considerado que las 4 gradas definidas son la unión de las tribunas, gradas, palcos, etc., y que el nombre que recibe la grada resultante está dado por la orientación del estadio.

grada dispone de una serie de puertas con un identificador único el sistema debe almacenar la relación de puerta o puertas y grada.

En este estadio nos encontramos con que las gradas se dividen de forma vertical, esto es, por alturas, como es el caso de la grada Este, que tiene tres, y que son conocidas como Grada Este, Tribuna Este Baja y Tribuna Este Alta. Esto se repite en otras gradas, diferenciando normalmente la grada como la parte más baja y la tribuna como la parte más alta.

Se han definido los siguientes atributos para las gradas:

- **Nombre**: En nuestro caso, las gradas se nombrarán con los puntos cardinales, Grada Norte, Grada Sur, Grada Este y Grada Oeste.
- **Puertas**: Corresponderá a un vector que contendrá las puertas de que dicha grada dispone.
- **Alturas:** Corresponderá a un vector que contendrá las alturas o capas de que dicha grada dispone, de este modo se podrá relacionar un sector (que se definirá más adelante) con una grada.

5.3.2.2. SECTORES

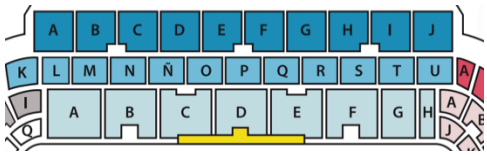

Ilustración 5.8. Sectores de la Grada Este de El Molinón - Enrique Castro "Quini".

Ya hemos comentado que una grada se distribuye en alturas formando capas. En la [Ilustración 5.8] podemos ver diferenciadas tres capas y que dentro de cada una de estas capas se pueden observar sectores, como son A, B, C, D, E, F, G, H en la capa más baja, y que delimitan los grupos de asientos.

Por lo tanto, cada capa de altura dispone de sectores identificados con letras de forma única, pero negativamente para este estudio esta nomenclatura no es única para todas las

capas de las gradas, por lo que únicamente con el sector no podemos ubicar el asiento del espectador; por ello se necesitará la combinación de la grada, la capa y el sector para ubicar el asiento en el estadio.

Estos sectores contienen los asientos de los espectadores, los cuales son las unidades mínimas y que corresponden con los nodos finales de nuestro sistema de recomendaciones. Estos asientos se agrupan en filas y columnas dentro del sector.

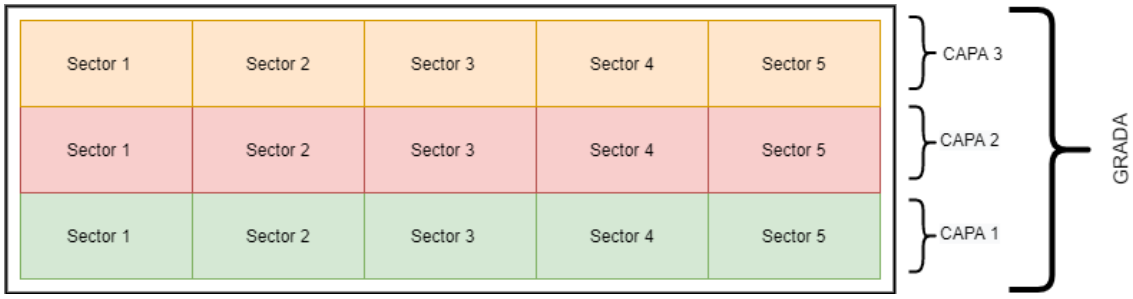

Figura 5.5. Esquema de una grada genérica y sus correspondientes capas y sectores.

La [Figura 5.5] muestra el esquema de una grada cualquiera diferenciando las partes en las que se divide. Podemos apreciar como hay 3 capas o niveles (colores verde, rojo y amarillo) y en cada uno de ellos una serie de sectores (numerados del 1 hasta el 5) que, en este caso, repiten el nombre para la grada pero son únicos para cada capa.

5.3.2.3. ASIENTO

Un asiento es la ubicación a la que se dirige el espectador cuando accede al estadio, o lo que es lo mismo será nuestro nodo final “absoluto”.

Los asientos se ubican en los sectores de la grada y se organizan en columnas y filas. Por ello, dentro de un sector se puede ubicar un asiento contando filas y columnas de izquierda a derecha y de abajo hacia arriba.

Se han definido los siguientes atributos para los asientos:

- **Sector:** El sector será equivalente a una "matriz"[23] de n filas y m columnas. Además, el asiento estará ubicado en la posición ($n_{fila}, m_{columna}$).
- **Fila:** Será la posición vertical dentro del sector.
- **Columna:** Será la posición horizontal dentro del sector.

5.3.2.4. PASILLOS

Los pasillos corresponden a uno de los principales componentes del sistema de recomendaciones dado que, en el fin del sistema, se recomendará seguir una ruta, que en definitiva lo que supone es ir por un pasillo o por otro.

De este modo, los pasillos equivalen a los enlaces entre los distintos nodos del estadio formando así el grafo que contiene las diferentes rutas que puede tomar el espectador hasta llegar a su asiento.

Cada pasillo presentará una serie de características o atributos, tales como el tipo de pasillo, su longitud, su inclinación, etc. Estos valores permitirán al sistema ofrecer recomendaciones personalizadas para cada espectador.

Se han definido los siguientes atributos:

- **origen:** Nodo origen.
- **destino:** Nodo destino.
- **longitud:** Unidades de longitud del pasillo.
- **tipoEnlace:** Escaleras / Pasillo plano.
- **barandillas:** Booleano para definir si tiene barandillas.
- **pasilloBuenEstado:** Booleano para definir si está en buen estado**.**

[23] A diferencia de una matriz normal, en una matriz de asientos no se cuenta de arriba hacia abajo y derecha, sino que se cuentan las posiciones desde la esquina izquierda inferior hacia la derecha superior.

- **pasilloAmplio:** Booleano para definir si es amplio.
- **pasilloVentilado:** Booleano para definir si está ventilado.
- **pasilloSeco:** Booleano para definir si se moja cuando llueve.
- **pasilloIluminado:** Booleano para definir si está bien iluminado.
- **ocupacion:** Valor de ocupación inicial.

5.4. MODELO DE UNA GRADA DEL ESTADIO

Con el fin de tener una visión más amplia del sistema se ha realizado el siguiente diseño gráfico en 3D utilizando la herramienta SketchUp [11]. Este diseño nos permitirá, de una forma más sencilla, modelar y crear los grafos que componen el sistema, a la par que nos facilitará visualizar el entramado que componen las diferentes partes de una grada en un estadio de fútbol, ya que este diseño pretende asemejarse a una grada real de un estadio actual.

La [Ilustración 5.9] representa la parte trasera de un estadio inventado, en ella encontramos 3 puertas, cada una de ellas numeradas en este caso del uno al tres, y que en nuestro grafo corresponderán a un nodo inicial. Es decir, un espectador comenzará su recorrido por el grafo únicamente por una de estas puertas y, por lo tanto, no podrá utilizar ninguna de las otras como acceso al estadio.

Mediante la [Ilustración 5.10 e Ilustración 5.11] podemos ver la parte principal de la grada. En ella podemos diferenciar los sectores, separados por escaleras (verticales) y pasillos (horizontales). Por otro lado, podemos visualizar las “bocas” que dan acceso desde el interior de la grada al exterior de esta y que comunican con los pasillos.

Ilustración 5.9. Vista trasera de una grada de un estadio de fútbol.

Ilustración 5.10. Vista frontal de una grada de un estadio de fútbol.

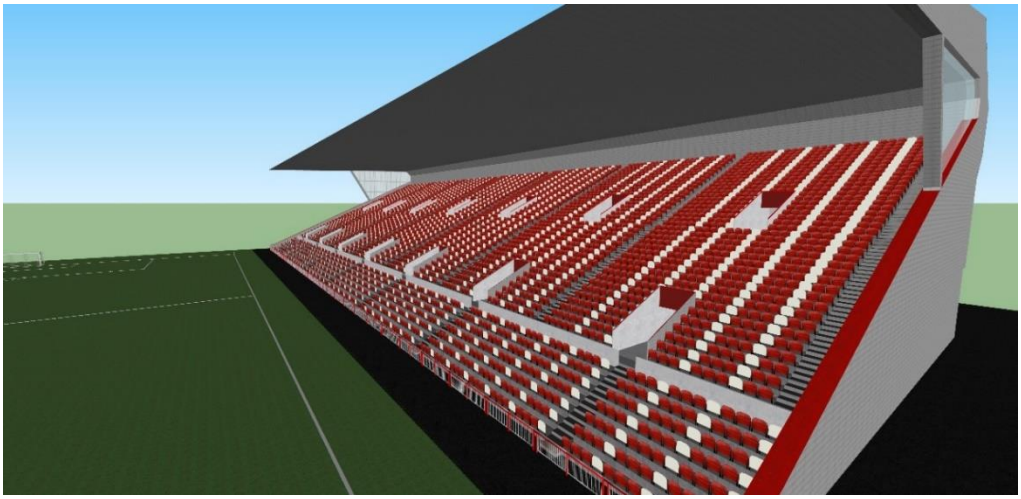

Ilustración 5.11. Vista oblicua de una grada de un estadio de fútbol.

Con el objetivo de poder visualizar más en detalle las estructuras del estadio se han dibujado y etiquetado, sobre una vista frontal, los diferentes sectores, nodos y enlaces que el grafo tendrá en esta parte [Ilustración 5.12]. De este modo resultará más sencillo diseñar el grafo y entender la relación de las representaciones matriciales del grafo con el modelo real.

La [Ilustración 5.13] representa el conjunto de enlaces y nodos que formarán parte del grafo final de la grada. Así, podemos observar cómo los diferentes pasillos y escaleras del estadio equivalen a los enlaces del grafo y serán los posibles caminos por los cuales los espectadores podrán transitar. También podemos observar cómo cada unión de escaleras con pasillos origina un nodo; en estos puntos se podrá optar por tomar un camino u otro y de este modo cambiarán las diferentes recomendaciones del sistema y las rutas del espectador.

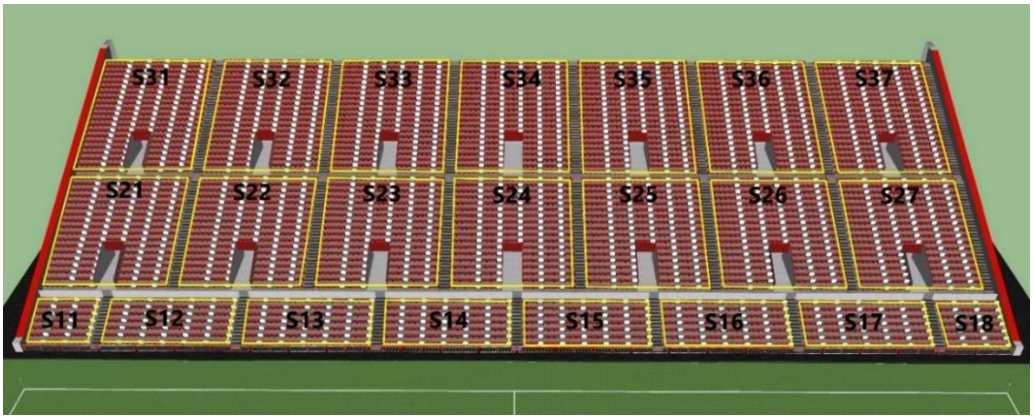

Ilustración 5.12. Vista frontal con los sectores sobreimpresionados.

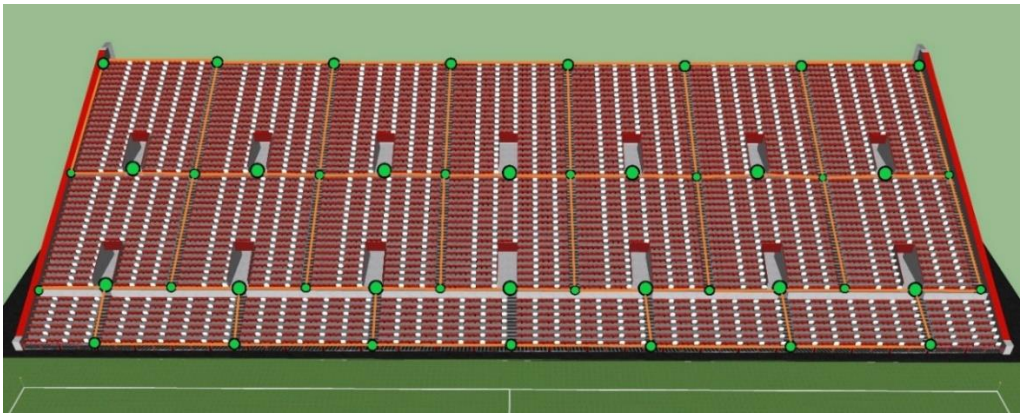

Ilustración 5.13. Vista frontal con enlaces y nodos sobreimpresionados.

5.4.1. SUBMODELO DE UNA GRADA DEL ESTADIO

Con el objetivo de desarrollar un grafo sobre el que trabajar se ha optado por dividir el modelo de la grada del estadio creada. De este modo es posible crear un grafo más pequeño con el que trabajar y crear el sistema, y posteriormente sería posible adaptar el código para toda la grada del modelo.

La [Ilustración 5.14] muestra la grada de este submodelo con la que se trabajará, siendo la parte sin sombrear y delimitada por las líneas azules la sección que se utilizará para crear el grafo, el cual se aprecia en la [Figura 5.6].

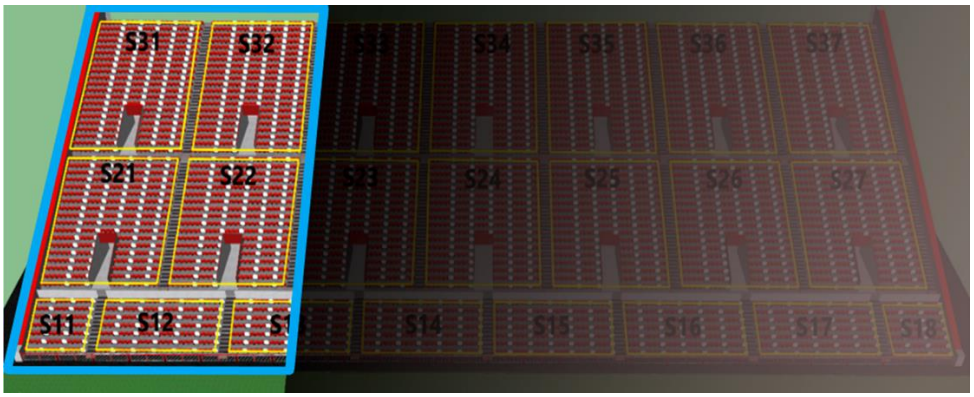

Ilustración 5.14. Submodelo de la grada empleada por el programa.

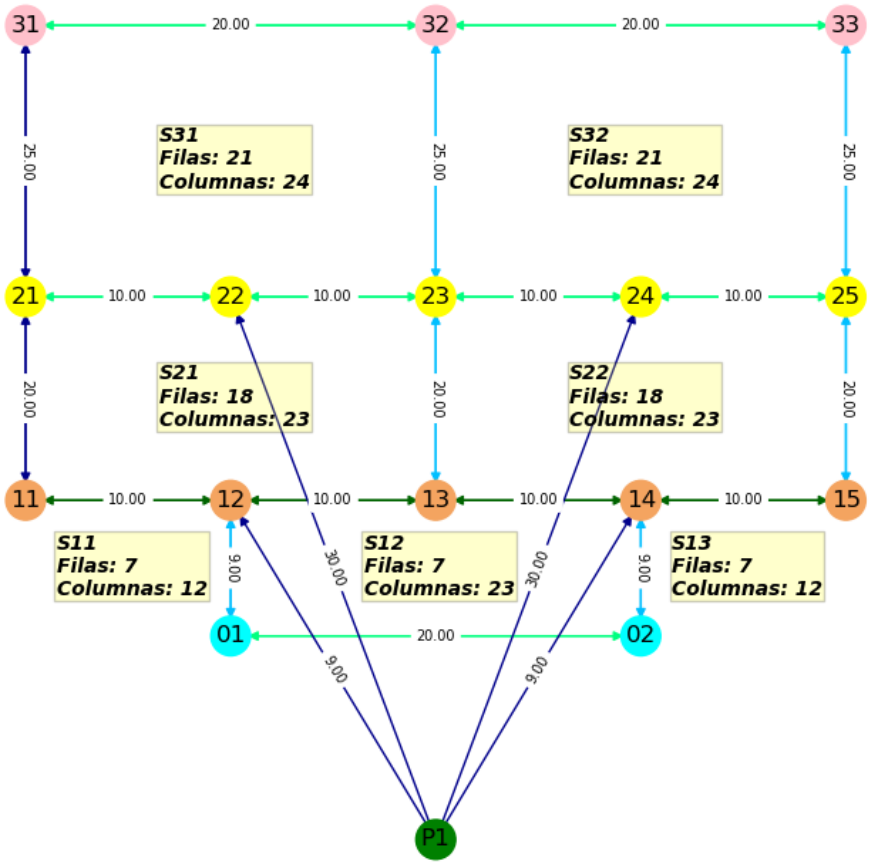

Figura 5.6. Grafo del submodelo utilizado por el programa.

6. CÓDIGO

El código del proyecto se encuentra alojado en la plataforma GitHub, la cual ha servido como herramienta de control de versiones durante la realización de este código. Enlace al repositorio: https://github.com/uo251345/TFG---Sistema-de-recomendaciones-para-accesos-a-estadios-de-futbol

Por otro lado, se ha creado una documentación asociada a este código; en ella se muestran las instrucciones de instalación y de uso del programa, así como las definiciones y especificaciones de los distintos métodos, clases y librerías que son utilizadas por el código del programa. Esta documentación se puede localizar en el Apéndice – DOCUMENTACIÓN DEL CÓDIGO o en la versión web de la API accesible a través del enlace: https://sistema-de-recomendaciones-para-accesos-a-estadios-de-futbol.readthedocs.io/es/latest/

7. EJECUCIÓN Y RESULTADOS

A continuación se mostrarán los resultados obtenidos mediante las distintas ejecuciones del sistema.

7.1. DATOS INCIALES DEL ESPECTADOR (ENTRADA)

Con la finalidad de ofrecer una visualización más realista del funcionamiento del código, así como de las salidas producidas por este, se ha creado un ejemplo de una entrada[24] [Ilustración 7.1] que el espectador recibiría y que será introducida al programa como argumentos, tal y como se mostrará a continuación.

Ilustración 7.1. Ejemplo de una entrada del espectador.

7.2. GRAFO GENERAL

Al inicio de la ejecución del programa se muestra una representación de la grada o *Grafo General* [Ilustración 7.2], en la cual se pueden apreciar todos los sectores que la componen, así como también el conjunto de nodos y enlaces; en estos últimos podemos encontrar reflejada la distancia que el espectador debería recorrer al transitar por cada uno de ellos. Además de manera informativa se muestran los enlaces pintados en diferentes colores, tal y como indica

[24] Un carnet sería análogo a una entrada, ya que también tendríamos la puerta de entrada, sector, fila y columna del asiento.

la leyenda, resultando de este modo más fácil comprender tanto la distribución de los pasillos en la grada como el motivo de las rutas recomendadas producidas por el sistema.

Cabe destacar que esta primera ventana tiene una funcionalidad más enfocada a la depuración del código que a servir de información para el usuario, puesto que en un sistema real el espectador que accediera al estadio y quisiera obtener una recomendación de su ruta partiría de los datos de su entrada y únicamente esperaría obtener una ruta final sin tener en cuenta cómo se estructura el estadio.

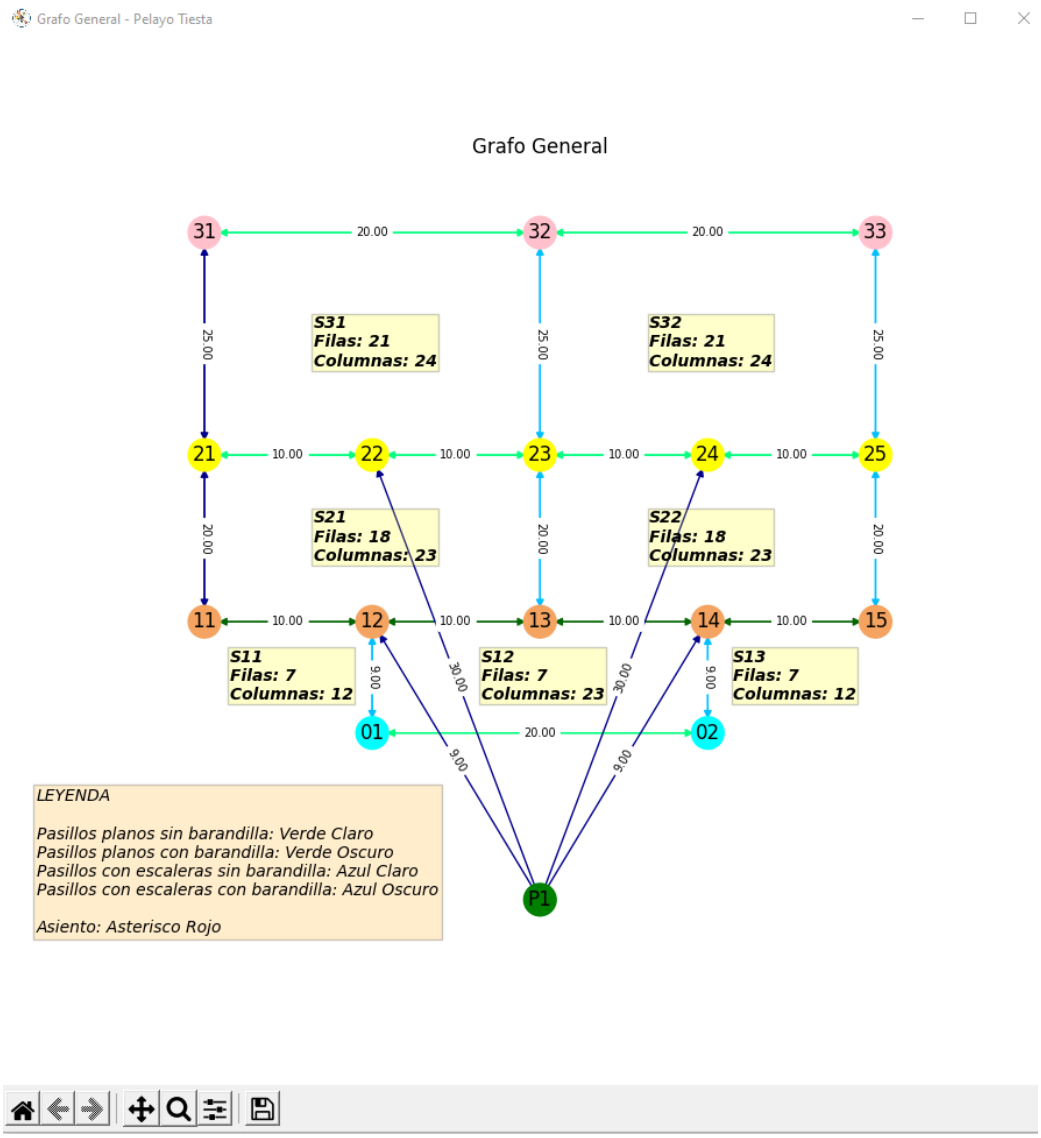

Ilustración 7.2. Grafo General de la grada.

7.3. LECTURA DE LOS DATOS DEL ESPECTADOR

La [Ilustración 7.3] muestra cómo el sistema ha solicitado al usuario que introduzca los datos de la entrada para poder ubicar el asiento dentro del *Grafo General*. Estos datos corresponden a los representados en la [Ilustración 7.1].

En el caso de que el usuario introduzca datos no válidos, como por ejemplo una puerta de entrada o un sector que no existe, se muestran mensajes indicando el problema, tal y como se puede apreciar en la [Ilustración 7.4].

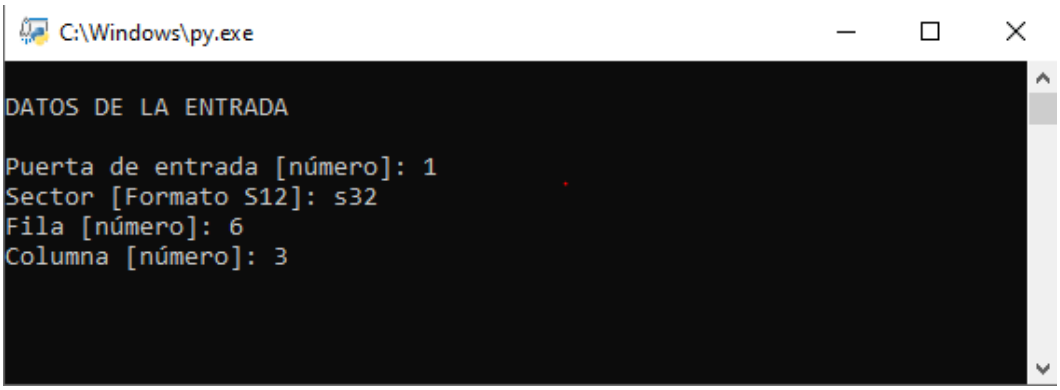

Ilustración 7.3. Datos de la entrada en el programa.

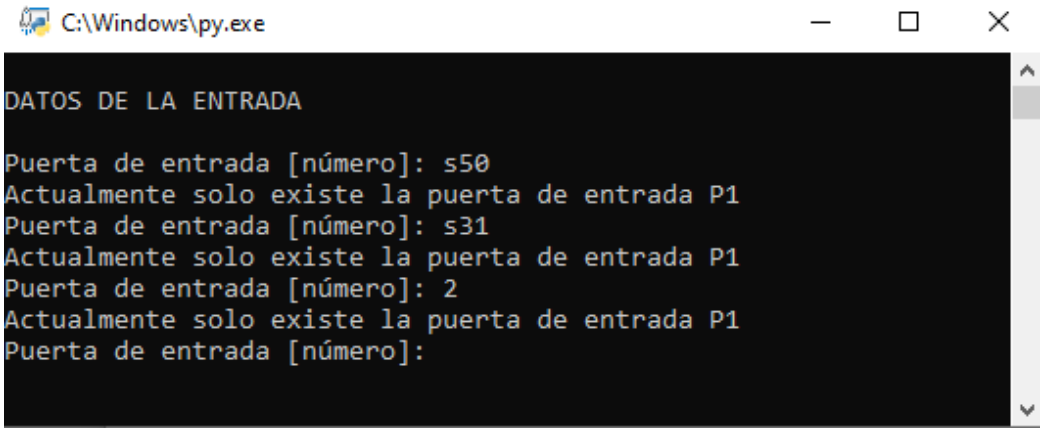

Ilustración 7.4. Información de errores en la lectura de los datos.

7.4. ACTUALIZACIÓN DEL GRAFO GENERAL CON EL ASIENTO

Una vez que el usuario ha introducido de forma correcta todos los datos al sistema, el flujo de la ejecución continuará para mostrar el asiento en la figura del *Grafo General*.

Como resultado se obtiene la ventana mostrada en la [Ilustración 7.5], la cual, a diferencia de la [Ilustración 7.2], indica los datos del espectador (ID, puerta de entrada y nodo final) y, además, añade el asiento en la figura (representado con una estrella roja).

También podemos apreciar cómo el *Grafo General* ha variado, se ha borrado el enlace entre los nodos "23" y "32", se ha añadido un nodo denominado "NP" o nodo prefinal y se han creado dos enlaces nuevos entre este y los nodos "23" y "32". Este proceso corresponde al algoritmo descrito en el apartado SELECCIÓN DEL NODO FINAL.

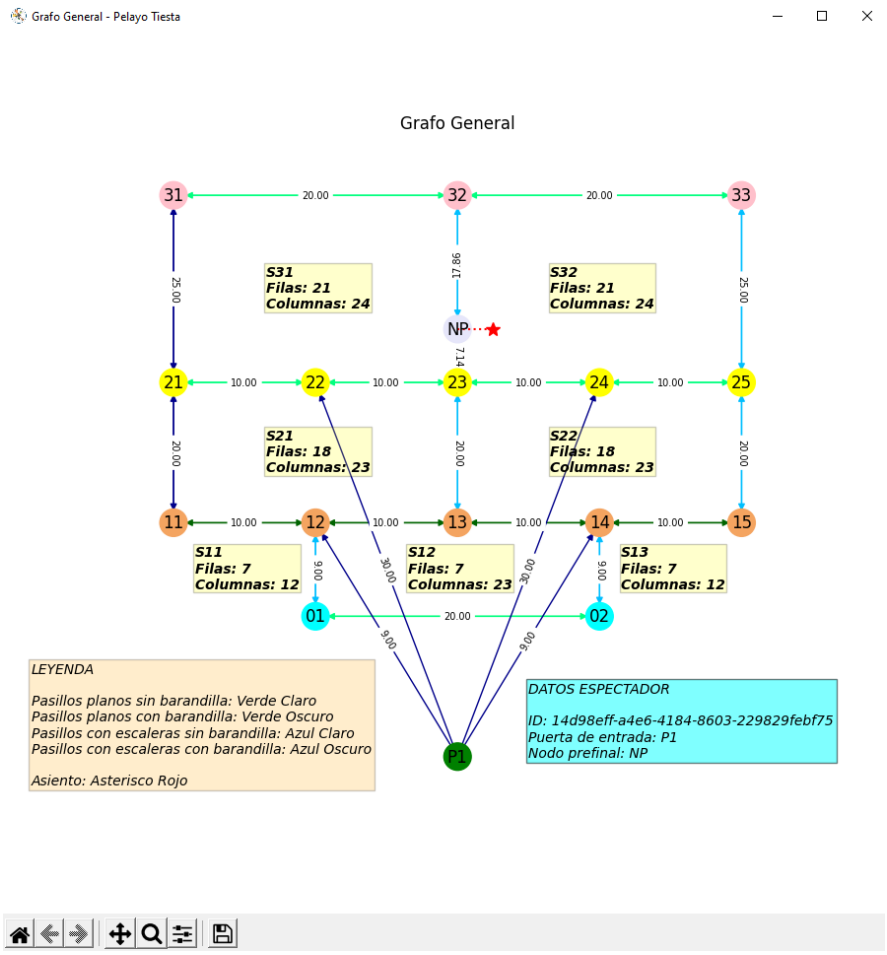

Ilustración 7.5. Grafo General con asiento y nodo Prefinal.

7.5. SELECCIÓN DE LA RUTA RECOMENDADA

En este punto el usuario puede seleccionar cuál es la ruta que desea obtener entre un menú de 4 opciones [Ilustración 7.6]:

- Ruta más corta.
- Ruta más rápida.
- Ruta según atributos.
- Ruta con control de aglomeraciones[25].

Cada una de las opciones ofrece una salida distinta, ya que cada una de ellas aplica un algoritmo distinto.

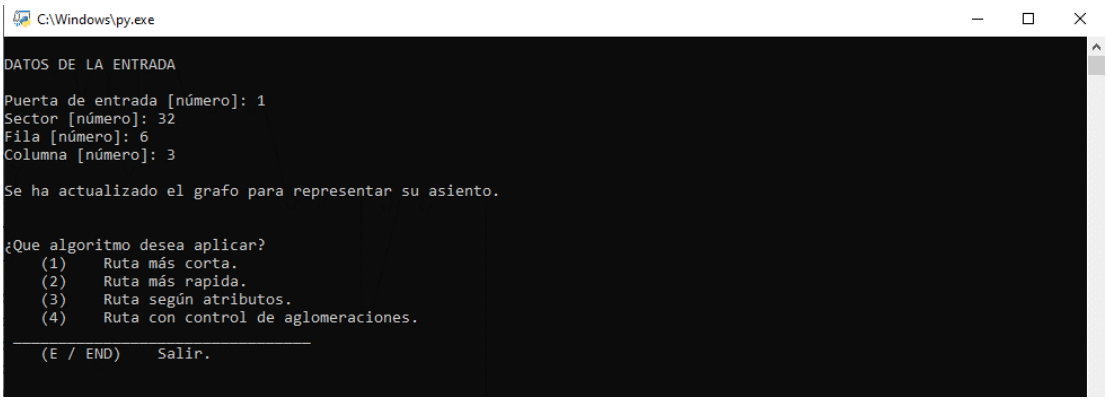

Ilustración 7.6. Menú de selección de la ruta a recomendar.

7.6. RUTA MÁS CORTA

La opción de ruta más corta obtiene el recorrido desde el nodo inicial o puerta de enlace hasta el nodo prefinal minimizando la distancia recorrida por los enlaces.

[25] En el caso de las rutas con control de aglomeraciones no se obtiene una ruta al instante, sino que se realiza una simulación donde el usuario camina por el estadio y va recibiendo indicaciones sobre la ruta a tomar.

Para realizar estos cálculos el algoritmo rellena el peso de cada enlace con el valor del atributo longitud de estos. De este modo, al aplicar el algoritmo de Dijkstra se obtiene la ruta más corta en función de las longitudes.

El programa, por lo tanto, mostrará al usuario en una ventana la ruta recomendada sobreimpresionada en el *Grafo General* con los enlaces en color verde, tal y como muestra la [Ilustración 7.7].

Por lo tanto, para este espectador el recorrido más corto a su asiento sería la ruta [P1, 12, 13, 23, NP], con una longitud total de $9 + 10 + 20 + 7.14 = 46.14\ \ unidades\ de\ longitud$.

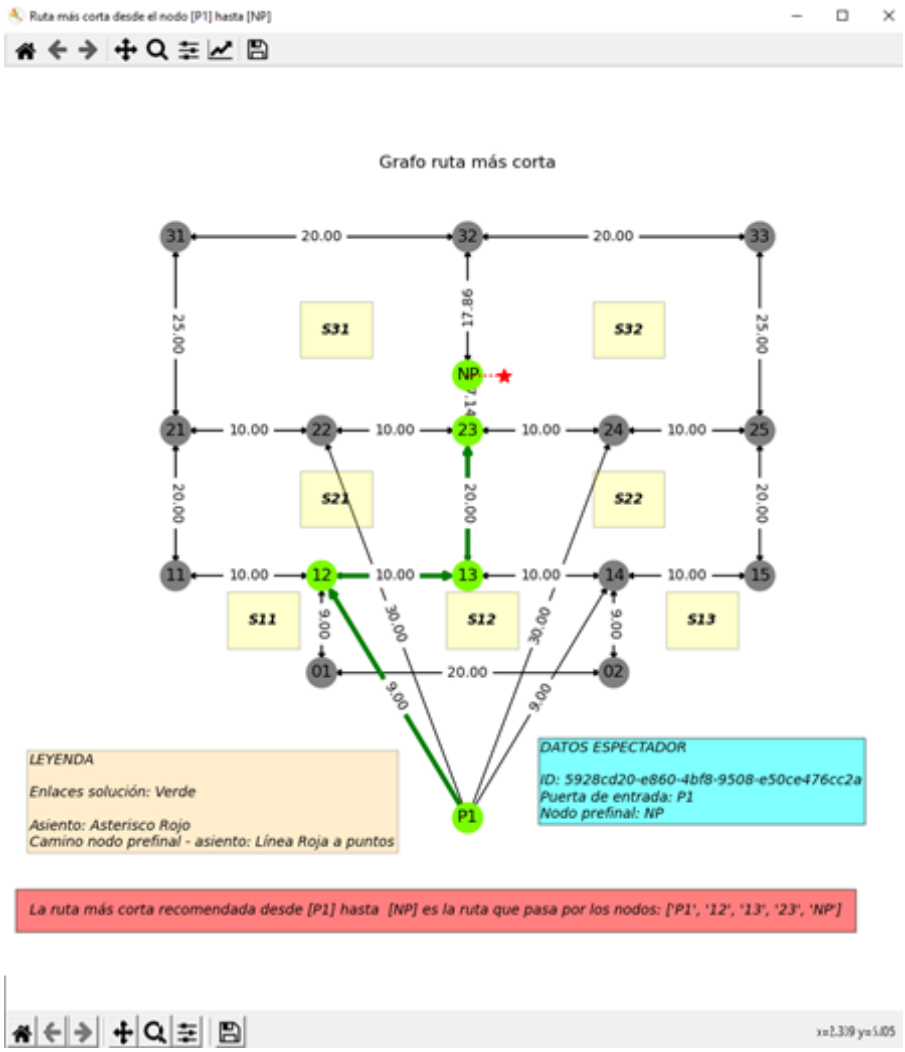

Ilustración 7.7. Salida ruta más corta.

7.7. RUTA MÁS RÁPIDA

La opción de ruta más rápida consigue la ruta que minimiza el peso de los enlaces, siendo este peso el tiempo requerido para recorrerlos, obtenido tal y como se describió en el apartado CÁLCULO DEL PESO DE LOS ENLACES.

Es importante destacar que, como ya se mencionó, un pasillo tiene dos direcciones y, por ejemplo, un enlace para la escalera que comunica los nodos A y B tiene diferente peso para la subida A-B y la bajada B-A.

La [Ilustración 7.8] nos muestra la salida que se recomendará al usuario, siendo la ruta final del espectador la formada por los nodos [P1, 12, 13, 23, NP].

Al realizar las ejecuciones para diferentes nodos se ha comprobado que las salidas para las rutas más rápidas son iguales que las correspondientes para las rutas más cortas. Por ello se ha considerado eliminar este algoritmo del proyecto, pero se ha llegado a la conclusión de que sería un error. Creemos que este algoritmo, tal y como se ha definido y para un modelo de datos como el tratado (un estadio con escaleras de igual elevación), no es de gran utilidad, pero sí es una importante base para futuras implementaciones de algoritmos que recomienden la ruta más rápida de forma realista. Es decir, este algoritmo se debería redefinir de forma que tenga en cuenta el número de personas que hay en cada enlace, y para ello se habrá de realizar un estudio de la fluidez de tránsito por los enlaces y la velocidad a la que cada usuario puede circular por un pasillo con *x* cantidad de usuarios ocupándolo. Además, se deberían efectuar análisis y pruebas, como se mencionó anteriormente, que obtuviesen las velocidades a las que las personas pueden caminar por pasillos planos y subir o bajar escaleras, en función de su edad, sexo, peso, altura, condiciones climáticas, etc. y también de la inclinación y el tipo de escaleras y pasillos planos presentes en el estadio.

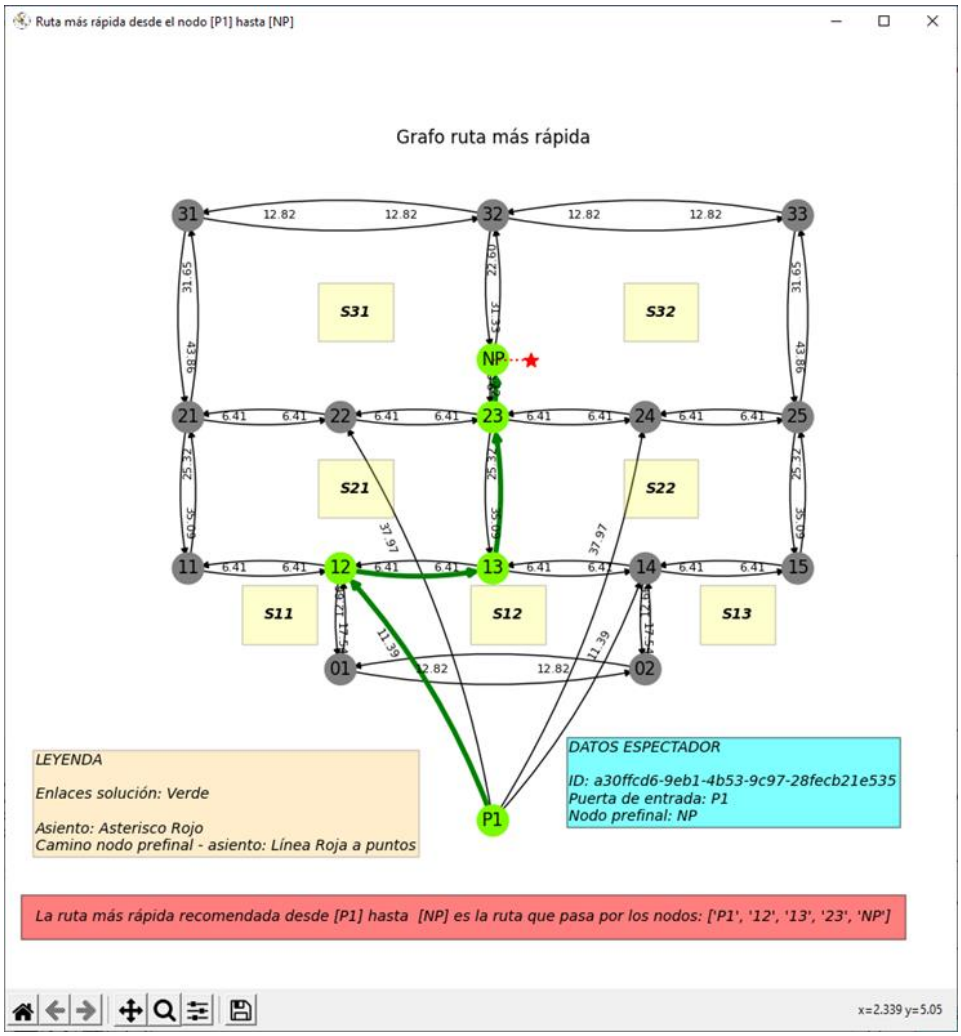

Ilustración 7.8. Salida ruta más rápida[26].

7.8. RUTA SEGÚN ATRIBUTOS

La opción de ruta según atributos está formada por dos partes. En primer lugar, se necesita obtener los requisitos que el usuario desea que la ruta tenga, tal y como se muestra en la

[26] En ilustración el peso de un enlace, en este caso el tiempo de recorrerlo, viene representado como la etiqueta numérica más cercana al nodo, es decir, el número más cercano a la punta de la flecha. Por ejemplo, en el enlace 21-31 el tiempo de subida será 31.65 unidades de tiempo, mientras que el tiempo de bajada será 43.86 unidades de tiempo.

[Ilustración 7.9], en la que se aprecia cómo el usuario puntúa de 0 a 5 cada uno de los atributos que los enlaces pueden tener.

Con estos valores el sistema obtiene los factores para cada atributo como se describe en el apartado RUTA SEGÚN REQUISITOS PERSONALES.

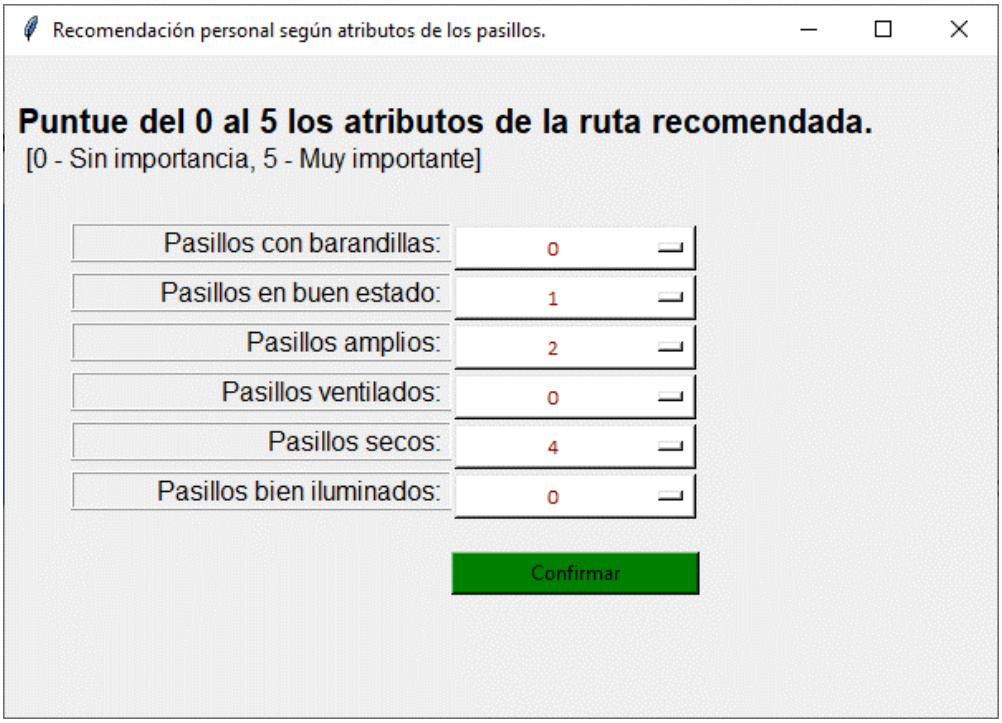

Ilustración 7.9. Panel de puntuaciones de atributos de los pasillos por parte del usuario.

En segundo lugar, una vez se han obtenido estos factores y se han aplicado para calcular los nuevos pesos de cada enlace, se aplica el algoritmo de Dijkstra y se obtiene la ruta que minimice el peso de los pasillos o, lo que es lo mismo, que maximice los enlaces que cumplen los requisitos del usuario.

Esta ruta se muestra en la [Ilustración 7.10], en la cual podemos apreciar cómo la ruta [P1, 24, 23, NP] es diferente a la obtenida en las dos opciones previas. Esto se debe principalmente a que el usuario valora muy positivamente rutas con pasillos secos y los enlaces 12-13 y 13-23 son pasillos que se mojarían en caso de lluvia, mientras que el pasillo P1-24 es más largo, pero no se moja y cumple los demás requisitos, por lo que tiene un peso menor.

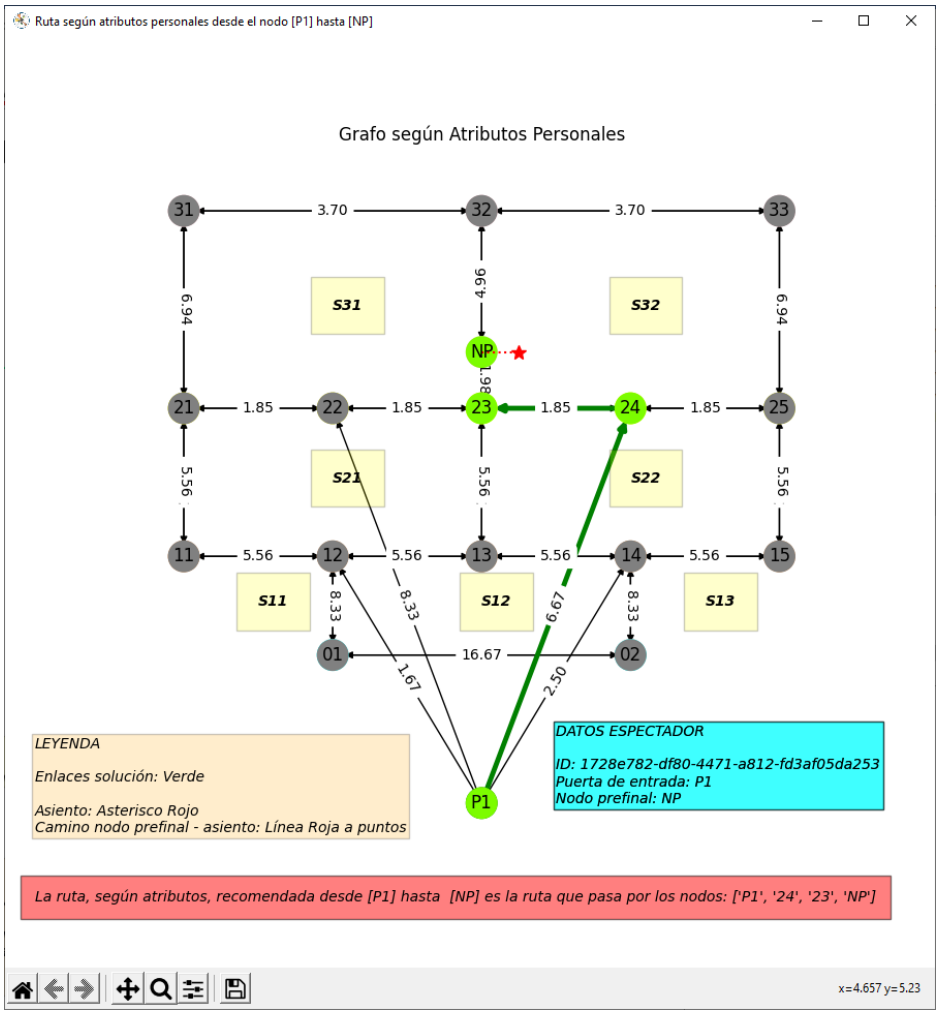

Ilustración 7.10. Salida ruta según atributos personales.

7.9. RUTA CON CONTROL DE AGLOMERACIONES (SIMULACIÓN)

La opción de ruta con control de aglomeraciones no genera una salida de forma directa, sino que se realiza una simulación de un espectador caminando por el estadio y parándose en cada nodo. A su vez, se modifica la ocupación del estadio creando así modelos dinámicos del número de personas que se encuentran transitando por los diferentes pasillos.

Es por esto por lo que un espectador que acceda a un mismo asiento en dos ejecuciones distintas podrá recibir dos rutas recomendadas diferentes, ya que estas dependen de cómo esté la ocupación y esto es algo totalmente aleatorio en la simulación.

Esta simulación es equivalente a un supuesto donde un usuario accediera al estadio con su teléfono móvil y solicitara al sistema (que podría ser una aplicación móvil) que le diera una indicación de por donde llegar a su asiento teniendo en cuenta las personas en cada enlace. En ese caso, cada vez que el espectador llegase a un nodo el sistema recalcularía la ruta en dicho instante con los nuevos valores de ocupación y le recomendaría un nuevo enlace a tomar hasta el siguiente nodo.

Para mostrar cómo se pueden producir salidas distintas vamos a ejecutar esta opción varias veces para el mismo asiento y espectador y compararemos las rutas obtenidas en cada una de ellas.

7.9.1. PRIMERA EJECUCIÓN

Vamos a dividir cada nodo en el que se para el espectador como un paso. De este modo cada ejecución tendrá tantos pasos como vértices tenga la ruta recomendada.

7.9.1.1. PRIMER PASO

Al empezar la ejecución el sistema simula el llenado del estadio con espectadores. En un momento se detiene el rellenado y se sitúa al usuario en la puerta de entrada P1, tal y como se aprecia en la [Ilustración 7.11]. En dicho instante se analiza el grafo y se obtiene la ruta hasta el nodo prefinal o NP; de esta ruta se extrae el nodo siguiente al nodo inicial, en este caso el nodo 22. Esto se debe a que si nos fijamos en el grafo de la [Ilustración 7.12] podemos ver cómo la ruta [P1, 22, 23, NP] es la ruta con menor ocupación.

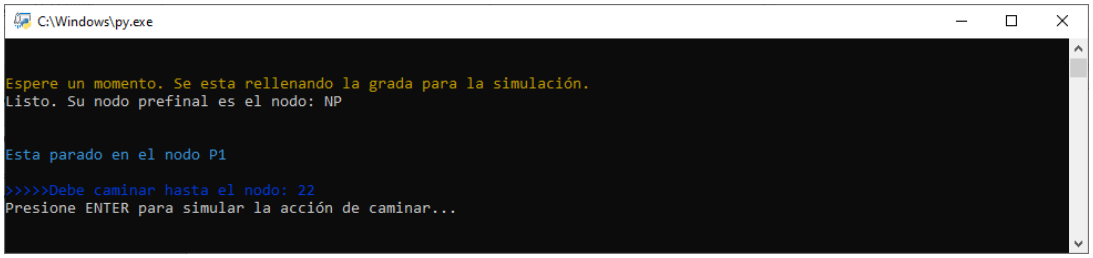

Ilustración 7.11. Salida por consola, primer paso - primera ejecución.

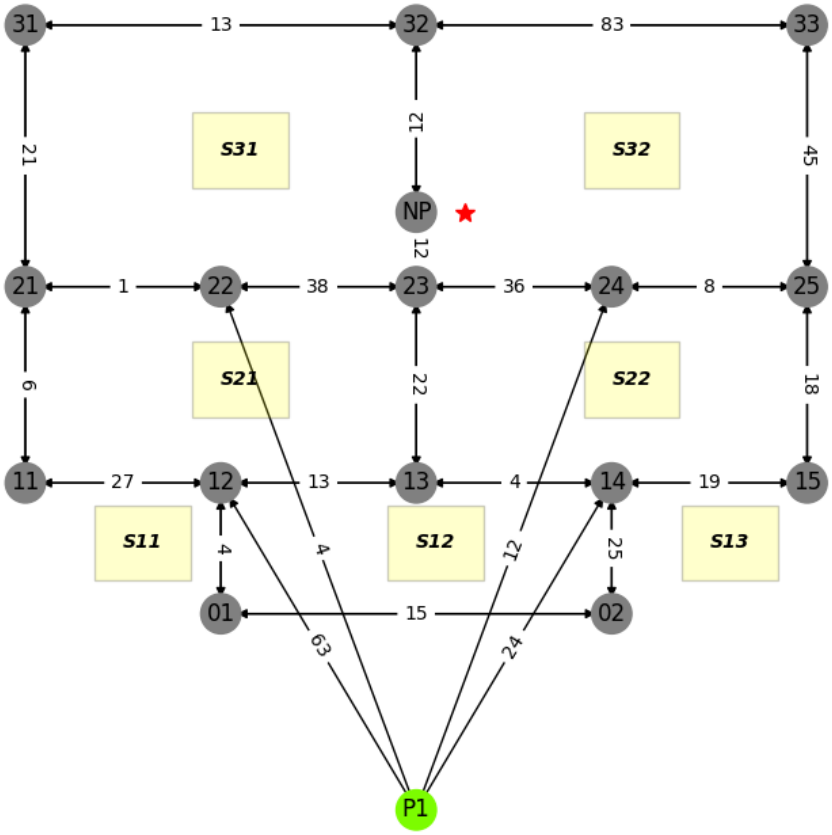

Ilustración 7.12. Grafo, primer paso - primera ejecución.

7.9.1.2. SEGUNDO PASO

En el momento en el que se pulse *"ENTER"* se simulará cómo el usuario camina al nodo 22 y mientras tanto la ocupación del estadio continuará variando.

En el momento en que el usuario llegue a ese nodo se recalculará la ruta y se le indicará el nodo siguiente al que debe ir, de forma recursiva como se hizo en el primer paso.

La [Ilustración 7.13 e Ilustración 7.14] muestra la salida por consola de la simulación y el estado del grafo en el momento de calcular el siguiente nodo al que debe caminar el espectador. Además, podemos ver cómo el enlace que ya ha sido recorrido se borra con el objetivo de evitar ciclos infinitos.

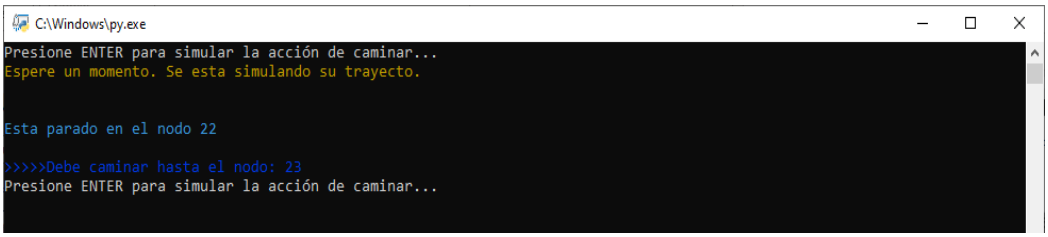

Ilustración 7.13. Salida por consola, segundo paso - primera ejecución.

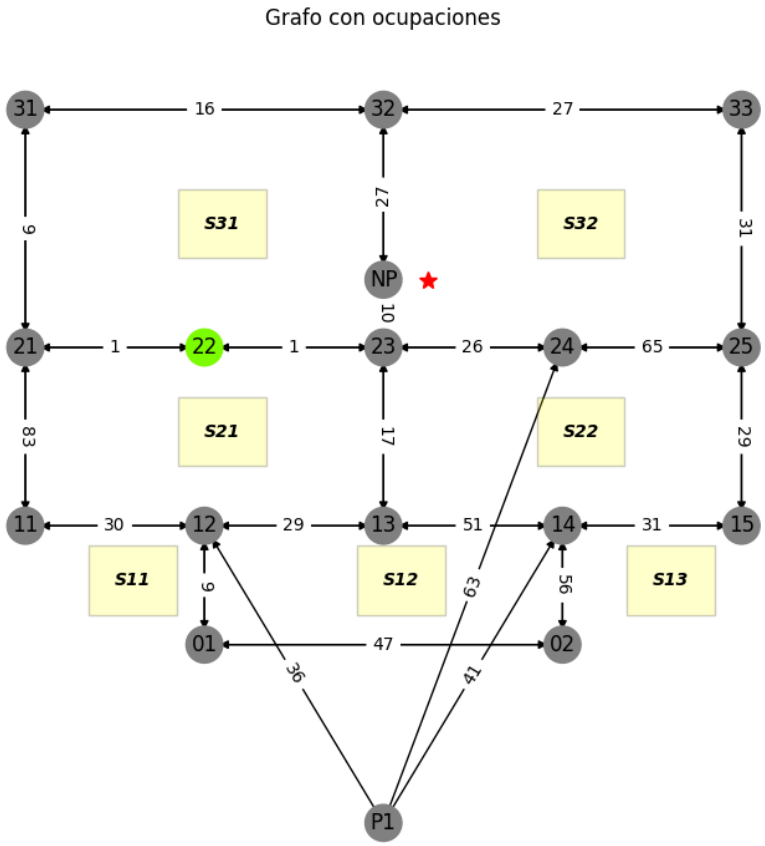

Ilustración 7.14. Grafo, segundo paso - primera ejecución.

7.9.1.3. TERCER PASO

Este tercer paso, al igual que los dos anteriores obtiene el siguiente nodo al que caminar, en este caso es el nodo NP, por lo que se sabe que quedará únicamente un paso más.

La [Ilustración 7.15 e Ilustración 7.16] muestra la consola y el estado del grafo en el momento en el que el espectador está en el nodo 23.

Ilustración 7.15. Salida por consola, tercer paso - primera ejecución.

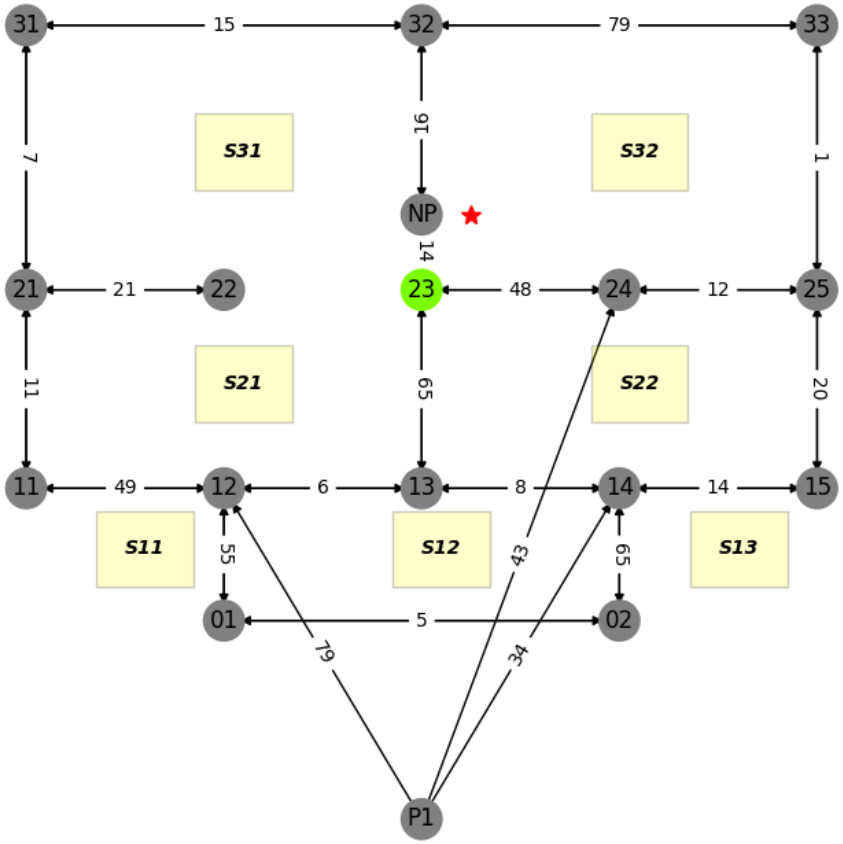

Ilustración 7.16. Grafo, tercer paso - primera ejecución.

7.9.1.4. CUARTO PASO

Este paso corresponde al último paso de la simulación, en él el espectador llega al nodo prefinal o NP y solamente le quedará caminar hasta el asiento.

La [Ilustración 7.17] muestra la salida por consola en la que se puede observar que se llegó al nodo prefinal y se indica, a modo de depuración, la ruta que el espectador ha seguido hasta llegar a su destino; en este caso [P1, 22, 23, NP].

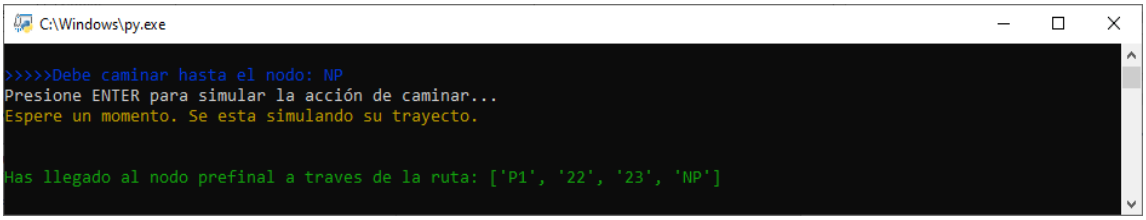

Ilustración 7.17. Salida por consola, cuarto paso - primera ejecución.

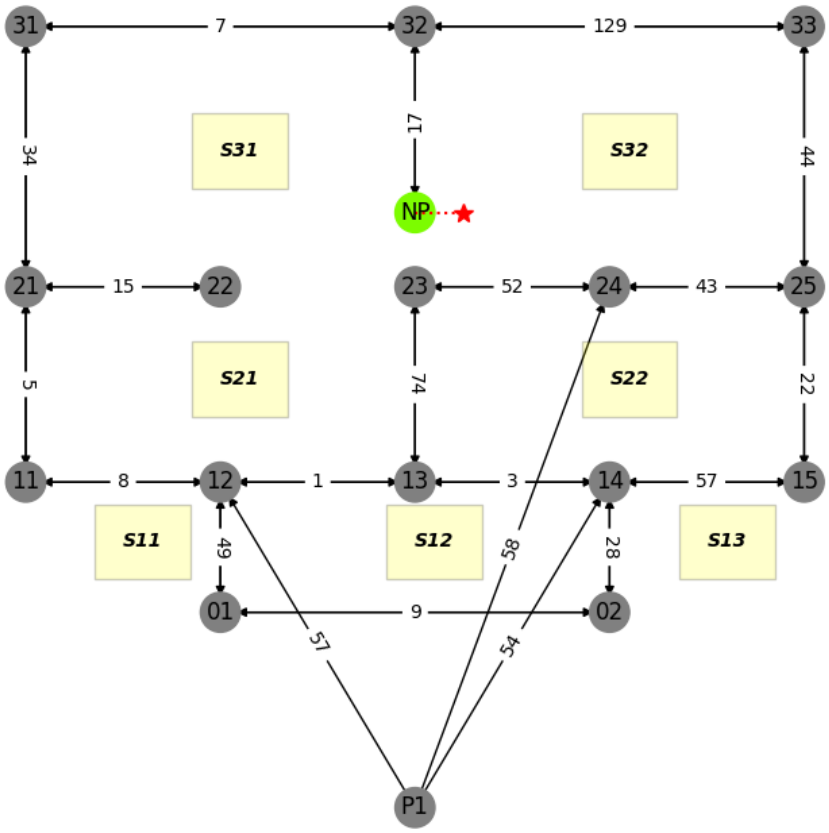

Ilustración 7.18. Grafo, cuarto paso - primera ejecución.

7.9.2. SEGUNDA EJECUCIÓN

Al inicio de esta segunda ejecución el estado se vacía de personas, por lo que se parte otra vez de un grafo sin ningún espectador. Posteriormente se realizan los mismos pasos que en la primera ejecución.

A continuación se mostrarán los resultados obtenidos en cada paso.

7.9.2.1. PRIMER PASO

Como nos muestra la [Ilustración 7.19], el nodo siguiente al que el espectador debe ir es el nodo 14. Esto se debe a que si observamos el grafo de la [Ilustración 7.20] vemos que el camino de la ejecución anterior [P1, 22, 23, NP] tiene una ocupación ahora de 70 personas mientras que el recorrido [P1, 14, 13, 23, NP] tiene una ocupación de 54 personas.

También se podría considerar ir por la ruta [P1, 14, 15, 25, 24, 23, NP] con una ocupación de 55 personas, menor que la ocupación de la ruta de la ejecución anterior pero no menor que la ruta [P1, 14, 13, 23, NP]. En ambos casos el nodo siguiente es el 14, por lo que cuando el espectador llegue a ese punto se decidirá el nodo a seguir.

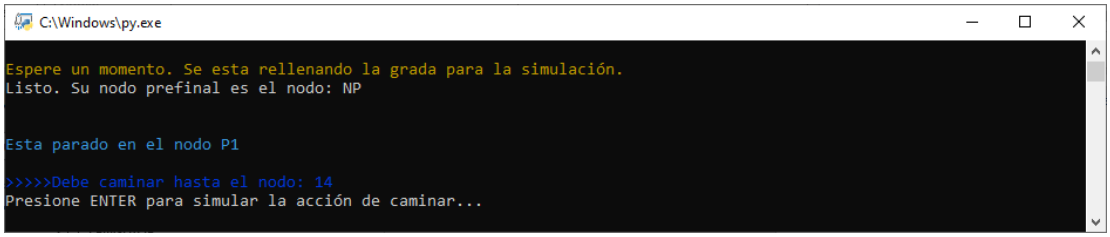

Ilustración 7.19. Salida por consola, primer paso - segunda ejecución.

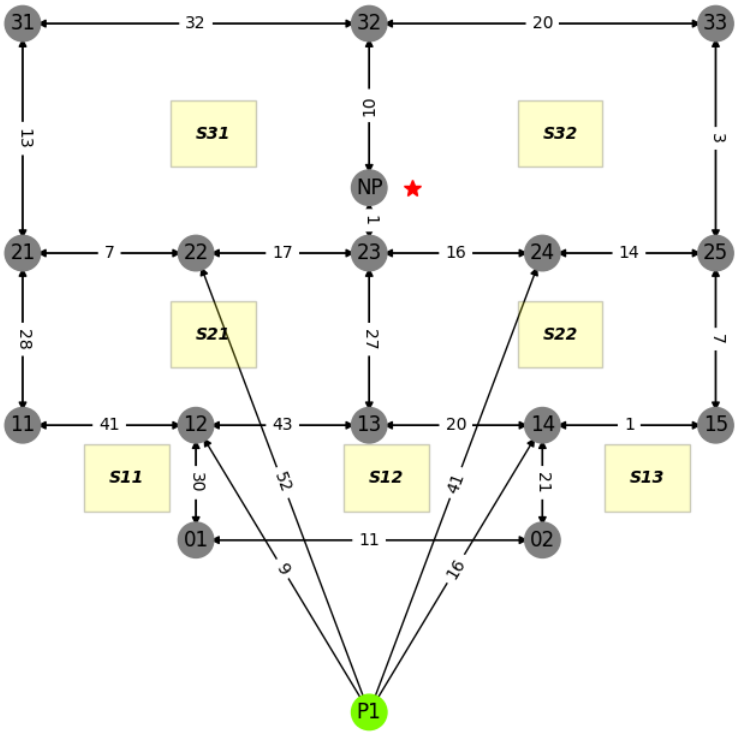

Ilustración 7.20. Grafo, primer paso - segunda ejecución.

7.9.2.2. SEGUNDO PASO

Tal y como vemos en la [Ilustración 7.21], el nodo siguiente al que debe caminar el espectador es el 15. Por lo tanto, la idea inicial de que el recorrido sería el que transcurriría por los nodos [P1, 14, 13, 23, NP] era errónea.

Además, al igual que ocurrió en el paso anterior, no sabemos la ruta definitiva observando la [Ilustración 7.22], ya que podríamos considerar ir por la ruta [14, 15, 25, 24, 23, NP] o por la ruta [14, 15, 25, 33, 32, NP][27]. Por ello no se decidirá qué ruta tomar hasta llegar a la bifurcación del nodo 25.

Ilustración 7.21. Salida por consola, segundo paso - segunda ejecución.

[27] Aunque existen más rutas, en este ejemplo solo se comentan esas dos.

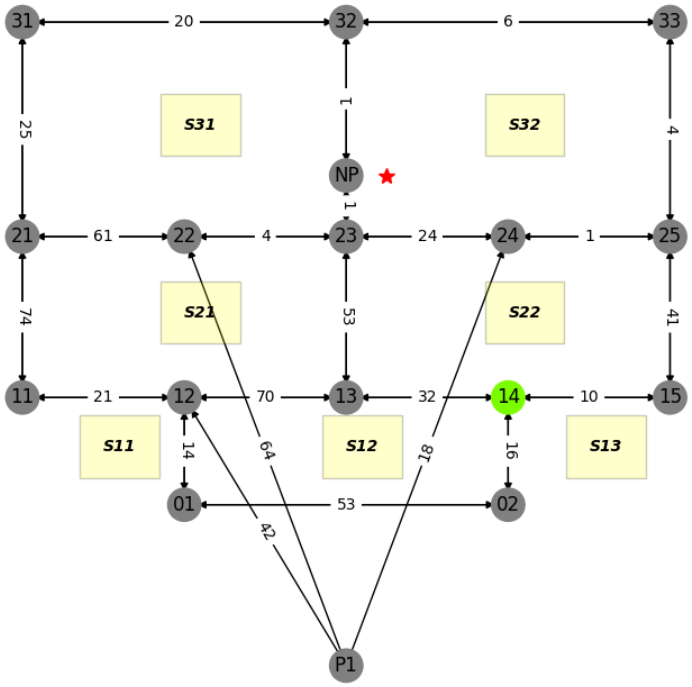

Ilustración 7.22. Grafo, segundo paso - segunda ejecución.

7.9.2.3. TERCER PASO

En este paso no podemos sacar ninguna conclusión y el usuario seguirá hasta el único nodo accesible, el 25. Esto se debe a que el algoritmo ha ido borrando los enlaces ya recorridos y de esta forma se evita recorrer un pasillo y posteriormente deshacerlo en sentido contrario.

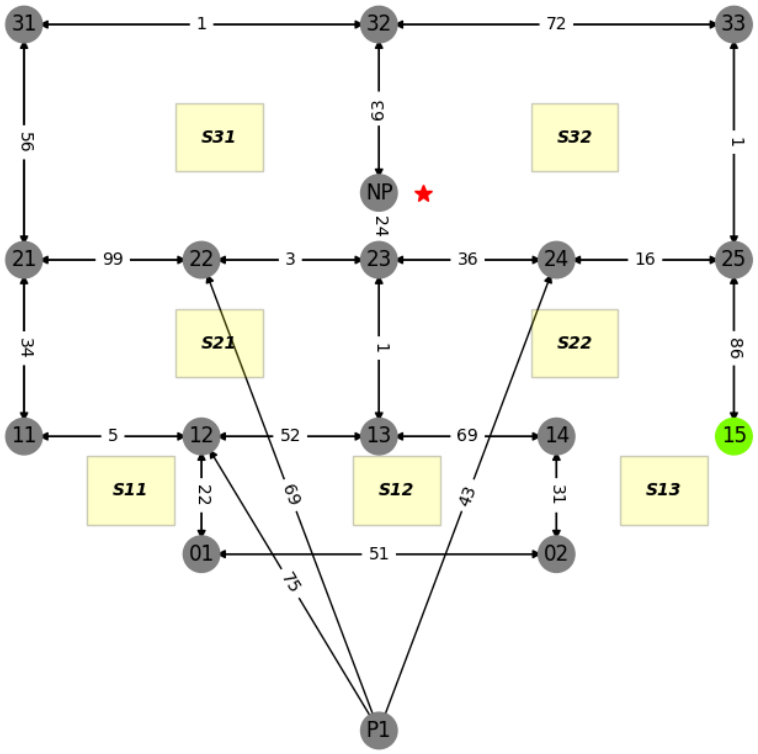

Ilustración 7.23. Grafo, tercer paso - segunda ejecución.

7.9.2.4. CUARTO PASO

Como vemos en la [Ilustración 7.25] desde el nodo 25 el usuario tiene varias rutas a tomar. En primer lugar, la formada por los nodos [25, 33, 32, NP] con una ocupación de 150 personas y, en segundo lugar, la ruta [25, 24, 23, NP] con una ocupación de 176 personas[28].

Es por ello por lo que la ruta que el espectador deberá tomar es la que le lleva al nodo 33, algo que el programa confirma en la [Ilustración 7.24].

Ilustración 7.24. Salida por consola, cuarto paso - segunda ejecución.

[28] Aunque existen más rutas, en este ejemplo solo se comentan esas dos.

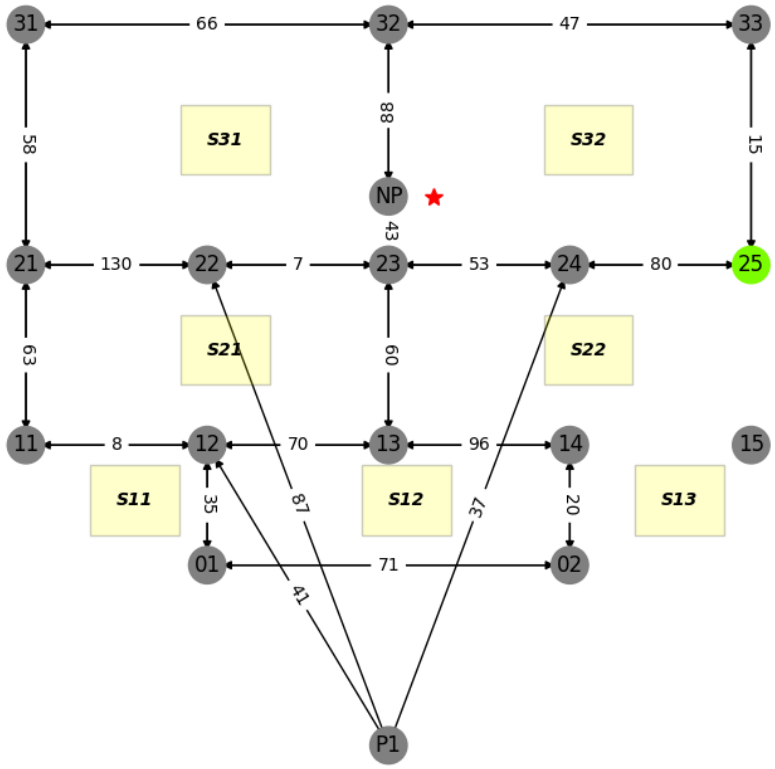

Ilustración 7.25. Grafo, cuarto paso - segunda ejecución.

7.9.2.5. QUINTO PASO

Al igual que ha ocurrido en el tercer paso, estamos en un nodo en el cual el usuario solamente puede caminar hasta la siguiente bifurcación (nodo 32).

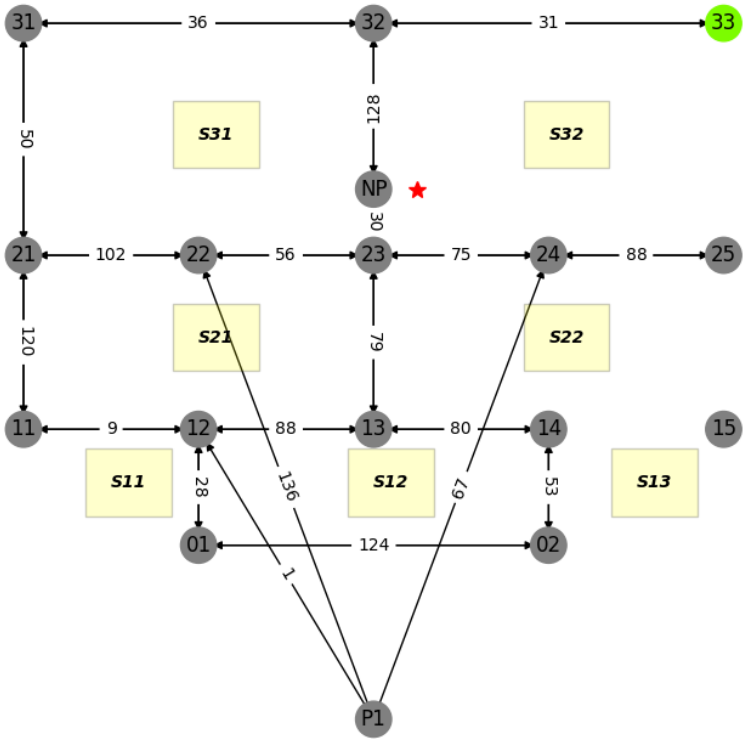

Ilustración 7.26. Grafo, quinto paso - segunda ejecución.

7.9.2.6. SEXTO PASO

En este sexto paso el espectador se encuentra en el nodo 32 y su nodo siguiente es su nodo destino o nodo prefinal, ya que entre ambos hay una ocupación de 66[29] personas, como se observa en la [Ilustración 7.28], mientras que el camino [32, 31, 21, 22, 23, NP] tiene 244 personas.

Por lo tanto, al espectador solamente le queda un paso por recorrer hasta llegar al nodo previo a su asiento.

[29] El enlace 32-NP tiene una ocupación de 66 personas y no de 99, ya que el número está orientado al contrario del resto de los enlaces. Esto se debe a un problema al añadir enlaces de la librera NetworkX y Matplotlib.

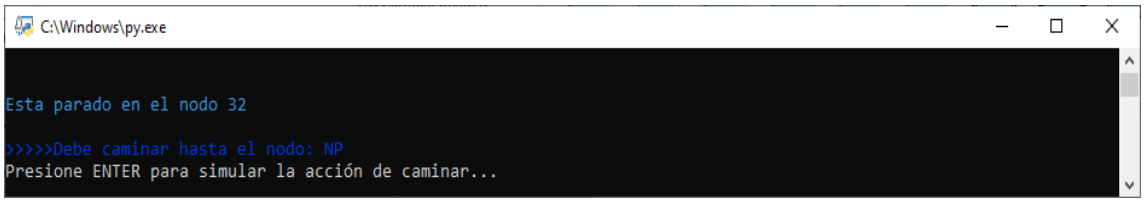

Ilustración 7.27. Salida por consola, sexto paso - segunda ejecución.

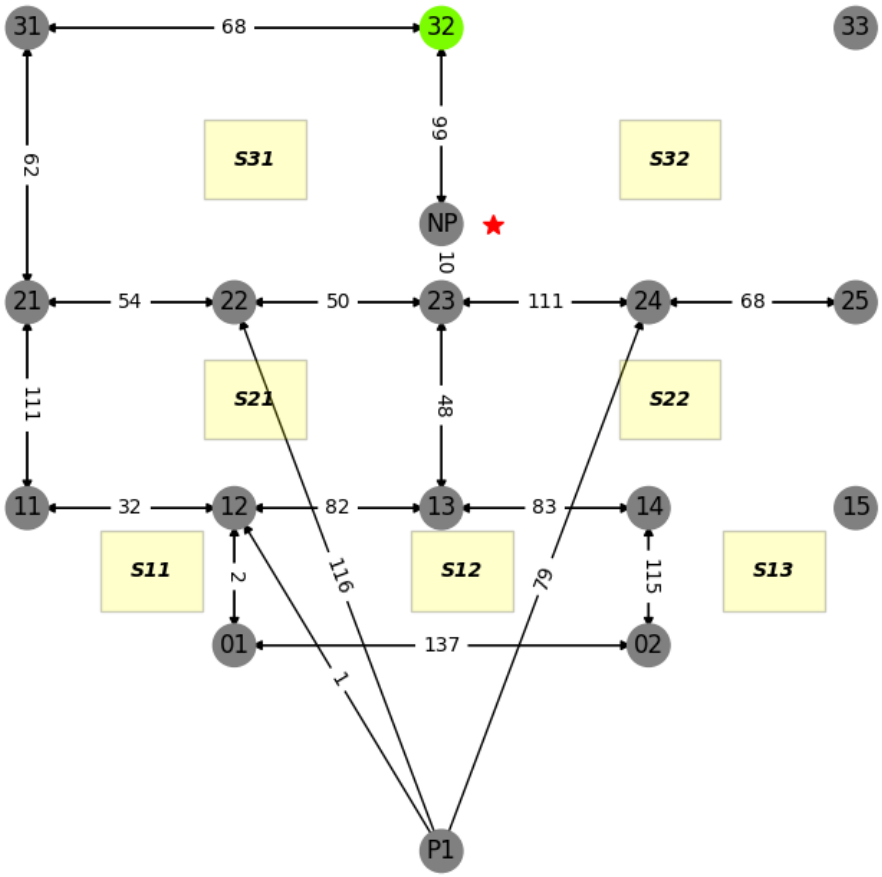

Ilustración 7.28. Grafo, sexto paso - segunda ejecución.

7.9.2.7. SÉPTIMO PASO

Como se ha indicado antes, este es el último paso ya que el espectador ha llegado al nodo prefinal o NP y solamente debe caminar por la fila de asientos hasta su butaca, tal y como se observa en la [Ilustración 7.30].

En esta segunda ejecución el espectador ha llegado a su destino por los nodos [P1, 14, 15, 25, 33, 32, NP], como refleja la [Ilustración 7.29].

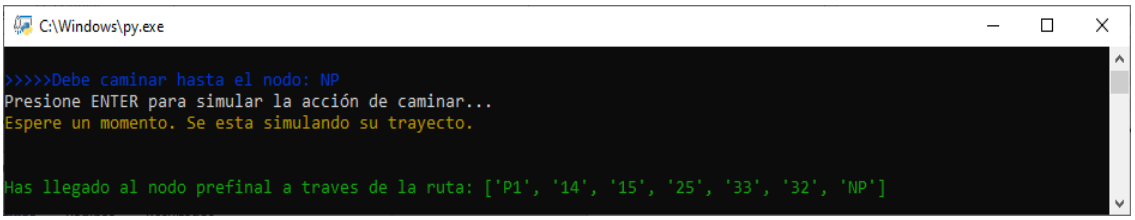

Ilustración 7.29. Salida por consola, séptimo paso - segunda ejecución.

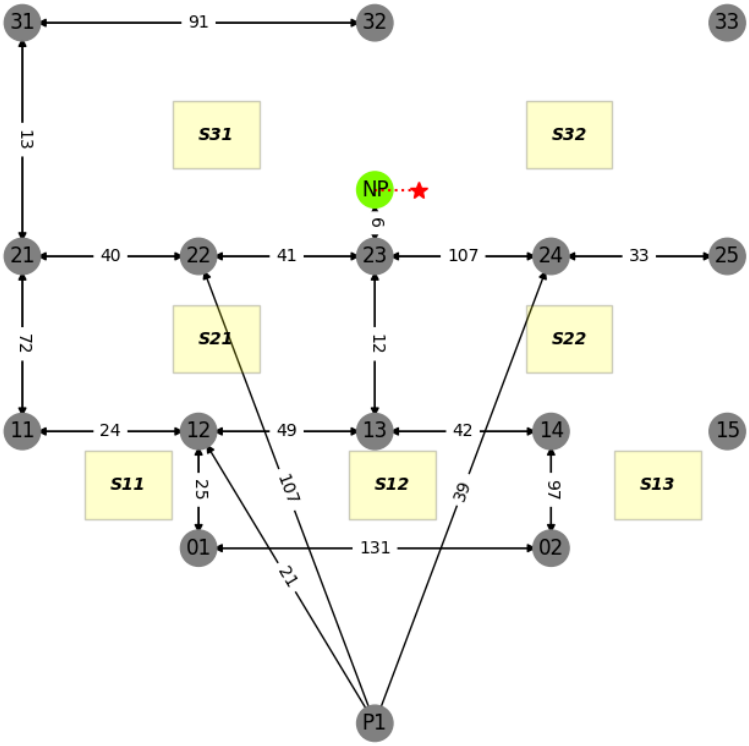

Ilustración 7.30. Grafo, séptimo paso - segunda ejecución.

7.9.3. COMPARATIVA ENTRE LAS DOS EJECUCIONES

En la primera simulación se ha llegado al nodo prefinal a través de la ruta [P1, 22, 23, NP] con una distancia recorrida de $30 + 10 + 7.14 = 47.14\ unidades\ de\ longitud$.

En el caso de la segunda simulación se obtuvo la ruta [P1, 14, 15, 25, 33, 32, NP], la cual tiene una distancia recorrida de $9 + 10 + 20 + 25 + 20 + 17.86 = 101.86\ unidades\ de\ longitud$. Es decir, el espectador que accede en el

segundo momento o segunda ejecución debe recorrer una distancia de $54.72\ unidades\ de\ longitud$ más que en el primer momento o primera ejecución.

Se podría considerar entonces que el sistema no está bien si solo nos fijamos en la distancia que el usuario ha recorrido en ambas ejecuciones, pero no es así puesto que para cada instancia del estadio la ocupación es diferente. Si nos imaginamos un estadio real, las personas se mueven constantemente cuando acceden a sus asientos y, por lo tanto, una recomendación en un momento de tiempo *x* es distinta a una realizada en un momento *x+1*. Este tema y las posibles mejoras de estos algoritmos se abordará en el apartado de FUTURAS MEJORAS.

8. CONCLUSIONES Y TRABAJO FUTURO

8.1. CONCLUSIONES

En este proyecto se han realizado los primeros pasos para la construcción de un sistema de recomendaciones destinado a la obtención de rutas para los espectadores que acceden a un estadio de fútbol y que tienen como objetivo llegar a su asiento asignado. Para ello se ha realizado un estudio de la situación actual de los accesos a los eventos deportivos de fútbol de la Liga Española en sus dos categorías profesionales, Primera y Segunda división. Estos datos han permitido analizar la viabilidad del sistema y su implementación en entornos reales en la actualidad.

Se han estudiado las diferentes técnicas y recursos utilizados por los sistemas de recomendaciones existentes, eligiendo con ello la mejor solución para este proyecto, y especificando y definiendo los algoritmos que se han implementado posteriormente.

Se ha generado un modelo estructural en 3D de prueba con la finalidad de simular las gradas, los pasillos y los asientos de un estadio real. Este modelo se ha transformado, siguiendo las técnicas documentadas sobre la teoría de grafos, para generar un grafo con los nodos y los enlaces de la grada modelada, permitiendo con ello realizar pruebas y aplicar los diferentes algoritmos desarrollados sobre este.

Siguiendo las pautas y los diferentes algoritmos definidos en este proyecto, se ha generado y documentado un código capaz de generar recomendaciones personalizadas para el espectador partiendo de los datos obtenidos de la entrada de este y el grafo de la grada del estadio creado.

Los resultados obtenidos hasta el momento permiten ver un prototipo de sistema capaz de generar rutas personalizadas para cada espectador. Aunque es cierto que estas salidas se han obtenido para un modelo de datos muy reducido, se puede apreciar una base bien definida de cara a una implementación de mayor escala. Por ello será necesario continuar desarrollando este sistema, aplicando mejoras y correcciones sobre determinados algoritmos con el objetivo de obtener un producto final enfocado a entornos y estadios de fútbol reales.

Además, a medida que se ha ido desarrollando el proyecto se ha percibido que este tipo de sistemas de recomendaciones no es exclusivo para eventos en estadios de fútbol, puesto que

la mayoría de los estadios y estructuras con gradas de toda índole presentan unas características similares. Por ello se podrían crear sistemas similares enfocados a las diferentes construcciones arquitectónicas para eventos con gradas y espectadores, tales como circuitos automovilísticos, hipódromos, estadios de baloncesto, pistas de tenis, etc.

8.2. FUTURAS MEJORAS

Como se ha comentado anteriormente este sistema inicial de recomendaciones permite implementar gran cantidad de mejoras, por lo que se van a citar a continuación las más relevantes.

8.2.1. EVALUACIÓN DE LOS DIFERENTES ALGORITMOS DE RECOMENDACIONES

Como se ha tratado en el apartado EVALUACIÓN DE LOS SISTEMAS DE RECOMENDACIONES, los sistemas de recomendaciones deben ser evaluados antes y durante su implementación en entornos reales. Es por ello que previamente a la obtención de un sistema final se deben realizar evaluaciones de las recomendaciones producidas por este, por ejemplo, obteniendo puntuaciones por parte de los espectadores sobre la ruta que se le ha ofrecido, para posteriormente analizar la afinidad del usuario y el recorrido recomendado.

8.2.2. MODELO DE UN ESTADIO REAL

Dada la situación que el mundo vive actualmente ha resultado imposible realizar estudios de campo sobre alguno de los estadios de fútbol de España. Por ello una posible mejora del sistema sería realizar un modelo de un estadio real, tomando para ello mediciones de la longitud y la anchura de los diferentes pasillos, obteniendo el número exacto de filas y columnas de todos los sectores de las gradas y también rellenando de forma más realista los atributos o características de cada pasillo.

8.2.3. ALGORITMO DE RUTA MÁS RÁPIDA

Como se ha mencionado en la definición del algoritmo, se deberá realizar un estudio más detallado de la velocidad y flujo de tránsito por los diferentes pasillos en función de los diferentes tipos de espectadores. Esto permitirá crear un sistema con variables de velocidad más realistas y ajustadas a las propiedades de cada espectador.

8.2.4. ALGORITMO DE RUTA SEGÚN ATRIBUTOS PERSONALES

Será necesario introducir más atributos al programa para que los espectadores puedan puntuar o elegir sus preferencias. Para ello se deberán realizar y tratar encuestas de personas que accedan a los estados en la actualidad y de esta forma obtener las características más importantes que ellos consideran que las rutas deben tener a la hora de elegir un camino u otro.

8.2.5. ALGORITMO DE RUTA CON CONTROL DE AGLOMERACIONES

Uno de los puntos débiles del algoritmo de rutas con control de aglomeraciones es que no se está teniendo en cuenta la distancia recorrida para llegar al asiento y únicamente se está controlando que los siguientes pasillos minimizan la cantidad de gente que está circulando por ellos. Por esto una posible mejora sería controlar esta distancia recorrida y establecer unos criterios para recomendar un trayecto u otro. Es decir, definir si es mejor recorrer un trayecto de 100 metros con 10 personas u otro de 20 metros con 40 personas.

Otro punto por mejorar de este algoritmo es incluir más variables en las rutas, como, por ejemplo, la anchura del pasillo. De este modo, al igual que en el caso anterior, se deberá valorar si es mejor un pasillo estrecho, lo que equivale a que la gente estará más próxima, u otro de mayor anchura pero con mayor ocupación.

8.2.6. CREACIÓN DE UNA APLICACIÓN MÓVIL

Con el objetivo de acercar este proyecto a un entorno real, será necesario crear una aplicación móvil a partir de lo descrito en este documento. Esto nos permitirá mejorar el sistema de recomendaciones, por ejemplo, mediante el uso de la antena GPS para el posicionamiento geográfico en el estadio o con la antena Bluetooth para comunicar varios dispositivos y transmitir información entre ellos.

Es por ello que uno de los siguientes pasos que debe seguir este proyecto está enfocado a desarrollar una aplicación móvil que permita diseñar e implementar funcionalidades extras a las descritas en el trabajo.

9. BIBLIOGRAFÍA

[1] Condiciones de compra campaña de abonados 2019 [Internet]. [consultado el 10 de octubre de 2020]. Conservado en: pdf, Gijón. Disponible en: https://files.proyectoclubes.com/sporting/201907/04140800condiciones-de-compra__abonados--1-.pdf

[2] LaLiga 2018-2019 Memoria Anual [Internet]. Actividades deportivas - Laliga 2017-2018 memoria anual; [consultado el 12 de octubre de 2020]. Disponible en: https://memorias.laliga.com/2017/es/actividades-deportivas#asistencia-de-espectadores

[3] Mahmood T, Ricci F. Improving Recommender Systems with Adaptive Conversational Strategies. Association for Computing Machinery [Internet]. 2009 [consultado el 20 de mayo de 2021]:73-82. Disponible en: https://doi.org/10.1145/1557914.1557930

[4] Resnick P, Varian HR. Recommender systems. Communications of the ACM [Internet]. Marzo de 1997 [consultado el 15 de enero de 2021]; 40(3):56-58. Disponible en: https://doi.org/10.1145/245108.245121

[5] Burke R. User Modeling and User-Adapted Interaction [Internet]. 2002 [consultado el 25 de mayo de 2021];12(4):331-70. Disponible en: https://doi.org/10.1023/a:1021240730564

[6] Burke R. Hybrid Web Recommender Systems. [Internet]. 2007 [consultado el 8 de enero de 2021]; Lecture Notes in Computer Science, vol 4321. Disponible en: https://doi.org/10.1007/978-3-540-72079-9_12

[7] Shafer J.B., Frankowski D, Herlocker J, Sen S. Collaborative Filtering Recommender Systems. The Adaptive Web. [Internet]. 2007 [consultado el 25 de enero de 2021];4321. Disponible en: https://doi.org/10.1007/978-3-540-72079-9_9

[8] Google Inc. [Internet]. Content-based Filtering | Recommendation Systems | Google Developers; [consultado el 2 de febrero de 2021]. Disponible en: https://developers.google.com/machine-learning/recommendation/content-based/basics

[9] Tran T, Cohen R. Hybrid Recommender Systems for Electronic Commerce. Computer Science [Internet]. 2000 [consultado el 25 de febrero de 2021];18(4):78-83. Disponible en: https://www.aaai.org/Papers/Workshops/2000/WS-00-04/WS00-04-012.pdf

[10] Rocca B. Towards Data Science [Internet]. Introduction to recommender systems; 2 de junio de 2019 [consultado el 5 de mayo de 2021]. Disponible en: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada

[11] Software de diseño 3D | Modelado 3D en la web | SketchUp [Internet]; [consultado el 1 de marzo de 2021]. Disponible en: https://www.sketchup.com/es

[12] Jasika N, Alispahic N, Elma A, Ilvana K, Elma L, Nosovic N. Dijkstra's shortest path algorithm serial and parallel execution performance analysis. En: 2012 Proceedings of the 35th International Convention MIPRO [Internet]; 16 de julio de 2012; Opatija; [consultado el 9 de marzo de 2021]. p. 1811-5. Disponible en: https://ieeexplore.ieee.org/document/6240942

[13] Floyd RW. Algorithm 97: Shortest path. Communications of the ACM [Internet]. Junio de 1962 [consultado el 25 de febrero de 2021];5(6):345. Disponible en: https://doi.org/10.1145/367766.368168

[14] Choi JH, Galea ER, Hong WH. Individual Stair Ascent and Descent Walk Speeds Measured in a Korean High-Rise Building. Fire Technology [Internet]. 28 de diciembre de 2013 [consultado el 25 de mayo de 2021];50(2):267-295. Disponible en: https://doi.org/10.1007/s10694-013-0371-4

[15] WHO | World Health Organization [Internet]. Información basíca sobre la COVID-19; [consultado el 6 de abril de 2021]. Disponible en: https://www.who.int/es/news-room/q-a-detail/coronavirus-disease-covid-19

[16] Barth D. Official Google Blog [Internet]. The bright side of sitting in traffic: Crowdsourcing road congestion data; 25 de agosto de 2009 [consultado el 23 de enero de 2021]. Disponible en: https://googleblog.blogspot.com/2009/08/bright-side-of-sitting-in-traffic.html

[17] Official Google Blog [Internet]. The bright side of sitting in traffic: Crowdsourcing road congestion data; [consultado el 20 de abril de 2021]. Disponible en: https://googleblog.blogspot.com/2009/08/bright-side-of-sitting-in-traffic.html

10. PLANIFICACIÓN TEMPORAL

A continuación se muestra el diagrama de Gantt de la planificación temporal seguida durante el desarrollo de este proyecto [Figura 10.1], el cual ha tenido una duración de 31 semanas. La duración individual de cada tarea se muestra en el centro de las casillas representado en semanas.

	OCTUBRE				NOVIEMBRE				DICIEMBRE				ENERO				FEBRERO				MARZO				ABRIL				MAYO			
TAREAS	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4
Definición y especificación de la idea inicial	4																															
Investigación sobre sistemas similares					2																											
Búsqueda bibliográfica							2																									
Investigación de las tecnologías existentes									3																							
Documentar los algoritmos a desarrollar													4																			
Detallar el modelo de datos														3																		
Construcción de un modelo de datos																	1															
Revisión de los algoritmos en base al modelo																		2														
Programación del sistema prototipo																					3											
Validación de las salidas obtenidas																						2										
Ajuste y corrección de errores																									2							
Documentación de los resultados obtenidos																										2						
Conclusiones y trabajos futuros																												1				
Completar la memoria																													3			

Figura 10.1. Diagrama de Gantt de la planificación temporal del proyecto.

11. APÉNDICES

11.1. DOCUMENTACIÓN DEL CÓDIGO

TFG - Sistema de recomendaciones para accesos a estadios de fútbol

Versión 1.0.0

Pelayo Tiesta Cosío

27 de mayo de 2021

Capítulos

CAPÍTULO 1

Introducción

1.1 Descripción

Este Sistema, desarrollado en Python 3, ha sido creado como un código solución complementario al TFG - Sistemas de recomendaciones para accesos a estadios de fútbol. Dicho TFG trata de definir y crear un sistema de recomendaciones para los usuarios que desean acceder a un estadio de futbol, de forma que se brinden a estos diferentes soluciones de rutas en función de las necesidades de cada espectador.

Este sistema ofrece 4 salidas o rutas distintas al usuario:

- Ruta más corta desde la puerta de entrada hasta el asiento.
- Ruta más rápida desde la puerta de entrada hasta el asiento.
- Ruta más corta y que maximice las necesidades del usuario.
- Ruta con simulación de control de aglomeraciones.

1.2 Instalación

El código no requiere de instalación, pero sí de una serie de pasos de configuraciones para poder ejecutarlo.

1.2.1 Instalar Python3:

Python debe estar instalado en el equipo para poder ejecutar el programa.

Se puede descargar desde la Página oficial de Python

1.2.2 Instalar librerías:

Se deben instalar la siguientes librarías para poder ejecutar el programa:

Matplotlib

```
$ pip install matplotlib
```

Networkx

```
$ pip install networkx
```

Colorama

```
$ pip install colorama
```

1.2.3 Distribución de ficheros y carpetas:

Los ficheros deben estar colocados de la siguiente forma:

```
CODIGO
├── Main.py
├── Funciones.py
├── Clases.py
└── Ficheros
        ├── Sectores.csv
        ├── Nodos.json
```

1.3 Ejecución

Para ejecutar el programa se debe lanzar el programa Main.py

En Windows:

Doble click en el fichero Main.py

Desde la CMD

```
$ python Main.py
```

```
$ python3 Main.py
```

En Linux:

```
$ python Main.py
```

```
$ python3 Main.py
```

1.4 Uso del programa

Al ejecutar el programa Main.py:

1. Se muestra el grafo del estadio al usuario.
2. El usuario debe introducir los datos de su asiento:
   - Sector - Con un formato SXX (Donde XX es el número que identifica al sector)
   - Fila
   - Columna
   - Puerta de entrada

Nota: Si el sector o la columna está fuera del rango de filas o columnas del sector se producirá una excepción.

Nota: Actualmente solo se permite la puerta de entrada "P1".

---

3. Se selecciona el tipo de ruta que se desea obtener.

Nota: Si el tipo es "Ruta más corta y que maximice las necesidades del usuario", se abrirá una ventana para solicitar las preferencias del usuario.

CAPÍTULO 2

Librerías

2.1 Librerías utilizadas en el proyecto:

2.1.1 Matplotlib:

Matplotlib es una biblioteca completa para crear visualizaciones estáticas, animadas e interactivas en Python.

El uso principal en este proyecto es representar de forma gráfica los nodos que componen la grada del estadio. De esta forma es posible entender de manera más intuitiva las rutas recomendadas, así como facilitar la depuración tanto del modelo de la grada como de las soluciones ofrecidas al usuario.

2.1.2 Networkx:

Networkx es una biblioteca para la creación, manipulación y estudio de grafos y redes.

El uso principal en este proyecto es la creación de grafos compuestos por enlaces que simulen los diferentes pasillos del estadio, así como de las uniones de estos. De este modo se podrá visualizar el grafo de forma gráfica, además de realizar sobre este modificaciones, lecturas y aplicaciones de algoritmos para la obtención de diferentes rutas.

2.1.3 tkinter:

tkinter es módulo Python que permite crear y mostrar interfaces gráficas al usuario.

El uso principal en este proyecto será la creación de un formulario para leer las preferencias de la ruta del espectador.

2.1.4 Numpy:

Numpy es una biblioteca que facilita y acelera el trabajo con datos vectoriales y matrices en Python.

2.1.5 json:

json es un módulo Python utilizado para codificar y descodificar estructuras de datos en formato .json.

2.1.6 threading:

threading es un módulo Python utilizado para crear hilos de paralelismo en los que realizar diferentes tareas simultáneamente.

CAPÍTULO 3

Funciones

Módulo con las funciones usadas en el proyecto.

@author: Pelayo Tiesta

Funciones.**DibujarGrafoGeneral**(*Grafo*, *mostrarPesos*, *sectores*, *peso='weight'*)

Función para dibujar el Grafo General

Parámetros

- **Grafo** (*networkx.classes.multidigraph.MultiDiGraph*) – Grafo a dibujar.
- **mostrarPesos** (*bool*) – Boolean - True si se muestran los pesos
- **sectores** (*Lista*) – Sectores del grafo
- **peso** (*string, optional*) – String con la etiqueta peso. (Default: weight)

Funciones.**DibujarGrafoMasCorta**(*Grafo*, *mostrarPesos*, *origen*, *destino*, *ruta*, *espectador*, *sectores*, *asiento*, *NodoPrefinal_pos*, *Asiento_pos*, *peso='weight'*)

Función para dibujar el grafo con la ruta más corta

Parámetros

- **Grafo** (*networkx.classes.multidigraph.MultiDiGraph*) – Grafo a dibujar.
- **mostrarPesos** (*bool*) – Boolean - True si se muestran los pesos.
- **origen** (*string*) – String - Nodo inicial de la solución.
- **destino** (*string*) – String - Nodo final de la solución.
- **ruta** (*lista*) – Lista - (Longitud [Nodos_Solucion])
- **espectador** (Espectador) – Espectador que accede a un asiento.
- **sectores** (*Lista*) – Lista de sector.
- **asiento** (Asiento) – Asiento del espectador.
- **NodoPrefinal_pos** (*List*) – Posición del nodo Prefinal.
- **Asiento_pos** (*List*) – Posción del asiento.
- **peso** (*string, optional*) – String con la etiqueta peso. (Default: weight)

Devuelve Plt donde se dibuja

Tipo del valor devuelto module

Funciones.**DibujarGrafoMasRapida**(*Grafo*, *mostrarPesos*, *origen*, *destino*, *ruta*, *espectador*, *sectores*, *asiento*, *NodoPrefinal_pos*, *Asiento_pos*, *peso='weight'*)

Función para pintar el grafo con la ruta más rápida.

Parámetros

- **Grafo** (*networkx.classes.multidigraph.MultiDiGraph*) – Grafo a dibujar.
- **mostrarPesos** (*bool*) – Boolean - True si se muestran los pesos.
- **origen** (*string*) – String - Nodo inicial de la solución.
- **destino** (*string*) – String - Nodo final de la solución.
- **ruta** (*lista*) – Lista - (Longitud [Nodos_Soluciçon])
- **espectador** (Espectador) – Espectador que accede a un asiento.
- **sectores** (*Lista*) – Lista de sector.
- **asiento** (Asiento) – Asiento al que se accede.
- **NodoPrefinal_pos** (*List*) – Posición del nodo prefinal.
- **Asiento_pos** (*List*) – Posción del asiento.
- **peso** (*string, optional*) – String con la etiqueta peso. (Default: weight)

Devuelve Plt donde se dibuja

Tipo del valor devuelto module

Funciones.**DibujarGrafoAtributos**(*Grafo*, *mostrarPesos*, *origen*, *destino*, *ruta*, *espectador*, *sectores*, *asiento*, *NodoPrefinal_pos*, *Asiento_pos*, *peso='weight'*)

Función para pintar el grafo según atributos.

Parámetros

- **Grafo** (*networkx.classes.multidigraph.MultiDiGraph*) – Grafo a dibujar.
- **mostrarPesos** (*bool*) – Boolean - True si se muestran los pesos.
- **origen** (*string*) – String - Nodo inicial de la solución.
- **destino** (*string*) – String - Nodo final de la solución.
- **ruta** (*lista*) – Lista - (Longitud [Nodos_Solución])
- **espectador** (Espectador) – Espectador que accede a un asiento.
- **sectores** (*Lista*) – Lista de Sector.
- **asiento** (Asiento) – Asiento al que se accede.
- **NodoPrefinal_pos** (*List*) – Posición del nodo Prefinal.
- **Asiento_pos** (*List*) – Posción del asiento.
- **peso** (*string, optional*) – String con la etiqueta peso. (Default: weight)

Devuelve Plt donde se dibuja

Tipo del valor devuelto module

Funciones.**DibujarGrafoOcupacion**(*Grafo*, *mostrarPesos*, *origen*, *destino*, *espectador*, *sectores*, *asiento*, *nodoParado*, *dibujarEnlacePrefinalAsiento=False*, *NodoPrefinal_pos=None*, *Asiento_pos=None*, *peso='weight'*)

Función para dibujar el grafo de atributos

Parámetros

- **Grafo** (*networkx.classes.multidigraph.MultiDiGraph*) – Grafo a dibujar.
- **mostrarPesos** (*bool*) – Boolean - True si se muestran los pesos.
- **origen** (*string*) – String - Nodo inicial de la solución.
- **destino** (*string*) – String - Nodo final de la solución.
- **espectador** (Espectador) – Espectador que accede a un asiento.
- **sectores** (*Lista*) – Lista de Sector.
- **asiento** (Asiento) – Asiento al que se accede.
- **nodoParado** (*String*) – Nodo en el que está parado el usuario.
- **dibujarEnlacePrefinalAsiento** (*Boolean, optional*) – Bool para dibujar el enlace del nodo prefinal al asiento. (Default: False)
- **NodoPrefinal_pos** (*Tupla, optional*) – Posición del nodo prefinal. (Default: None)
- **Asiento_pos** (*Tupla, optional*) – Posición del asiento. (Default: None)
- **peso** (*string, optional*) – String con la etiqueta peso. (Default: weight)

Devuelve Plt donde se dibuja

Tipo del valor devuelto module

Funciones.**dibujarAsientoGrafoGeneral**(*posicionX_asiento*, *posicionY_asiento*, *plt*)

Funciones.**addnodoPrefinal**(*G*, *posicionUnion*, *sectorEspectador*, *enlacePreFinal*, *fila*, *nombreNodoPrefinal='NP'*)

Función para agregar al grafo un nodo prefinal equivalente a la proyección del asiento en las escaleras.

Parámetros

- **G** (*networkx.classes.multidigraph.MultiDiGraph*) – Grafo a actualizar.
- **posicionUnion** (*Tupla*) – Posición de la proyección del asiento sobre el enlace de su derecha o izquierda.
- **sectorEspectador** (Sector) – Sector sobre el que se encuentra el asiento del espectador.
- **enlacePreFinal** (*Tupla*) – Enlace actual del grafo que se borrará para agregar el nuevo nodo prefinal.
- **fila** (*Integer*) – Valor de la fila donde está el asiento.
- **nombreNodoPrefinal** (*String, optional*) – Nombre del nodo prefinal resultante. Valor por defecto "NP".

Devuelve nombreNodoPrefinal – Nombre del nodo prefinal.

Tipo del valor devuelto String

Funciones.**dibujarRectaNodoPrefinal_Asiento**(*NodoPrefinal_pos*, *Asiento_pos*, *plot*)
Función para dibujar una línea que emule el camino desde la escalera hasta el asiento.

Parámetros

- **NodoPrefinal_pos** (*Tupla*) – Posiciones del nodo prefinal.
- **Asiento_pos** (*Tupla*) – Posiciones del asiento.
- **plot** (*module*) – Plt donde se dubujará la línea.

Funciones.**dibujarDatosEspectadorGeneral**(*Grafo*, *espectador*, *plot*)
Función para dibujar los datos del espectador en el plot

Parámetros

- **Grafo** (*networkx.classes.multidigraph.MultiDiGraph*) – Grafo General.
- **espectador** (Espectador) – Espectador.
- **plot** (*mdoule*) – Plt donde se dibuja.

Funciones.**dibujarDatosEspectadorMasCorta**(*Grafo*, *espectador*, *plot*)
Función para dibujar los datos del espectador en el plot

Parámetros

- **Grafo** (*networkx.classes.multidigraph.MultiDiGraph*) – Grafo General.
- **espectador** (Espectador) – Espectador.
- **plot** (*mdoule*) – Plt donde se dibuja.

Funciones.**dibujarDatosEspectadorMasRapida**(*Grafo*, *espectador*, *plot*)
Función para dibujar los datos del espectador en el plot

Parámetros

- **Grafo** (*networkx.classes.multidigraph.MultiDiGraph*) – Grafo General.
- **espectador** (Espectador) – Espectador.
- **plot** (*mdoule*) – Plt donde se dibuja.

Funciones.**dibujarDatosEspectadorAtributos**(*Grafo*, *espectador*, *plot*)
Función para dibujar los datos del espectador en el plot

Parámetros

- **Grafo** (*networkx.classes.multidigraph.MultiDiGraph*) – Grafo General.
- **espectador** (Espectador) – Espectador.
- **plot** (*mdoule*) – Plt donde se dibuja.

Funciones.**botonAtributosPersonalesClick**(*ventana_self*, *escalerasConBarandillas*, *escalerasEnBuenEstado*, *pasillosAmplios*, *pasillosVentilados*, *pasillosSecos*, *pasillosIluminados*)
Método auxiliar para capturar el click del botón Confirmar en el formulario de atributos

y modificar el diccionario respuestas

Funciones.**leerSectoresCSV**(*fichero*, *Grafo*)
Función para leer los sectores de la grada de un fichero CSV

Parámetros

- **fichero** (*String*) – Fichero .csv con los sectores.
- **Grafo** (*networkx.classes.multidigraph.MultiDiGraph*) – Grafo del estadio.

Devuelve sectores – Lista con los sectores leídos.

Tipo del valor devuelto Lista

Funciones.**importarGrafoJSON**(*Fichero*)
Función para leer el grafo a partir de un fichero JSON

Parámetros **Fichero** (*String*) – Fichero de extensión .json.

Devuelve Grafo – Grafo leído del fichero JSON.

Tipo del valor devuelto networkx.classes.multidigraph.MultiDiGraph

Funciones.**exportarGrafoJSON**(*Grafo*)
Función para generar un fichero JSON a partir del grafo dado.

Parámetros Grafo (*networkx.classes.multidigraph.MultiDiGraph*) – Grafo a exportar.

Devuelve

Tipo del valor devuelto None.

class Funciones.**MenuAtributosPersonales**(*master*, *status*, **options*)
Clase para crear un menú de atributos a puntuar

class Funciones.**Respuestas**
Clase para las respuestas del menú de atributos

rellenar (*escalerasConBarandillas*, *escalerasEnBuenEstado*, *pasillosAmplios*, *pasillosVentilados*, *pasillosSecos*, *pasillosIluminados*)
Función para rellenar los campos de las respuestas

Parámetros

- **escalerasConBarandillas** (*Integer*) – Valor respondido para este campo.
- **escalerasEnBuenEstado** (*Integer*) – Valor respondido para este campo.
- **pasillosAmplios** (*Integer*) – Valor respondido para este campo.
- **pasillosVentilados** (*Integer*) – Valor respondido para este campo.
- **pasillosSecos** (*Integer*) – Valor respondido para este campo.
- **pasillosIluminados** (*Integer*) – Valor respondido para este campo.

Devuelve

Tipo del valor devuelto None.

class Funciones.**ventanaAtributosPersonales**
Clase para dibujar una ventana con los atributos a puntuar

center (*window*)
Función para centrar la ventana

CAPÍTULO 4

Clases

Módulo con las clases usadas en el proyecto.

@author: Pelayo Tiesta

class Clases.**Sector** (*nombre*, *nodoArribaIzquierda*, *nodoArribaDerecha*, *nodoAbajoIzquierda*, *nodoAbajoDerecha*, *filas*, *columnas*, *posicion*)
Clase para crear un objeto sector en el estadio

class Clases.**Asiento** (*sector*, *fila*, *columna*)
Clase para crear un objeto asiento ubicado en el estadio

enlaceProyeccion ()
Función para obtener el enlace de la proyección del asiento.

Muestra Exception – Produce excepción si no existe pasillo derecho ni izquierdo.

Devuelve

Tipo del valor devuelto Enlace - Tupla

nodoPrePreFinal ()
Función para rellenar el nodoPrePrefinal. Se calcula primero si se ira al lado derecho o izquierdo en funciónde la ubicación horizontal del asiento. Posteriormente se calcula si el nodoPrePrefinal será el nodo superioro inferior

Muestra Exception – Se produce si no tiene enlace derecho o izquierdo (Asiento no accesible).

Devuelve

Tipo del valor devuelto String - Nodo PrePrefinal

class Clases.**Espectador** (*ID*, *asiento*, *puertaEntrada*)
Clase para crear un objeto espectador al que se recomendara una ruta

class Clases.**ThreadingOcupacion** (*Grafo*)
Clase para crear un hilo que simule la entrada de personas en la grada

run ()
Método para correr en segundo plano y modificar el peso del grafo con la ocupación.

termínate ()
Función para terminar el hilo