

HANDITOUR

ALEJANDRO GONZÁLEZ DAMÍN



Curso 20/21

[ANTECEDENTES 2](#_Toc73296094)

[DESCRIPCIÓN 3](#_Toc73296095)

[Ilustración 1: index 3](#_Toc73296096)

[Ilustración 2: Ciudad 4](#_Toc73296097)

[Ilustración 3: detalle monumento 1](#_Toc73296098)

[Ilustración 4: alojamientos 2](#_Toc73296099)

[Ilustración 5: detalles de un alojamiento 3](#_Toc73296100)

[Ilustración 6: aparcamientos 4](#_Toc73296101)

[Ilustración 7: transporte 5](#_Toc73296102)

[Ilustración 8: login y registro 6](#_Toc73296103)

[Instalación y preparación 7](#_Toc73296104)

# ANTECEDENTES

La idea inicial que tuvimos para realizar este proyecto fue, hacer una aplicación web para el beneficio del colectivo de discapacitados. Con esta premisa nació la idea de crear un portal para facilitar hecho de poder viajar. De ahí nació la idea de handitour, que es un acrónimo de las palabras inglesas handicap minusválido y tour viaje.

Para crear haditour nos hemos basado en ciudades las cuales hemos podido visitar con anterioridad como son Cádiz, Sevilla, Valencia, Madrid, Barcelona o Londres. En cada una ellas, se podrá ver los puntos de interés de cada una, como monumentos, alojamientos, transportes…

El proyecto esta programado en Django, un framework de Python que usa MVC, además de Django, handitour contiene en casi todas sus pantallas un mapa dinámico programado en Javascript junto con peticiones Ajax y jqury que muestran el contenido dinámico deseado dependiendo de las premisas de cada pantalla.

En cuanto al estilo de la aplicación hemos usado y personalizado el sistema de rejillas y las clases de bootstrap 4.

# DESCRIPCIÓN

En esta sección del documento, desarrollaremos el resultado del proyecto. Iremos viendo página por página y analizare su contenido. Para empezar el index de la aplicación web.

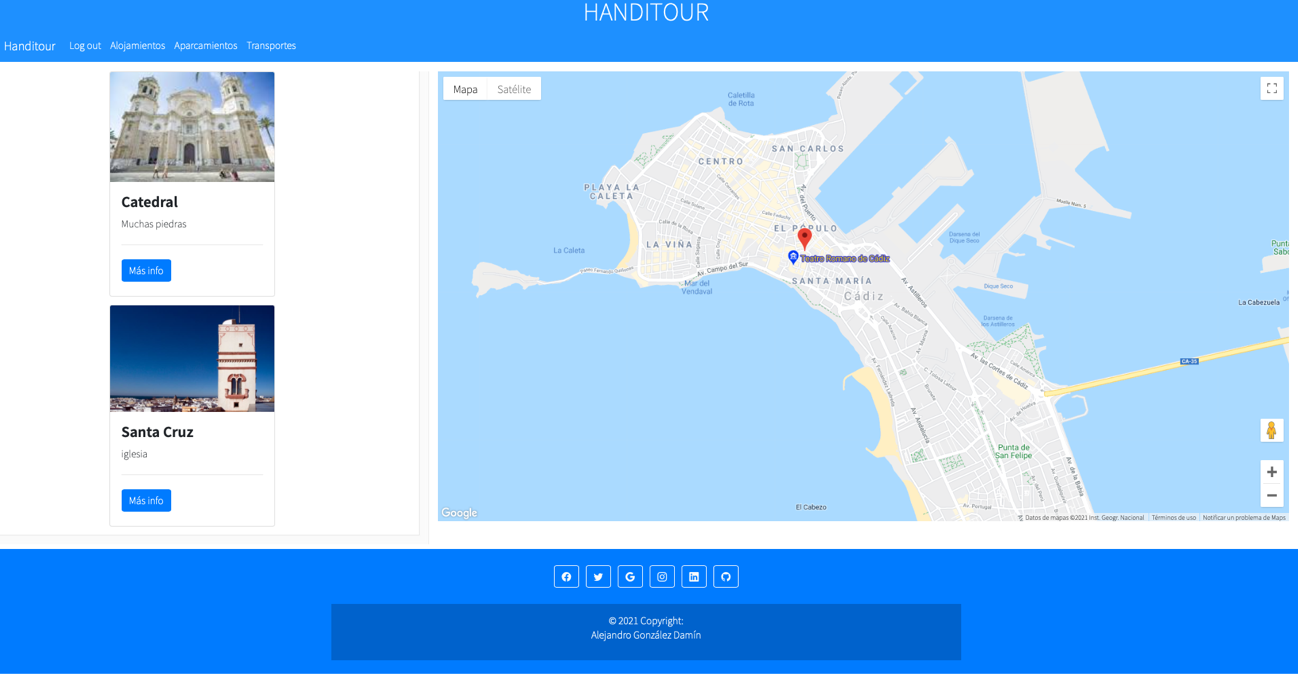


## Ilustración 1: index

En la ilustración 1 se muestra la disposición de nuestro index. En la parte superior de la pantalla se encuentra el header. Este se compone del título en el centro del mismo y la barra de navegación que muestra el título que cuando se hace click redirecciona hacia el index, al lado nos encontramos un ítem del menú que nos llevará a hacer log in o log out según si encuentra o no sesión del usuario.

En la parte central de la ilustración encontramos una carta por cada ciudad insertada en nuestra base de datos. Al hacer el evento hover sobre la carta apreciamos un cambio en la opacidad de la imagen mostrando el nombre de la ciudad donde se encuentre el puntero del ratón.

Para acabar la explicación de nuestra pantalla inicial, explicaremos la composición de nuestro footer. Su composición es simple, está compuesto por varios iconos que apuntan a las redes sociales del contenido de nuestra aplicación junto con el nombre del autor del portal.



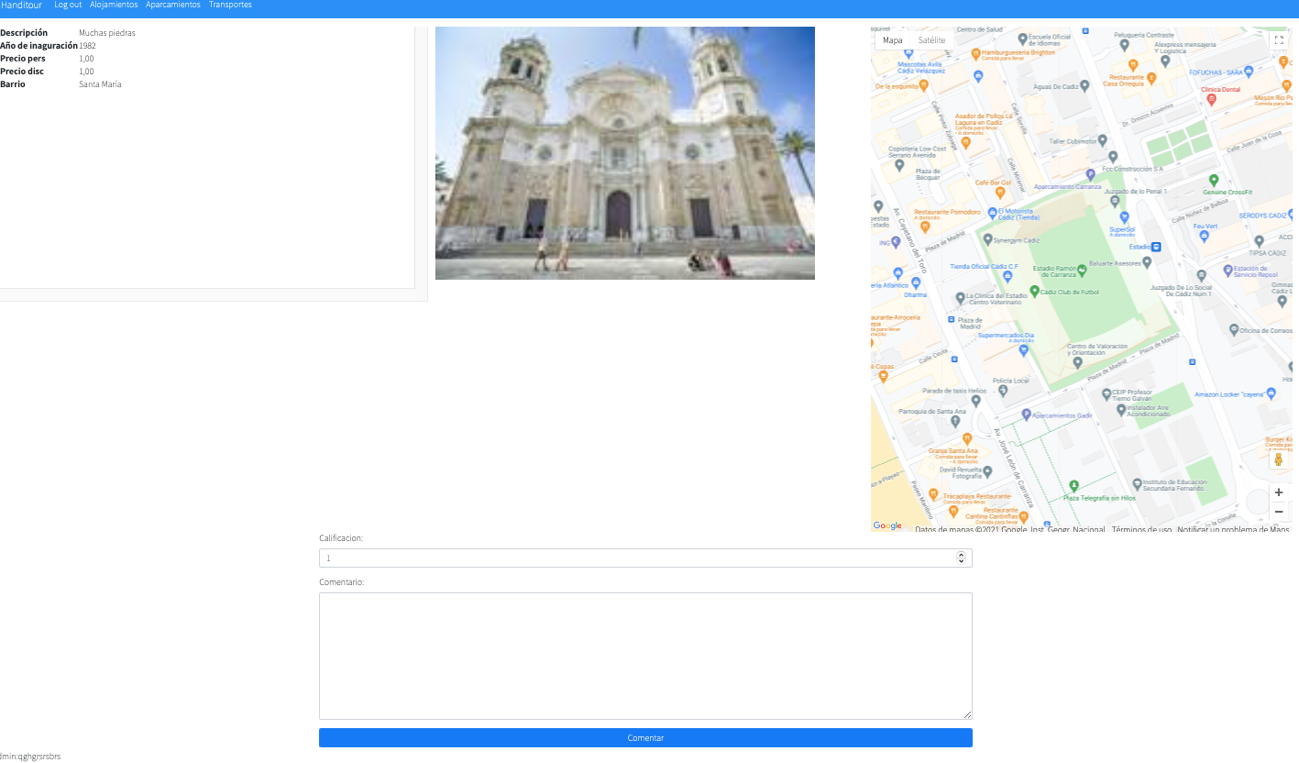
## Ilustración 2: Ciudad

En la ilustración 2 podemos apreciar la vista de una ciudad. En el header nos encontramos prácticamente con el mismo header que en la ilustración 1, salvo que esta vez la barra de navegación tiene diferentes ítems con las diferentes secciones que más adelante explicaremos detenidamente.

En el cuerpo de esta página tenemos dos grandes partes muy bien diferenciadas:

1. En la parte izquierda de la imagen podemos apreciar un listado con los monumentos que podemos visitar en la ciudad seleccionada. Este listado está compuesto por un tipo de carta con más elementos que las cartas desarrolladas en el index. Estas se componen de título, que muestra el nombre del monumento, subtitulo, en el que aparece una breve descripción de la historia del monumento y un botón de más información que redirecciona a la pantalla de detalle, que ampliaremos en posteriores pantallas.
2. En la parte de la derecha de la imagen nos encontramos con un mapa que sitúa todos los monumentos del listado comentado anteriormente. El mapa está programado en javascript usando la api de google maps, concretamente la de geolocalización. Para mostrar todos los monumentos insertados en la base de datos, también he tenido que usar jquery junto con Ajax, con todo esto conseguimos enviarle al mapa la posición exacta de cada monumento.

Para acabar con la explicación de esta pantalla también cabe destacar la parte del footer que es igual que hemos expuesto anteriormente en la ilustración uno, correspondiente al index.



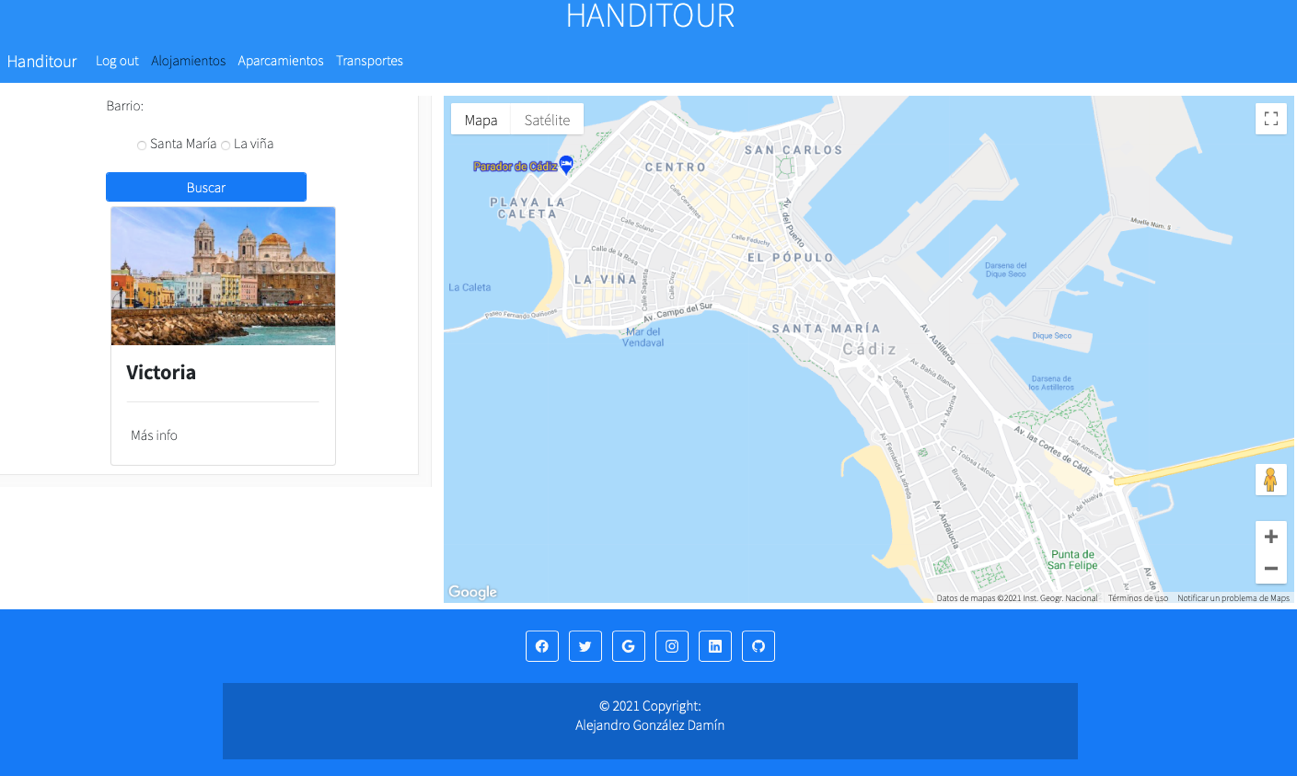
## Ilustración : detalle monumento

Para empezar a comentar esta ilustración, cabe destacar que aunque no se pueda apreciar en la misma tiene header y footer, pero por las dimensiones de la página no se han podido ilustrar correctamente. La composición de los mismos ha sido desarrollada exactamente igual que en la ilustración 2. Dicho esto pasaremos a analizar la parte central de nuestra página de detalle.

En la parte justo debajo del header tenemos la información relativa al monumento seleccionado en la pantalla anterior. Con tres partes muy diferenciadas:

1. En la parte más a la izquierda de la pantalla nos encontramos una tabla en la que se describe toda la información relativa al monumento seleccionado.
2. En la parte central de la pantalla tenemos la imagen característica del monumento.
3. En la parte derecha de la pantalla nos encontramos con un mapa, el cual está programado igual que los anteriores salvo con una excepción, no le hemos querido meter ningún filtro para que el usuario tenga referencias sobre lo que hay alrededor del monumento, por ejemplo restaurantes, souvenirs, transportes, etc.

En esta pantalla si el usuario está logueado le aparecerá un formulario donde podrá opinar sobre el monumento y sus alrededores. Por el contrario si el usuario no está logueado le aparecerá un párrafo con una redirección hacia la pantalla de log in. Tanto como si ha hecho log in como no le aparecerá los comentarios hechos por otros usuarios.

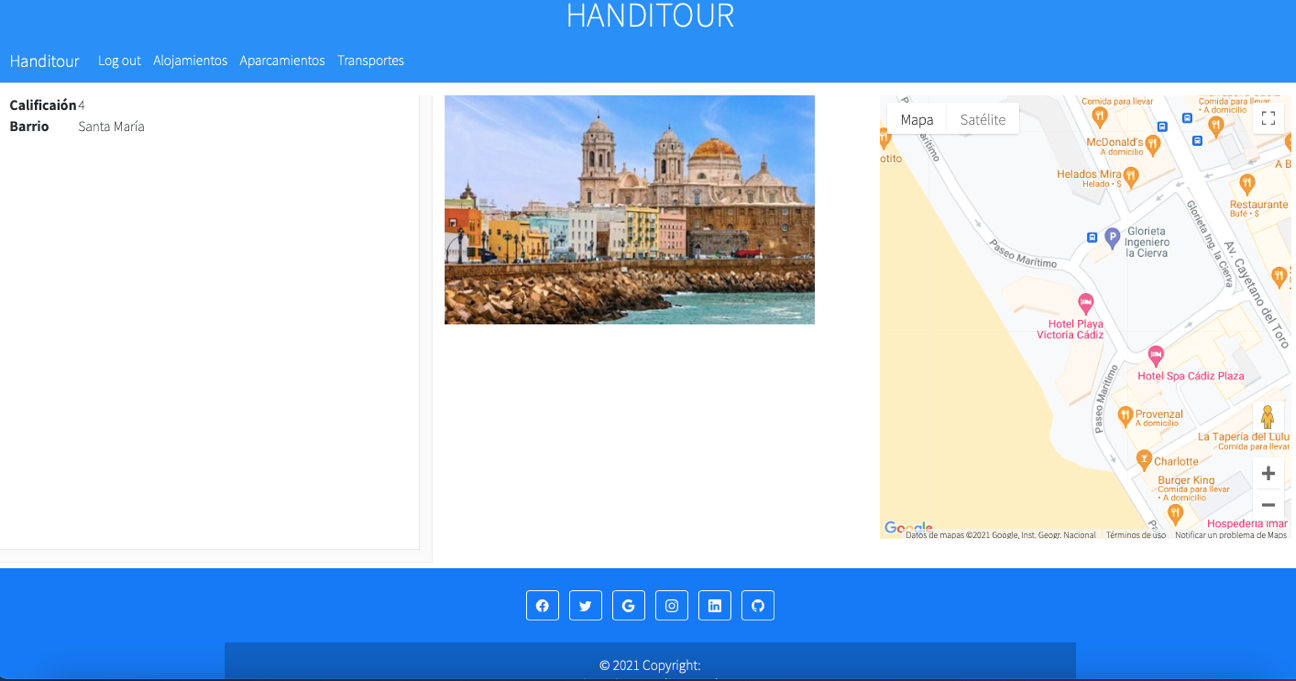


## Ilustración : alojamientos

En este apartado tenemos los alojamientos de una ciudad. Al igual que en los anteriores apartados footer y header son iguales.

En esta pantalla podemos diferenciar dos grandes partes:

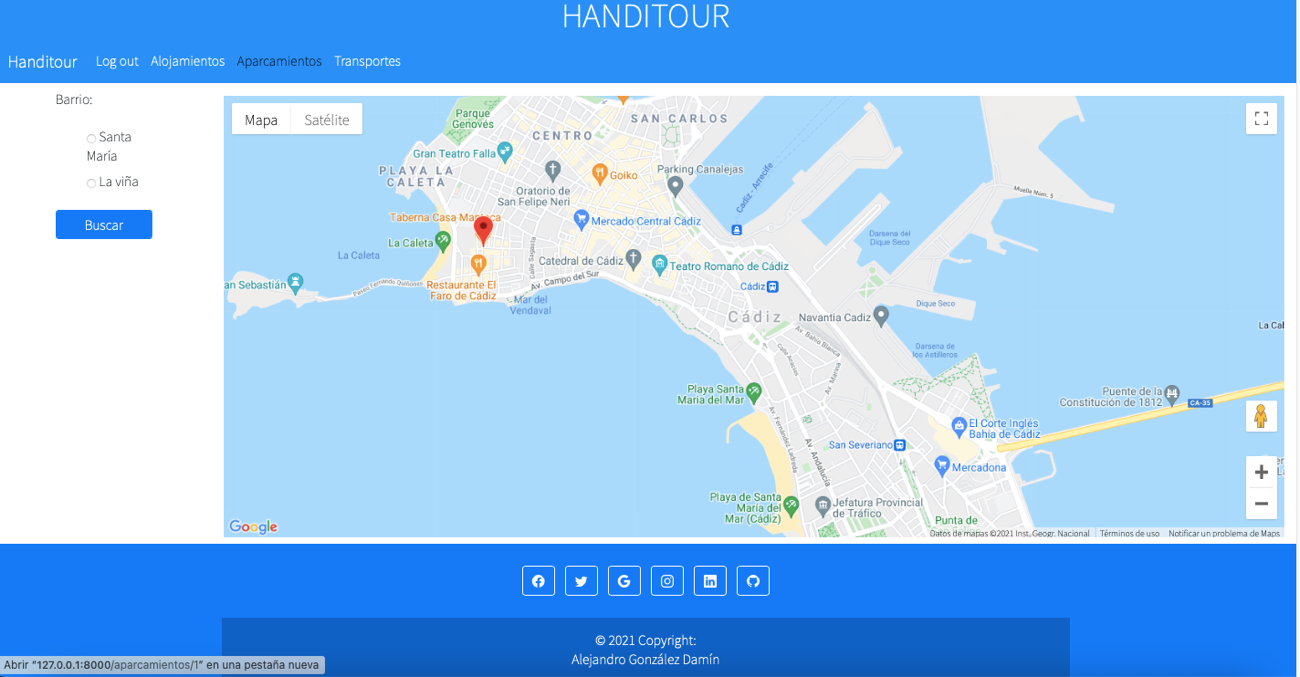
1. En la izquierda tenemos un buscador y todos los alojamientos de una ciudad. Si seleccionamos un barrio nos muestra todos los alojamientos de ese barrio.
2. En la parte derecha se encuentra el mapa programado al igual que los anteriores, con una diferencia si el usuario desea hacer uso del buscador el barrio deseado con sus alojamientos correspondientes. También el mapa tiene filtrado para que nada más muestre alojamientos en esta pantalla.



## Ilustración 5: detalles de un alojamiento

En esta pantalla podemos visitar las características de un alojamiento en concreto. En cuanto a la estructura de la misma header y footer es común a todas las páginas, por lo tanto no tenemos mucho más que decir ya que lo hemos mostrado en anteriores punto de este apartado.

En el centro de la pantalla como es una vista de detalle, lo hemos hecho coincidir con la vista de detalle de un monumento. Para repasar dicha estructura el centro de la pantalla lo hemos dividido en tres columnas, en la primera de ellas tenemos los datos relativos al alojamiento, seguidamente se muestra la imagen del mismo y en la última su localización exacta.



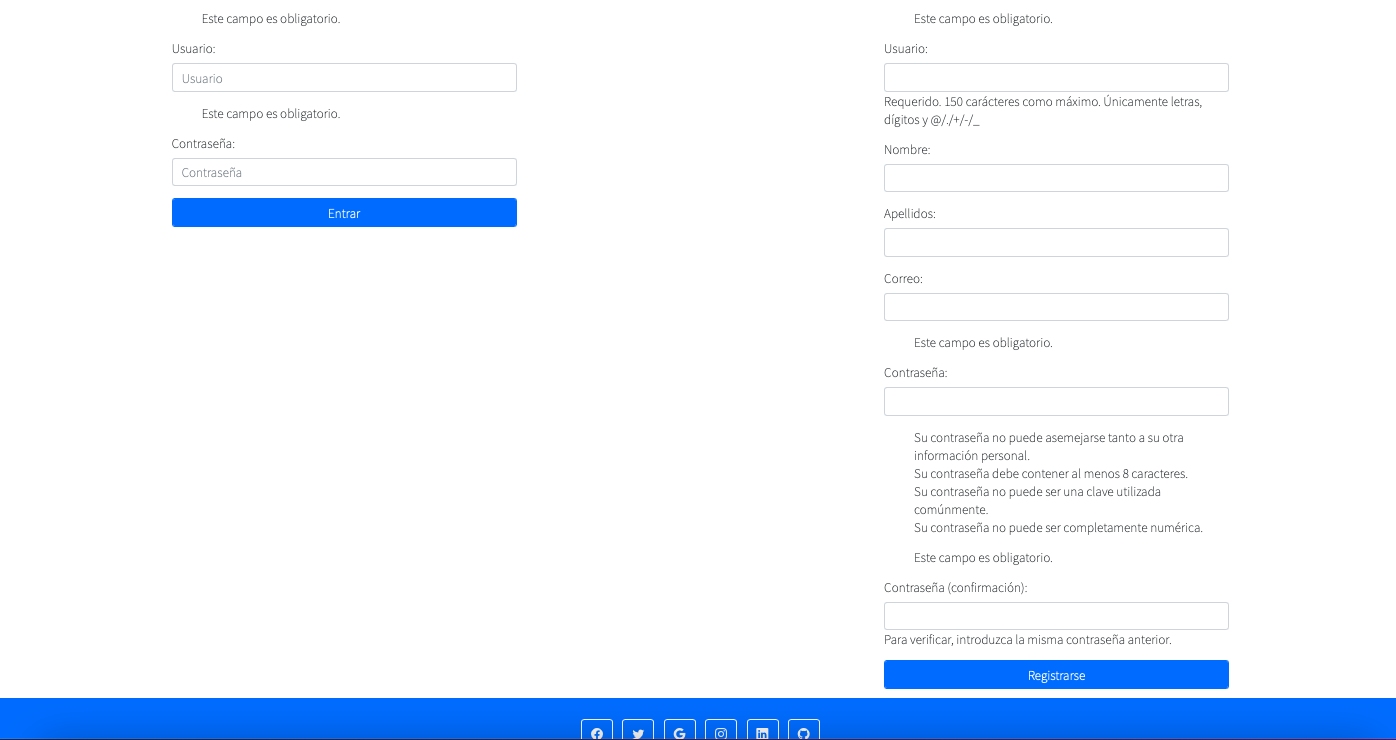
## Ilustración 6: aparcamientos

En esta pantalla header y footer son exactamente igual que en las anteriores. El objetivo de esta pantalla es que el usuario tenga a mano los aparcamientos de minusválidos que hay en la ciudad. Para hacer más fácil al usuario encontrar aparcamiento le hemos programado al igual que en puntos anteriores un filtro de barrios que centrará el mismo en el mapa.



## Ilustración 7: transporte

En la siguiente pantalla al igual que en las anteriores header y footer son igual que en los apartados anteriores. En esta pantalla el usuario puede consultar las líneas de autobuses de la ciudad y su localización mediante un mapa. También podrá filtrar las paradas por lo barrios y consultar que líneas que pasan por cada parada.



## Ilustración 8: login y registro

Con esta ilustración llegamos al final de esta sección, pero no menos importante. Para que el usuario pueda hacer algunas interacciones con nuestro portal, primero deberá haberse registrado en el mismo y haber iniciado sesión. En cuanto al desarrollo de esta pantalla django nos proporciona un usuario predeterminado y los formularios propios para que esta tarea sea más llevadera. A parte hace el control de inicio de sección y final de sección mediante sus respectivas vistas.

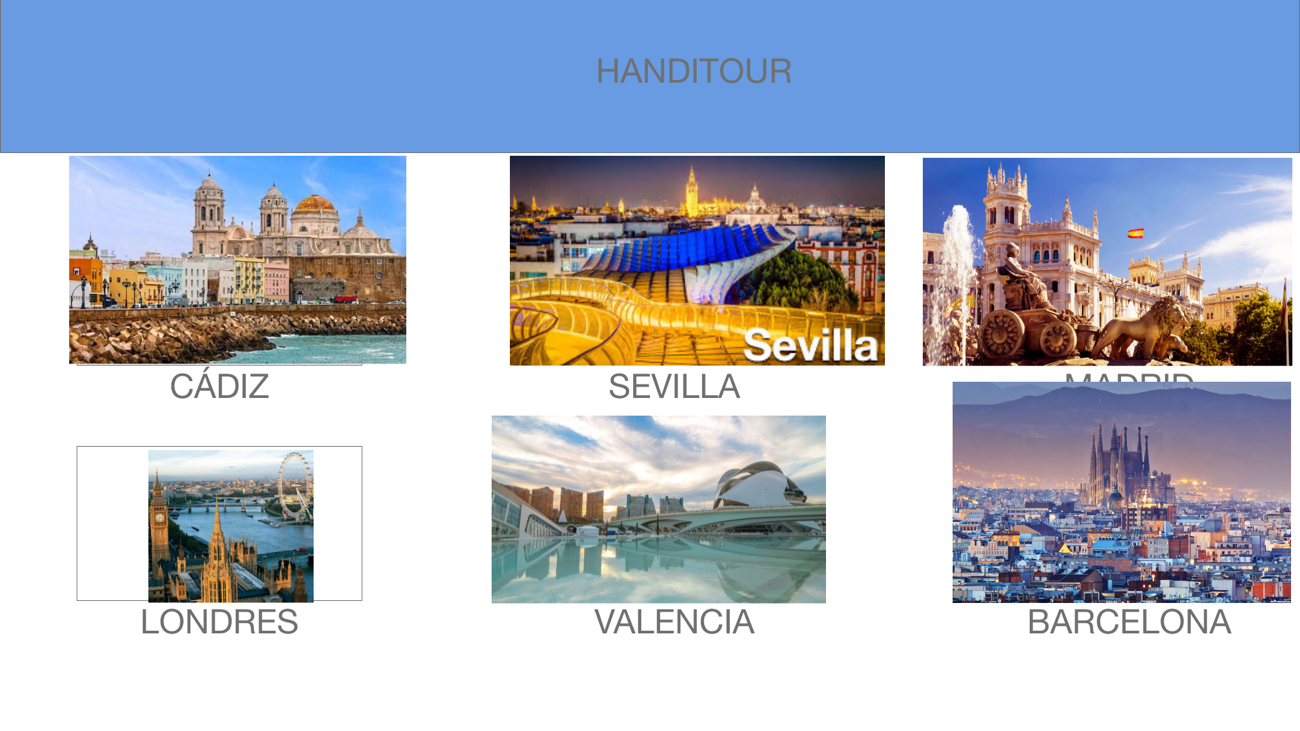
# Instalación y preparación

Para empezar a trabajar en este proyecto primero hemos tenido que verificar que teníamos Python instalado en nuestro ordenador. Una vez verificado esto mediante un pip instalamos el framework django. Esto nos dejará un archivo muy importante que usaremos a lo largo de todo nuestro desarrollo que se llama manage.py. con el uso del comando Python manage.py startapp junto con el nombre de nuestra aplicación. Esto último creará la carpeta con todos los ficheros necesarios para ejecutar la aplicación. Seguidamente haciendo uso de nuevo del manage.py “fabricamos las migraciones”, con el comando Python manage.py makemigrations. Una vez que tenemos las migraciones preparadas usamos el comando Python manage.py migrate, el conjunto de estos comandos inicializará la base de datos de nuestro proyecto. Una vez creada la app habrá que instalarla en el proyecto. Para hacer esto tenemos que abrir el módulo settings.py, buscar la línea dónde tenemos la lista de apps instalada y añadimos un nuevo ítem compuesto de comilla simple el nombre de la app en minúscula seguido de punto app seguido de otro punto con el nombre de nuestra aplicación con la primera letra en mayúscula junto con Config.

En cuanto a la parte sobre control de versiones, nos hemos creado una cuenta de github con el dominio de correo electrónico proporcionado por el instituto. Seguidamente en la misma cuenta nos hemos creado un repositorio, que lo hemos puesto en común con el resto de alumnos que presentan el proyecto en esta convocatoria. Después de la puesta en común de nuestro proyecto, en la carpeta local del mismo hemos usado el comando git init, con el que le decimos a nuestra carpeta local que esa carpeta va a ser una carpeta remota. Una vez hecho esto ya sólo queda clonar el repositorio que hemos creado en nuestra carpeta local y usamos los comandos git commit y git push, que sería la subida inicial de nuestro código creado en los pasos anteriores. En cuanto a la frecuencia que hemos ido actualizando el repositorio ha sido un commit push cada vez que acabábamos una sesión de desarrollo.

# Guías de estilos y prototipado

En esta sección vamos a comentar las guías de estilos de nuestra aplicación y el desarrollo de los prototipos. En cuanto a las guías de estilos hemos modificado la guía que nos proporciona boostrap 4. Esta herramienta nos proporciona una guía de estilos híbridas juntando dos técnicas avanzadas para la colocación y responsividad, como son grid y flexbox. El grid nos divide los contenedores como máximo en 12 columnas, así nos permite colocar los elementos de nuestra página de forma fácil y directa. También a la hora de cambiar la resolución de la pantalla nada más que tenemos que decirle cuantas columnas queremos dependiendo del tamaño de la pantalla. Junto a grid boostrap 4 hace uso de flexbox, esta herramienta les da flexibilidad a los contenedores para hacer más fácil la disposición de columnas que nos proporciona el grid. A continuación les mostramos los mockups que diseñamos antes de empezar a desarrollar en código nuestro proyecto. Debido a la repetición de las pantallas expuestas en el punto de descripción de este documento, hemos diseñado sólo y exclusivamente aquellas que no se repiten.



## Ilustración : prototipo index



## Ilustración : prototipo ciudad



## Ilustración : prototipo monumento

## 

## Ilustración : prototipo alojamientos



## Ilustración : prototipo aparcamientos

## 

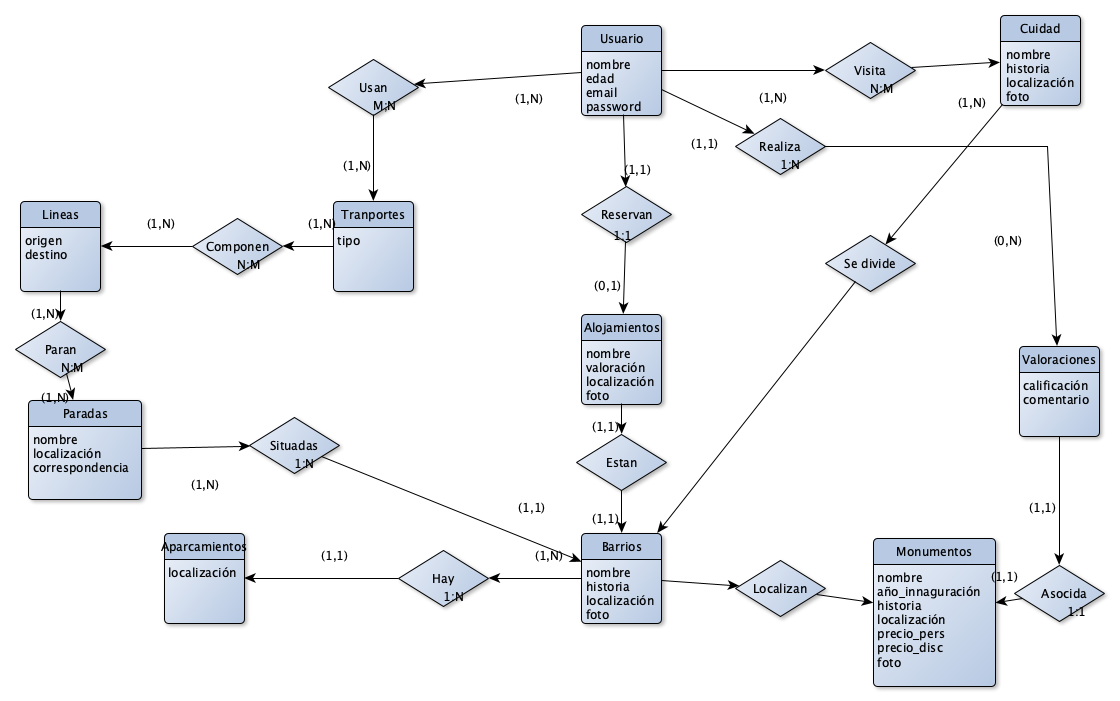
## Ilustración : prototipo transporte

# Diseño

En esta sección describiremos los casos de uso y el diagrama entidad relación de nuestro proyecto. El portal tiene principalmente 12 casos de uso, los cuales son los siguientes:

1. Registro: en este caso de uso el usuario podrá registrarse en nuestro portal.
2. Log in: el usuario una vez que haya hecho el registro podrá entrar como usuario en nuestro portal.
3. Index: este caso de uso muestra todas las ciudades insertadas en la base de datos.
4. Ciudad: este caso de uso lista todos los monumentos de una ciudad insertados en la base de datos.
5. Mapa: señala la situación geográfica en las distintas pantallas de nuestro portal.
6. Detalle de monumento: se muestra la información completa de un monumento.
7. Comentario a un monumento: el usuario puede hacer comentarios sobre un monumento.
8. Alojamientos: lista los alojamientos de una ciudad.
9. Filtro de barrio: filtra el mapa por barrios.
10. Aparcamientos: muestra la situación geográfica de los aparcamientos de una ciudad.
11. Transportes: lista las líneas de autobuses de una ciudad.
12. Log out: destruye la sección de un usuario.

A continuación mostraremos el diagrama entidad relación de la base de datos de nuestro proyecto.



## Ilustración 15: diagrama entidad relación

En la ilustración podemos ver el diagrama entidad relación de nuestro proyecto, este se compone de 10 entidades las cuales describiremos junto con sus relaciones a continuación:

1. Usuario: sus atributos son nombre, edad, email y contraseña.
2. Ciudad: compuesta por nombre, historia, localización y foto. Junto a usuario forma una relación de muchos a muchos.
3. Valoraciones: compuesta por calificación y comentarios. Que junto a usuario forma una relación de muchos a muchos.
4. Monumentos: se compone de nombre, año de inauguración, historia, localización, precio persona, precio discapacitado y foto. Junto a valoraciones forma una relación de uno a muchos.
5. Barrios: se compone de nombre, historia, localización y foto. Junto a monumentos forman una relación de uno a muchos.
6. Aparcamientos: compuesto de localización. Junto a barrios forma una relación de uno a muchos.
7. Alojamientos: compuesto por nombre, valoración, localización y foto. Junto a barrio forma una relación de uno a muchos.
8. Paradas: que consta de nombre, localización y correspondencia. Junto a barrios forma una relación de uno a muchos.
9. Líneas: sus atributos son origen y destino. Junto a paradas forman una relación de muchos a muchos.
10. Transportes: su único atributo es tipo. Junto a líneas forman una relación de muchos a muchos.

# Desarrollo

En cuanto a la secuencia de desarrollo que hemos usado en este proyecto, hemos ido encadenando una página con otra, es decir, si hemos desarrollado una lista a continuación desarrollamos los detalles de cada ítem de la lista. De este modo hemos obtenido el resultado final.

En cuanto a los conflictos que nos han surgido durante el desarrollo del proyecto, ha habido varios puntos a comentar en este apartado. Primeramente surgieron conflictos a la hora de desarrollar el mapa, ya que en el javascript no podía meter las variables de django si poníamos el script independiente del html, dado que estas nada mas que funcionan en el html. Probé una aplicación del propio django para imprimir los mapas llamada django-easy-map, esta aplicación funcionaba pero a la hora de tener un mapa con dos ubicaciones cuando recorría la consulta imprimía una mapa por cada iteración de la consulta. La solución parcial a este problema fue meter el código javascript en nuestro html, pero claro desde el punto de vista como programador no era una buena práctica. Definitivamente se nos ocurrió crear funciones javascript en archivos independientes con los argumentos necesarios y desde el html llamaba a dichas funciones pasándole la variable django como argumento. Otro conflicto que nos ha parado el desarrollo unos pocos de días fue crear el filtro de barrios. Lo hicimos con un formulario django al cuál le teníamos que pasar la id de la ciudad para que nos devolviera los barrios de la misma. La complicación estuvo en que no encontrábamos la forma de pasar de una ciudad a la lista de barrios, pero tubo fácil solución django proporciona un atributo llamado set para los casos de relaciones de uno a muchos. Mediante este atributo se solucionó nuestro problema.