



Proyecto Apparking

DESARROLLO DE APLICACIONES WEB

SERGIO SOTERAS SERRANO

TUTOR: LUIS MIGUEL MORILLAS

https://github.com/SergioSoteras/proyecto_apparking

Contenido

| | | |
|--------|------------------------------------|----|
| 1. | DESCRIPCIÓN DEL PROYECTO | 1 |
| 1.1. | ABSTRACT | 2 |
| 2. | DOCUMENTO ACUERDO | 3 |
| 3. | ANÁLISIS Y DISEÑO | 7 |
| 3.1. | DIAGRAMA DE CASOS DE USO | 7 |
| 3.2. | DIAGRAMA DE CLASES | 19 |
| 3.3. | DIAGRAMA DE ENTIDAD RELACIÓN | 20 |
| 3.4. | MAPA DE NAVEGACIÓN | 21 |
| 3.5. | GUÍA BÁSICA DE ESTILO | 22 |
| 3.6. | MOCKUPS | 24 |
| 4. | IMPLEMENTACIÓN Y PRUEBAS | 27 |
| 4.1. | IMPLEMENTACIÓN | 27 |
| 4.2. | PRUEBAS..... | 28 |
| 4.2.1. | Pruebas integradas..... | 28 |
| 4.2.2. | Pruebas en la app | 30 |
| 5. | DOCUMENTACIÓN | 39 |
| 5.1. | MANUAL DE INSTALACIÓN..... | 39 |
| 5.2. | MANUAL DE USUARIO..... | 40 |
| 5.2.1. | Grupos | 40 |
| 5.2.2. | Usuarios..... | 41 |
| 5.2.3. | Tablas de Parking..... | 43 |
| 6. | CONCLUSIONES | 45 |

1. DESCRIPCIÓN DEL PROYECTO

El proyecto Apparking es una aplicación web que nace de la necesidad de una comunidad de gestionar su aparcamiento subterráneo, que ofrece al administrador o presidente la posibilidad de distribuir las plazas de aparcamiento y los clientes de una manera rápida y sencilla.

Se trata de una urbanización de nueva obra, en la que constantemente entran nuevos inquilinos y tienen la posibilidad de elegir plaza de aparcamiento, y el presidente siempre se hace un lío con los papeles para ver las plazas disponibles y adjudicarlas.

Enseguida se me ocurrieron ideas para facilitarle el trabajo y tan solo haciendo un par de clics. Conforme íbamos concretando la idea de la aplicación, me interesaba más, puesto que veía bastante progresión en ella, podría seguir desarrollando la aplicación para llevarla a otros aparcamientos, incluso un modelo de negocio añadiendo reservas y pagos por parte de los clientes.

El funcionamiento de la aplicación comienza con el *superuser*, al que se ha llamado *admin*, el cual tendrá acceso al panel de control añadiendo */admin* a la url, donde podrá crear las plazas, clientes y dimensiones de las plazas, y los usuarios y sus permisos.

También dispone de accesos directos en la aplicación a un formulario para crear clientes de manera más fluida sin tener que acceder al panel de control.

Los usuarios podrán visualizar el plano del aparcamiento en directo, donde podrán comprobar cuantas plazas quedan disponibles, comprobar el cliente de una plaza ocupada y el listado de clientes del aparcamiento y utilizar un buscador de clientes.

FUNCIONALIDADES

- Registro de usuarios.
- Los usuarios podrán ver el plano del aparcamiento.
- Los usuarios podrán acceder a un formulario de contacto.
- Los usuarios deben registrarse para ver detalles de clientes.
- Sólo el *admin* tendrá acceso a crear, modificar o eliminar clientes desde la web y adjudicarles una plaza.
- Acceso a un panel de control para el *admin*.
- Tiempo de respuesta mínimo.

1.1. ABSTRACT

The main purpose of 'Apparking' is to offer a tailored management of a parking lot in a quick and easy way.

It allows a single customisation of the parking lot adjusted to the layout of the parking spaces, which makes the application unique and innovative. You can check the availability of parking spaces at a glance and find out which customer is occupying them.

You are also one click away from checking the customers in the parking lot, seeing their details and editing them.

It has a role-based access control (RBAC) based on a security function to control user access to tasks usually restricted to the superuser (admin).

Currently, only the admin can modify the application data, but in the future, it will have a payment gateway so that users can also book and pay for their parking slots.

2. DOCUMENTO ACUERDO

REQUISITOS FUNCIONALES QUE DEBE CUMPLIR LA APLICACIÓN

a) ADMIN

- *Login* y registro de usuarios.
- Panel de control para gestionar usuarios, clientes, plazas y dimensiones de las plazas.
- El *admin* podrá crear, modificar o eliminar clientes desde la lista de clientes de la web.
- Visualizar el plano del aparcamiento y las plazas disponibles y ocupadas, teniendo acceso directo a la información del cliente de las plazas ocupadas, pudiendo modificar dicha información, y acceso directo a la creación de un nuevo cliente en una plaza disponible.

b) USUARIO

- *Login* y registro de usuarios.
- Podrá visualizar la lista de clientes, así como sus detalles y utilizar el buscados de clientes.
- Visualizar el plano del aparcamiento y las plazas disponibles y ocupadas, teniendo acceso directo a la información del cliente de las plazas ocupadas.
- Acceso al formulario de contacto.

c) INVITADO

- *Login* y registro de usuarios.
- Visualizar el plano del aparcamiento y las plazas disponibles y ocupadas.
- Acceso al formulario de contacto.

REQUISITOS NO FUNCIONALES DE LA APLICACIÓN

- Interfaz amigable y sencilla.
- Tiempos de carga cortos.
- Adaptabilidad a distintos dispositivos (*Responsive*).
- Compatibilidad entre navegadores.
- Accesibilidad.

PLANIFICACIÓN TEMPORAL DEL PROYECTO



Se ha dividido el proyecto en distintas fases, de forma que pueda realizar una estimación de cuánto tiempo se va a dedicar a cada una de ellas.

En la fase de “Planificación” se definen todas las bases sobre las que se va a trabajar durante todo el proceso.

En la segunda fase se definen los aspectos del diseño de la aplicación mediante la elaboración de diagramas y *mockups*.

En la fase de desarrollo se programa la aplicación y se implementan todas las funcionalidades que harán que el proyecto funcione.

En la fase de mejoras se testea la aplicación y se corrigen los errores encontrados.

En la fase de documentación se realiza la documentación del proyecto, así como distintos comentarios aclaratorios sobre el código.

En la última fase se trata de desplegar el proyecto.

RECURSOS UTILIZADOS

Durante el proceso de desarrollo, se ha hecho uso de los siguientes recursos:

- Conexión a internet.
- Hardware: Ordenador portátil Lenovo i5 8GB RAM.
- Software:
 - Microsoft Word: Memoria del proyecto
 - Visual Studio Code: Entorno de desarrollo
 - Framework: Django
 - Lenguaje: Python 3.10
 - Oracle VM VirtualBox: Servidor
 - Balsamiq: Mockups
 - Draw.io: Diagramas

METODOLOGÍA

Para la realización de este proyecto he optado por el uso de metodologías ágiles, decantándome por SCRUM.

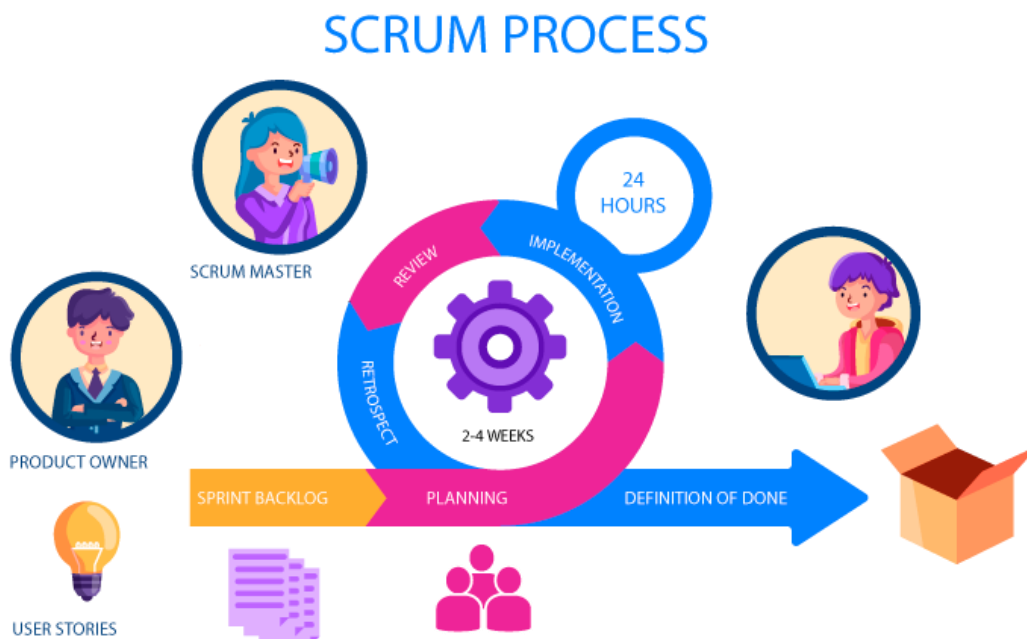
Se caracteriza por ser la «metodología del caos» que se basa en una estructura de desarrollo incremental, esto es, cualquier ciclo de desarrollo del producto y/o servicio se desgrana en «pequeños proyectos» divididos en distintas etapas: análisis, desarrollo y testing.

En la etapa de desarrollo encontramos lo que se conoce como interacciones del proceso o Sprint, es decir, entregas regulares y parciales del producto final.

Esta metodología permite abordar proyectos complejos que exigen una flexibilidad y una rapidez esencial a la hora de ejecutar los resultados. La estrategia irá orientada a gestionar y normalizar los errores que se puedan producir en desarrollos demasiado largos, a través de, reuniones frecuentes para asegurar el cumplimiento de los objetivos establecidos.

Las reuniones son el pilar fundamental de la metodología, donde diferenciamos entre: reuniones de planificación, diaria, de revisión y de retrospectiva, la más importante de todas ellas, ya que, se realiza después de terminar un sprint para reflexionar y proponer mejoras en los avances del proyecto.

Los aspectos clave por los que se mueve el SCRUM son: innovación, flexibilidad, competitividad y productividad.



ESTUDIO DE MERCADO

En caso de no ser una aplicación pedida por un cliente, sino una propuesta de innovación, existen muchas aplicaciones para reservar plaza en los distintos aparcamientos, pero ninguna ofrece un servicio sobre cada aparcamiento, no muestra las plazas disponibles ni el plano, solo introduces el día y el tiempo y si hay alguna plaza disponible te la adjudican, no puedes elegir.

Destacan como competencia directa *Parkimeter*, *Parkapp* o *OnePark*.



De momento el cliente solo ha pedido una gestión del aparcamiento para un *superuser*, que será el único que podrá modificar la aplicación, pero no se descarta en un futuro que sean los propios clientes quienes puedan reservar su plaza.

Existe la posibilidad de expandir la aplicación a un parking público, añadiendo pasarela de pago dónde los clientes podrán reservar y realizar el pago desde la web.

PRESUPUESTO

Acorde al diagrama de Gantt previamente realizado, se ha demorado 6 semanas en realizar la aplicación.

Un programador junior cobra en torno a los 13€ a la hora, por tanto:

Coste de desarrollo = 30 días laborales x (13€ x 8 horas) = 3120€

Coste de mantenimiento:

Hosting:

- Compartido: 3 a 8 €/mes
- VPS: 15 a 28 €/mes

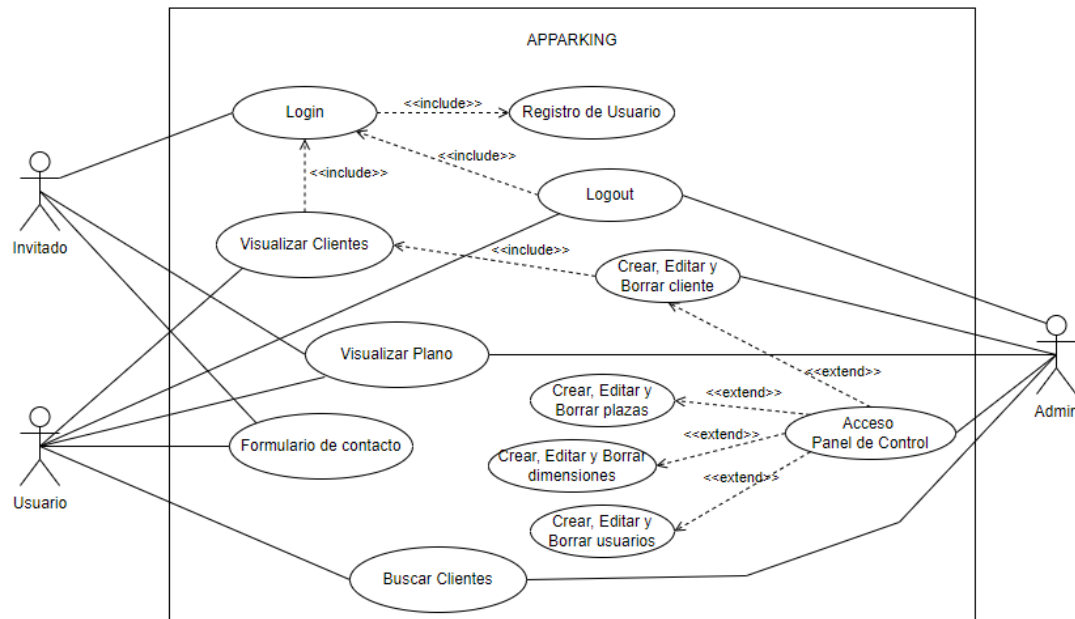
Dominio:

- Español (.es): 8 €/año
- No español (.com, .org...): 10 a 12 €/año

3. ANÁLISIS Y DISEÑO

3.1. DIAGRAMA DE CASOS DE USO

Mediante el siguiente diagrama de casos de uso vamos a poder identificar las distintas acciones que pueden realizar los “actores” que intervienen en la aplicación.



En primer lugar, los invitados tendrán acceso al formulario de contacto para cualquier duda o aclaración sobre la aplicación y podrán visualizar el plano del aparcamiento con su disponibilidad, pero no podrán ver el propietario de la plaza que se encuentra ocupada. Para ello debe estar logueado, y en su defecto, registrado.

Los usuarios registrados pueden acceder a la aplicación a través de un *login*. En caso de no recordar la contraseña existe una opción para ello.

Una vez el usuario está identificado, podrá visualizar a los clientes del aparcamiento y parte de sus datos (algunos están ocultos por la protección de datos), tener acceso al buscador de clientes, tanto por nombre como por apellido y cerrar la sesión, además de los casos de uso de los que disfrutaban los invitados.

En último lugar y más importante se encuentra el *admin*, el *superuser* de la aplicación, que tiene libertad absoluta a la hora de crear, modificar o eliminar clases de la aplicación (Usuarios, Plazas, Dimensiones y Clientes) desde el panel de control o, por facilitar el uso, desde la propia web tiene accesos directos para editar los clientes, que solo puede

ver el *admin*. Como no es recomendable modificar las plazas ya que podría estropear el plano del aparcamiento, solo podrá editarse desde el panel de control.

CASOS DE USO:

| |
|--|
| Nombre: REGISTRO DE USUARIO |
| Descripción: Registro de usuario en el sistema. |
| Actores: Invitado. |
| Precondiciones: El invitado selecciona la opción de Registrar. |
| Curso normal: <ol style="list-style-type: none">1. Escoge opción de Registrar.2. El sistema solicita rellenar el campo de usuario y contraseña.3. El invitado introduce los datos.4. El sistema valida los datos y si son correctos crea el usuario en la base de datos. |
| Postcondiciones: La cuenta queda registrada en la base de datos y el invitado podrá iniciar sesión. |
| Alternativa 1: Los datos introducidos no cumplen la validación y se muestra un error. |

| |
|--|
| Nombre: LOGIN |
| Descripción: Inicio de sesión en el sistema. |
| Actores: Invitado. |
| Precondiciones: El invitado debe haberse registrado anteriormente. |
| Curso normal: <ol style="list-style-type: none">1. Escoge opción de Iniciar Sesión.2. El sistema solicita rellenar el campo de usuario y contraseña.3. El invitado introduce los datos.4. El sistema valida los datos.5. El sistema identifica al usuario y crea una sesión. |
| Postcondiciones: El invitado pasa a ser Usuario y tendrá acceso a nuevas opciones. |
| Alternativa 1: Los datos introducidos no coinciden con ningún usuario de la base de datos y se mostrará un error. |

| |
|---|
| Nombre: LOGOUT |
| Descripción: Cierre de sesión en el sistema. |
| Actores: Usuario, <i>Admin</i> . |
| Precondiciones: El actor debe haber iniciado sesión o no podrá ver la opción de cerrar sesión. |
| Curso normal: <ol style="list-style-type: none">1. Escoge opción de Cerrar Sesión.2. El sistema cierra la sesión del usuario.3. El invitado es redirigido a la url anterior. |
| Postcondiciones: El usuario o <i>admin</i> pasa a ser invitado. |
| Alternativa 1: |

| |
|--|
| Nombre: VISUALIZAR PLANO |
| Descripción: Se muestra el plano del aparcamiento con las plazas libres de color verde, y las no disponibles de color rojo. |
| Actores: Invitado, Usuario, <i>Admin</i> . |
| Precondiciones: Clickear en la opción de Plazas. El mapa debe estar configurado en el css para crear el plano correctamente. |
| Curso normal: <ol style="list-style-type: none">1. Escoge opción de Plazas.2. El sistema comprueba qué plazas están disponibles y cuáles no.3. Asignará la clase de css correspondiente para colocarle el background color correspondiente a cada Plaza.4. El sistema mostrará la vista de las plazas. |
| Postcondiciones: El administrador podrá tener acceso directo a la modificación de un cliente en las plazas ocupadas, o a la creación de un nuevo cliente en las plazas disponibles. El usuario podrá ver a que cliente pertenece las plazas ocupadas. |
| Alternativa 1: El plano no está bien configurado en el css y no se muestra correctamente. |

| |
|---|
| Nombre: FORMULARIO DE CONTACTO |
| Descripción: Se muestra un formulario para ponerte en contacto con el <i>admin</i> del sistema. |
| Actores: Invitado, Usuario, <i>Admin</i> . |
| Precondiciones: Clickear en la opción de Contacto. |
| Curso normal: <ol style="list-style-type: none">1. Escoge opción de Contacto.2. El sistema muestra un formulario con varios campos.3. El usuario rellena los campos.4. El sistema comprobará que se han rellenado todos los campos correctamente.5. El sistema enviará el mensaje al <i>admin</i> del sistema. |
| Postcondiciones: El administrador recibirá el mensaje. |
| Alternativa 1: El sistema comprueba que hay algún campo en blanco y mostrará un error. |

| |
|---|
| Nombre: VISUALIZAR CLIENTES |
| Descripción: Se muestra la lista de todos los clientes del aparcamiento. |
| Actores: Usuario, <i>Admin</i> . |
| Precondiciones: Debe haber iniciado sesión en el sistema. Clicar en la opción de Clientes. |
| Curso normal: <ol style="list-style-type: none">1. Escoge opción de Clientes.2. El sistema comprueba que el usuario haya iniciado sesión.3. El sistema accede a la base de datos de clientes.4. El sistema mostrará en una lista todos los clientes del aparcamiento. |
| Postcondiciones: El administrador tendrá acceso directo a la modificación o eliminación de un cliente desde la lista. El usuario y el <i>admin</i> podrá ver los datos de cada cliente cliqueando sobre él. El usuario tendrá restringidos algunos datos por protección de datos. |
| Alternativa 1: El sistema comprueba que no ha iniciado sesión y mostrará un error que redirige al Inicio de Sesión. |

| |
|--|
| Nombre: BUSCAR CLIENTES |
| Descripción: Se muestran dos formularios para realizar la búsqueda por nombre o por apellido. |
| Actores: Usuario, <i>Admin</i> . |
| Precondiciones: Debe haber iniciado sesión en el sistema. Clicar en la opción de Buscar Clientes. |
| Curso normal: <ol style="list-style-type: none">1. Escoge opción de Buscar Clientes.2. El sistema comprueba que el usuario haya iniciado sesión.3. El sistema mostrará dos formularios, uno para buscar por nombre y otro para buscar por apellido.4. El usuario introduce los datos en el campo que quiera.5. El sistema realiza la búsqueda en la base de datos de clientes.6. Mostrará los clientes que coincidan con la búsqueda en una lista. |
| Postcondiciones: El administrador tendrá acceso directo a la modificación o eliminación de un cliente desde la lista. |
| Alternativa 1: El sistema comprueba que no ha iniciado sesión y mostrará un error y te redirige al Inicio de Sesión. Alternativa 2: El sistema comprueba los datos, pero si no coinciden con ningún cliente mostrará el siguiente mensaje: "Clientes no encontrados ☹". |

| |
|---|
| Nombre: CREAR CLIENTE |
| Descripción: El <i>admin</i> tiene accesos directos en la aplicación sin tener que acceder al panel de control en la barra de navegación y en plano del aparcamiento si la plaza está disponible. |
| Actores: <i>Admin</i> . |
| Precondiciones: Debe haber iniciado sesión en el sistema como <i>admin</i> , de lo contrario no verá los accesos directos. |
| Curso normal: <ol style="list-style-type: none">1. Escoge la opción de Crear Cliente.2. El sistema mostrará un formulario con los atributos de un Cliente.3. E <i>admin</i> rellenará los datos del formulario.4. El sistema validará los datos y mostrará un mensaje de éxito.5. El sistema te redirige al Listado de Clientes. |
| Postcondiciones: Se modifica la tabla de Clientes de la base de datos. |
| Alternativa 1: El sistema comprueba que haya algún error en los campos del formulario y mostrará un error. |

| |
|---|
| Nombre: MODIFICAR CLIENTE |
| Descripción: El admin tiene accesos directos en la aplicación sin tener que acceder al panel de control en la lista de clientes y en los detalles de éstos. |
| Actores: Admin. |
| Precondiciones: Debe haber iniciado sesión en el sistema como admin, de lo contrario no verá los accesos directos. |
| Curso normal: <ol style="list-style-type: none">1. Selecciona un cliente de la lista de Clientes.2. Escoge la opción de Modificar Cliente.3. El sistema mostrará un formulario con los datos del Cliente.4. E admin modificará los datos del formulario.5. El sistema validará los datos y mostrará un mensaje de éxito.6. El sistema te redirige al Listado de Clientes. |
| Postcondiciones: Se modifica la tabla de Clientes de la base de datos. |
| Alternativa 1: El sistema comprueba que hay algún error en los campos del formulario y mostrará un error. |

| |
|--|
| Nombre: ELIMINAR CLIENTE |
| Descripción: El admin tiene accesos directos en la aplicación sin tener que acceder al panel de control en la lista de Clientes. |
| Actores: Admin. |
| Precondiciones: Debe haber iniciado sesión en el sistema como admin, de lo contrario no verá los accesos directos. |
| Curso normal: <ol style="list-style-type: none">1. Escoge la opción de Clientes.2. El sistema mostrará una lista con todos los clientes.3. E admin seleccionará la opción de Eliminar Cliente.4. El sistema mostrará un mensaje de confirmación.5. El admin aceptará la confirmación de Eliminar Cliente.6. El sistema mostrará un mensaje de éxito.7. El sistema te redirige al Listado de Clientes. |
| Postcondiciones: Se modifica la tabla de Clientes de la base de datos. |
| Alternativa 1: El admin no confirma la eliminación y el cliente no es eliminado. |

| |
|--|
| Nombre: PANEL DE CONTROL |
| Descripción: Se muestra el panel de control desde donde el admin podrá editar las tablas de la base de datos. |
| Actores: Admin. |
| Precondiciones: Debe haber iniciado sesión en el sistema como admin. Añadir /admin a la url |
| Curso normal: <ol style="list-style-type: none">1. Introduce /admin en la url del sistema.2. El sistema comprueba que el usuario haya iniciado sesión como admin.3. El sistema accede al panel de control de la base de datos del sistema.4. El sistema mostrará todas las tablas de la base de datos y sus opciones de edición. |
| Postcondiciones: El administrador podrá editar todas las tablas (Usuarios, Plazas, Dimensiones, Clientes), creando, modificando o eliminando tuplas de dichas tablas. |
| Alternativa 1: El sistema comprueba que el usuario no es el admin, denegará el acceso y pedirá un hacer <i>login</i> con una cuenta admin. |

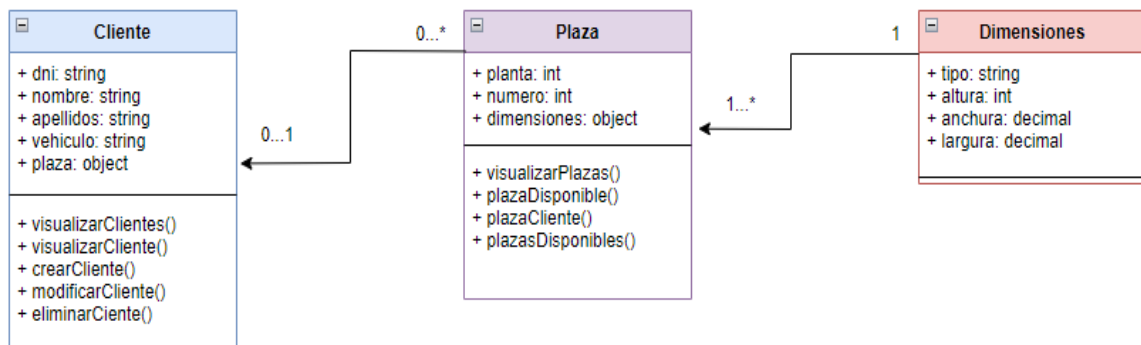
3.2. DIAGRAMA DE CLASES

A continuación se muestra un diagrama de clases que refleja las relaciones entre las clases, así como los atributos y métodos de éstos. Por tanto, vamos a estar trabajando sobre los siguientes modelos: Cliente y Plaza, ya que Dimensiones no tiene métodos.

En la clase Cliente tenemos el método visualizarClientes que mostrará un listado de todos los objetos Cliente, y el método visualizarCliente, mostrará los datos de un cliente que seleccionemos en particular.

El método crearCliente, modificarCliente y eliminarCliente, como su nombre indica, creará un objeto Cliente, editará sus atributos o lo eliminará.

En la clase Plaza, el método visualizarPlazas recorre todas las plazas y les asigna una clase de estilos, ubicadas en el css de la aplicación, que la ubicará en un grid container creando así el plano del aparcamiento. El método plazaDisponible, se encarga de comprobar si existe un cliente con esa plaza asignada, en cuyo caso devuelve *false*, y se pinta de color rojo en el plano, o *true* si está libre, y se pintará de color verde. El método plazaCliente devuelve el nombre del cliente cuyo atributo de plaza sea dicha plaza. Finalmente, el método plazasDisponibles, devuelve un *array* con sólo las plazas que están disponibles, para facilitar al admin la asignación de plaza a un cliente.



3.3. DIAGRAMA DE ENTIDAD RELACIÓN

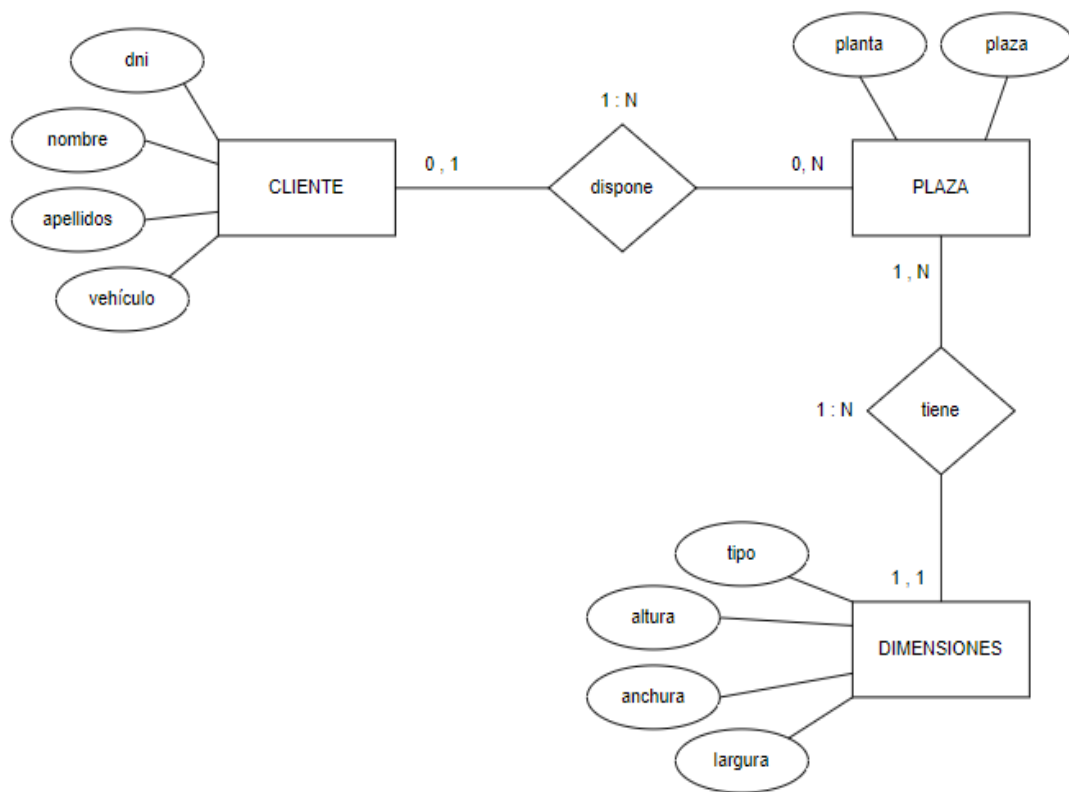
Comenzaré hablando de las relaciones entre Plaza y sus Dimensiones y terminaré con Plaza y Cliente.

Para que una Dimensión exista debe haber como mínimo una plaza con esas dimensiones, pero podrá haber más de una plaza con las mismas dimensiones.

Cada Plaza deberá tener una dimensión y sólo una.

Una Plaza podrá o no tener un Cliente, pero si tiene Cliente, solo podrá estar asignada a un único Cliente.

Un Cliente podrá tener varias Plazas o no tener ninguna.



3.4. MAPA DE NAVEGACIÓN

A continuación se presenta el mapa de navegación de la aplicación web, el cual sigue una estructura jerárquica, de tal forma que existe una página índice desde donde se accede al resto de páginas. Desde estas subpáginas se pueden acceder a otras y así sucesivamente, creando así distintos niveles.

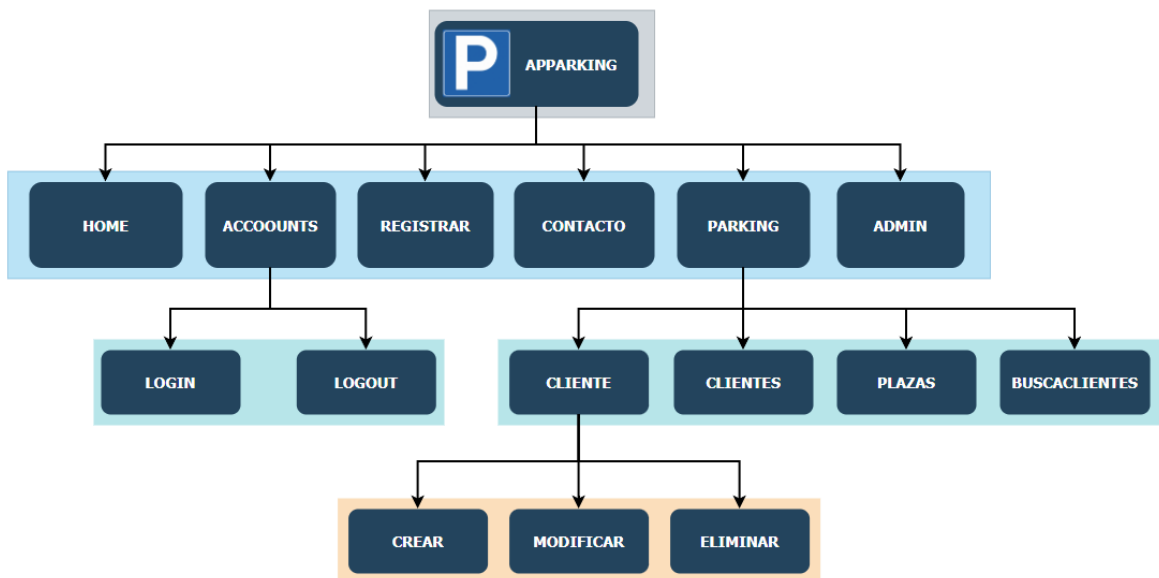
En primer lugar, el usuario es dirigido a la página de inicio o *Home*, donde podrá informarse acerca de los servicios que ofrece el aparcamiento, plazas disponibles y la ubicación.

En el segundo elemento es donde encontraremos parte de la interacción de la aplicación, se trata de un *Navbar* donde se mostrará un listado de las diferentes opciones de la aplicación, como son el plano de las plazas o la lista de clientes, entre otros.

En el apartado de contacto, ofrecemos la posibilidad a los usuarios de ponerse en contacto con nosotros a través de un formulario.

Finalmente tenemos los accesos a *login* y registrar, donde el admin podrá tener acceso al panel de control.

Dicho panel de control da acceso a la base de datos de la aplicación, donde se podrá editar las clases que intervienen en la aplicación (Usuarios, Plazas, Clientes y Dimensiones).



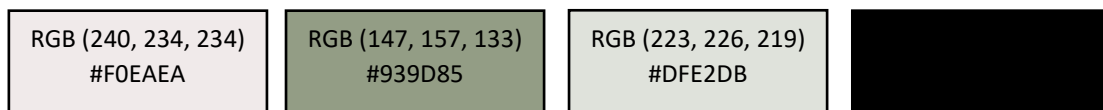
3.5. GUÍA BÁSICA DE ESTILO

Apparking cuenta con un diseño totalmente *responsive* con adaptabilidad a dispositivos tales como móviles, tablets, portátiles y pantallas grandes, ofreciendo una interfaz amigable y accesible a todo tipo de usuarios.

Cuenta con una base.html que tendrá todos los elementos que deben aparecer en las demás páginas como son el *Navbar*, el *footer*, el panel del *content* y la imagen de fondo, de la cual se extienden el resto de páginas.

Gama cromática

Los colores corporativos de Apparking son tres grises diferenciados para dar contraste y recordar a los cimientos de un aparcamiento, los cuales se combinan en distintos elementos de la aplicación.



Para el plano del aparcamiento y comprobar su disponibilidad usaremos:



Fuente

Se ha escogido la fuente “Monsterrat”, mediante la cual se trata de transmitir limpieza y modernidad al usuario.

Elementos interactivos

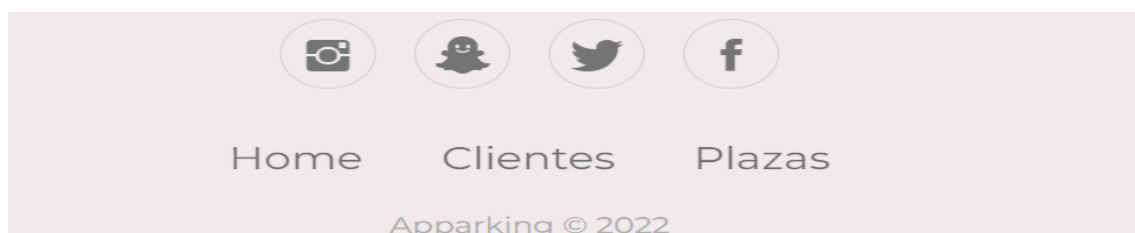
Navbar:

Mediante los elementos del Navbar de la vista del cliente, se permite al usuario la navegación entre las distintas páginas que componen la aplicación.



Footer:

Contiene los iconos de las redes sociales Instagram, Snapchat, Twitter y Facebook que redirigirán a dichas páginas y varios elementos de navegación.



Panel del content:

En todas las páginas de la aplicación, el contenido de cada página es generado por una vista y mostrado en una *template*, ambas configuradas por el desarrollador. Este contenido se mostrará en un panel que cambiará de tamaño si el contenido lo necesita.

Por favor, inicia sesión para ver esta pagina.

Nombre de usuario:

Contraseña:

[Lost password?](#)

Apparking

Visitas: 1

Plazas disponibles: 10

Con Apparking, gestionarás el parking de tu comunidad de vecinos con un simple Click.
Fácil y rápido de utilizar, comprobarás en segundos las plazas disponibles para los nuevos vecinos.
Apparking está configurado personalmente para ti, ajustamos el plano a tus necesidades.


Accesos y salidas
Lunes a Domingo.....24h

Vehiculo
Altura máxima..... 3 metros

Servicios del Aparcamiento

- Acceso para minusválidos
- Cámaras de Seguridad
- Ascensor

Donde estamos:



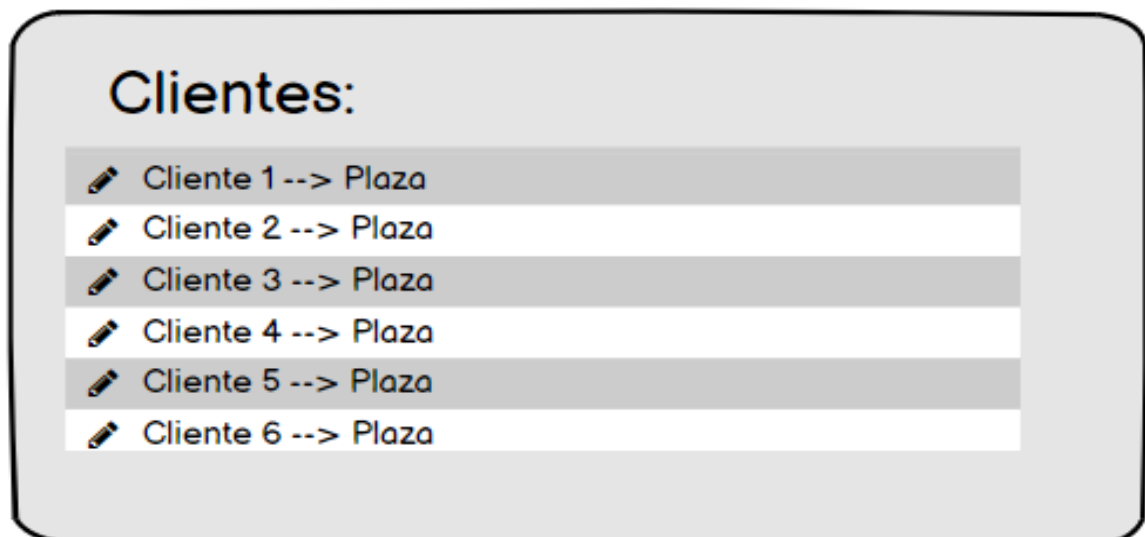
3.6. MOCKUPS

Base:



El resto de *mockups* serán solo sobre el interior del Panel del *content*, puesto que todas heredan de la base tanto el navbar como el footer y el fondo.

Clientes:



Crear Clientes:

Crear Cliente :

DNI

Nombre

Apellidos

Vehiculo

Plaza

▼

Crear



Búsqueda Clientes:

Búsqueda de Clientes:

Buscar

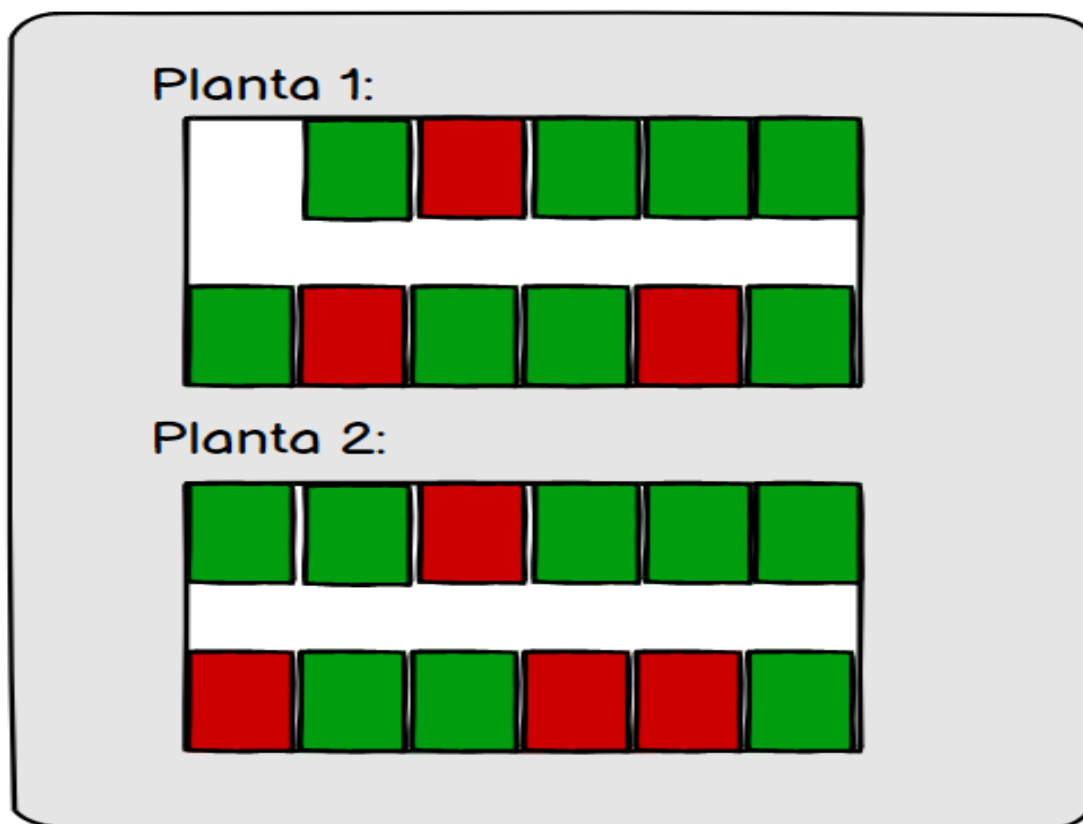
Resultados :

Cliente1 --> Plaza

Cliente2 --> Plaza



Plazas:



Contacto:

Formulario de Contacto

Nombre:

Email:

Asunto:

Mensaje:

4. IMPLEMENTACIÓN Y PRUEBAS

4.1. IMPLEMENTACIÓN

Visual Studio Code

Python 3.10

SQLite

Oracle VM Virtual Box

Nginx 1.21

Gunicorn 20.1.0

Django Framework 3.2.9

Django-debug-toolbar 3.2.2

Jquery 3.6

Bootstrap 5.1

Ajax

Leaflet 1.7.1

Git 2.33.1

Vagrant 2.2.19

Bibliografía:

<https://docs.djangoproject.com/en/4.0>

<https://leafletjs.com/reference.html>

<https://getbootstrap.com/docs/5.1>

<https://ellibrodepython.com/>

<https://www.vagrantup.com/docs>

<https://gunicorn.org/#docs>

Github:

https://github.com/SergioSoteras/proyecto_apparking

4.2. PRUEBAS

4.2.1. Pruebas integradas

Django permite realizar test integrados y he realizado una serie de pruebas sobre la creación de las clases o modelos, para comprobar el correcto funcionamiento del sistema. Están ubicados en la carpeta *test* de la aplicación parking. Para ejecutarlos bastará con ejecutar el comando “py manage.py test”.

Las tres clases que veremos serán:

Cliente:

```
#Cliente creado correctamente
class ClienteTestCase(TestCase):

    @classmethod
    def setUpTestData(cls):
        Dimensiones.objects.create(tipo='prueba', altura=2, anchura=2, largura=2)
        dimPrueba = Dimensiones.objects.get(tipo='prueba')
        Plaza.objects.create(planta=1, numero=1, dimensiones=dimPrueba)
        plaza1= Plaza.objects.get(numero=1)
        Cliente.objects.create(dni='73012345H', nombre='Sergio', apellidos='Soteras Serrano', vehiculo='C4', plaza = plaza1)
        Cliente.objects.create(dni='73543210H', nombre='Robustiano', apellidos='Rios Gil', vehiculo='Focus',)

    def test_str_cliente(self):
        cliente = Cliente.objects.get(dni='73012345H')
        self.assertEqual(cliente.__str__(), 'Sergio Soteras Serrano')
        self.assertEqual(cliente.vehiculo, 'C4')

    def test_cliente_plaza(self):
        cliente = Cliente.objects.get(dni='73012345H')
        self.assertEqual(cliente.plaza, Plaza.objects.get(numero=1))

#Cliente modificado correctamente
    def test_modificar_cliente(self):
        cliente = Cliente.objects.get(dni='73012345H')
        cliente.nombre = 'Pedro'
        cliente.save()
        self.assertEqual(Cliente.objects.get(dni='73012345H').nombre, 'Pedro')

#Cliente eliminado correctamente
    def test_eliminar_cliente(self):
        cliente = Cliente.objects.get(dni='73012345H')
        cliente.delete()
        self.assertEqual(Cliente.objects.first().nombre, 'Robustiano')
```

Plaza:

```
#Plaza creada correctamente
class PlazaTestCase(TestCase):

    @classmethod
    def setUpTestData(cls):
        Dimensiones.objects.create(tipo='prueba', altura=2, anchura=2, largura=2)
        dimPrueba = Dimensiones.objects.get(tipo='prueba')
        Plaza.objects.create(planta=1, numero=1, dimensiones=dimPrueba)
        plaza1= Plaza.objects.get(numero=1)
        Cliente.objects.create(dni='73012345H', nombre='Sergio', apellidos='Soteras Serrano', vehiculo='C4',)

    def test_plaza_creada_correctamente(self):
        plaza1= Plaza.objects.get(numero=1)
        self.assertEqual(plaza1.planta,1)

    def test_plaza_dimensiones(self):
        plaza1=Plaza.objects.get(numero=1)
        self.assertEqual(plaza1.dimensiones,Dimensiones.objects.get(tipo='prueba'))

    def test_str_plaza(self):
        plaza1=Plaza.objects.get(numero=1)
        self.assertEqual(plaza1.__str__(), 'Plaza 1 ( Planta 1 )')

#Plaza disponible
    def test_plaza_disponible(self):
        plaza1= Plaza.objects.get(numero=1)
        self.assertTrue(plaza1.disponible())
        cliente = Cliente.objects.get(dni='73012345H')
        cliente.plaza = plaza1
        cliente.save()
        self.assertFalse(plaza1.disponible())

#Cliente de la plaza
    def test_plaza_cliente(self):
        plaza1= Plaza.objects.get(numero=1)
        cliente = Cliente.objects.get(dni='73012345H')
        cliente.plaza = plaza1
        cliente.save()
        self.assertEqual(plaza1.nombre_cliente(), cliente)
```

Dimensiones:

```
#Dimensiones creada correctamente
class DimensionesTestCase(TestCase):

    @classmethod
    def setUpTestData(cls):
        Dimensiones.objects.create(tipo='prueba', altura=2, anchura=2, largura=2)

    def test_dimensiones_creada(self):
        dim = Dimensiones.objects.get(tipo='prueba')
        self.assertEqual(dim.altura,2)
        self.assertEqual(dim.anchura,2)
        self.assertEqual(dim.largura,2)

    def test_str_dimensiones(self):
        dim = Dimensiones.objects.get(tipo='prueba')
        self.assertEqual(dim.__str__(), 'prueba (2.00m x 2.00m x 2m)')
```

Son pruebas sencillas que Django hace por nosotros gracias a una clase llamada TestCase, usa una tabla creada solo para los test, que eliminará los datos después de cada prueba. El setUpTestData sirve para ejecutar ese código justo antes de cada prueba, por eso creo el objeto, para no tener que crearlo en cada prueba.

La mayoría de pruebas son para comprobar que efectivamente el objeto se crea con los atributos correctos, usando el `assertEqual` de un atributo del objeto con el que debería ser. El resto es el resultado de las funciones que he desarrollado en cada modelo, comparándolo con el `assert` adecuado para cada resultado, `assertEqual` para strings, `assertTrue` o `assertFalse` para booleanos...

Como ya he comentado anteriormente, para realizar los test introducimos en la consola el comando “`py manage.py test`” que realizará todos los test que encuentre y mostrará por consola si todo OK, o si ha habido algún problema y en qué test.

Ejecución:

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL

PS C:\Users\sergi\proyectos\apparcagz> py manage.py test
Creating test database for alias 'default'...
System check identified some issues:

-----
Ran 11 tests in 0.019s

OK
Destroying test database for alias 'default'...
PS C:\Users\sergi\proyectos\apparcagz> |
```

4.2.2. Pruebas en la app

Registro de Usuario:

Validación de nombre de usuario



Registro de Usuario

Nombre de usuario:

Requerido. 150 caracteres como máximo. Únicamente letras, dígitos y @./+/_/

Introduza un nombre de usuario válido. Este valor puede contener únicamente letras, números y los caracteres @./+/_/

Validación de Usuario existente



Registro de Usuario

Nombre de usuario:

Requerido. 150 caracteres como máximo. Únicamente letras, dígitos y @./+/_/

Ya existe un usuario con este nombre.

Validación de contraseña sencilla

Contraseña (confirmación):

Para verificar, introduzca la misma contraseña anterior.

Esta contraseña es demasiado corta. Debe contener al menos 8 caracteres.

Contraseña (confirmación):

Para verificar, introduzca la misma contraseña anterior.

La contraseña es demasiado similar a la de nombre de usuario.

Contraseña (confirmación):

Para verificar, introduzca la misma contraseña anterior.

Esta contraseña es demasiado común.

Esta contraseña es completamente numérica.

Validación de Repetir contraseña correcta

Contraseña:

- Su contraseña no puede asemejarse tanto a su otra información personal.
- Su contraseña debe contener al menos 8 caracteres.
- Su contraseña no puede ser una clave utilizada comúnmente.
- Su contraseña no puede ser completamente numérica.

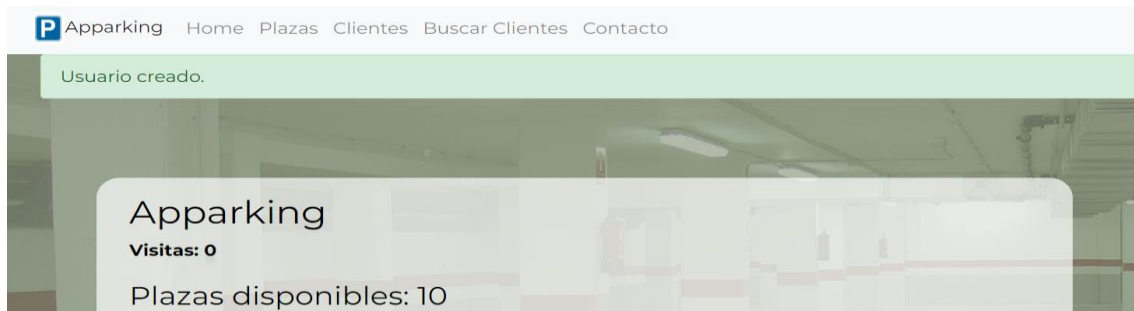
Contraseña (confirmación):

Para verificar, introduzca la misma contraseña anterior.

Los dos campos de contraseña no coinciden.

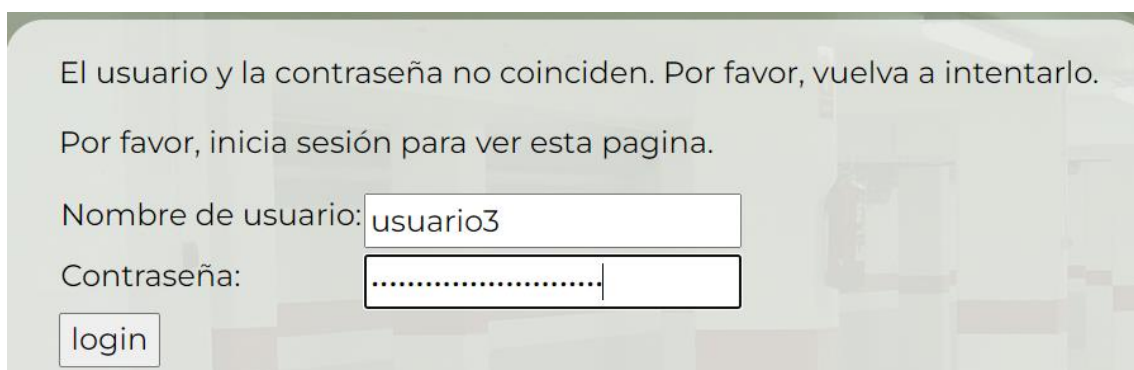
Mensaje de éxito si el Registro es correcto y redirección al Home.

Para futuras pruebas, el usuario es usuario3 y contraseña3.

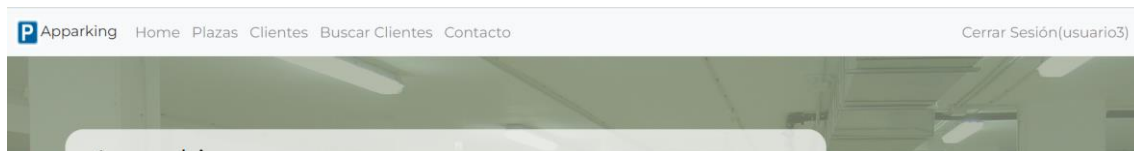


Login:

Login incorrecto

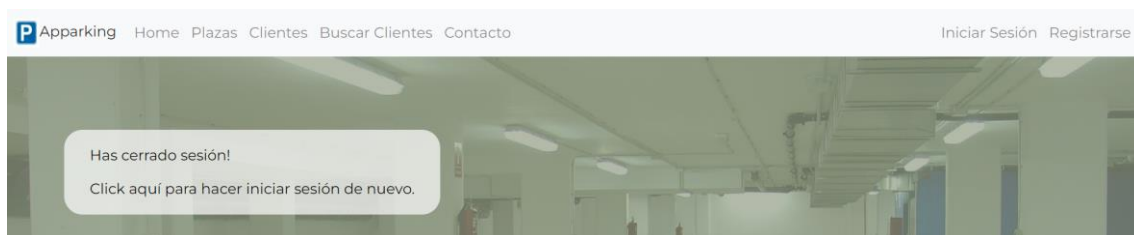


Login correcto



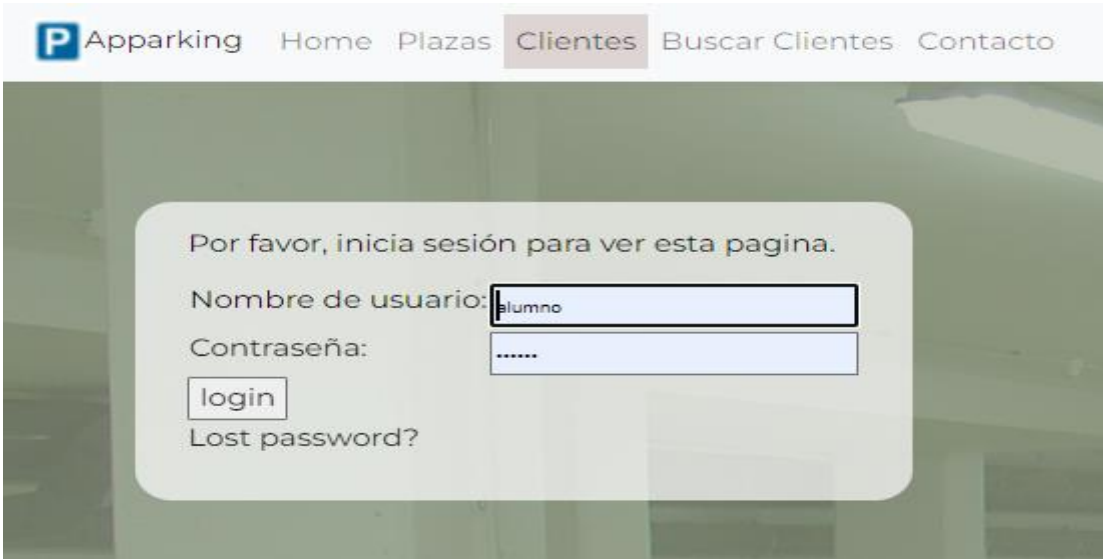
Se puede observar que, en el Navbar, ha desaparecido Iniciar Sesión / Registrarse por Cerrar Sesión (nombre de usuario)

Logout:

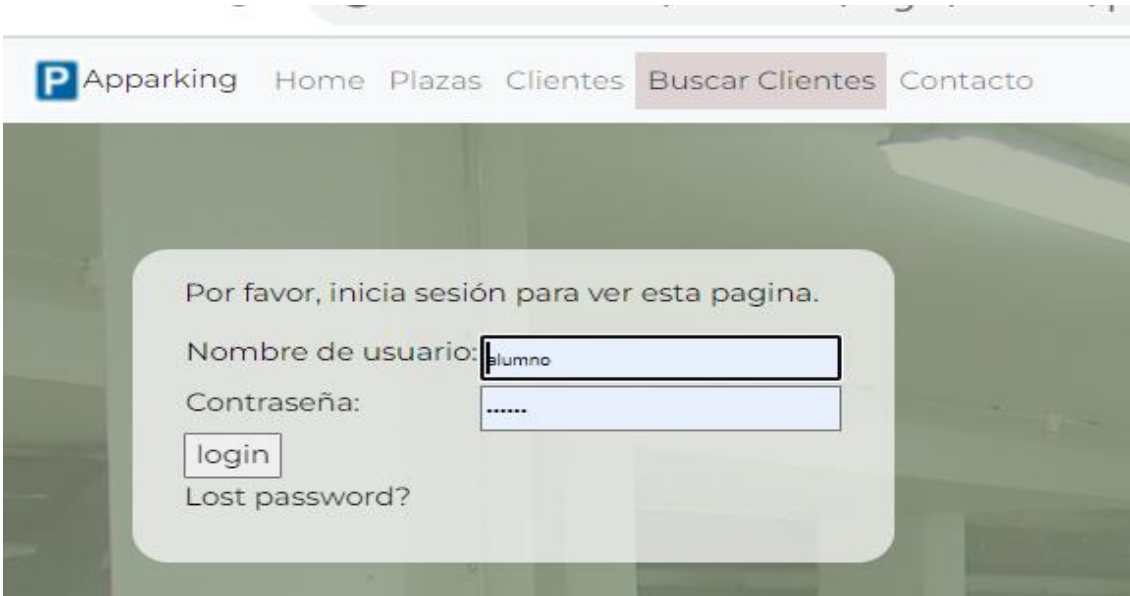


LoginRequired:

Tanto ver Clientes como Buscar Clientes requieren haber iniciado sesión



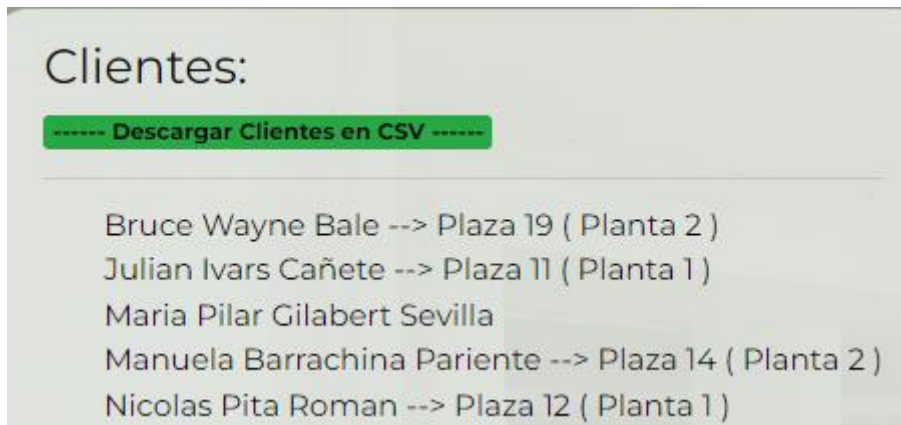
The screenshot shows the Apparking website with a navigation bar containing links: Apparking, Home, Plazas, Clientes, Buscar Clientes, and Contacto. The 'Clientes' link is highlighted. A modal dialog is displayed in the center with the text: 'Por favor, inicia sesión para ver esta pagina.' Below this text are two input fields: 'Nombre de usuario:' with the value 'alumno' and 'Contraseña:' with masked characters '.....'. There is a 'login' button and a link for 'Lost password?'.



This screenshot is identical to the one above, showing the Apparking website with the 'Buscar Clientes' link highlighted in the navigation bar. The same login modal is displayed, asking the user to log in to view the page, with the username 'alumno' and a masked password.

Clientes:

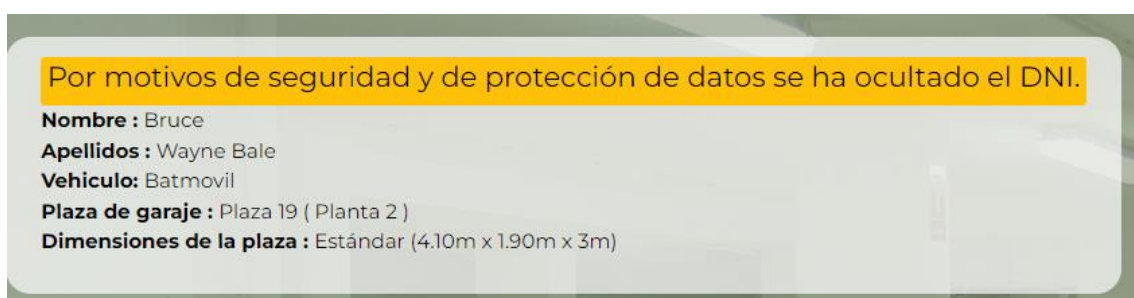
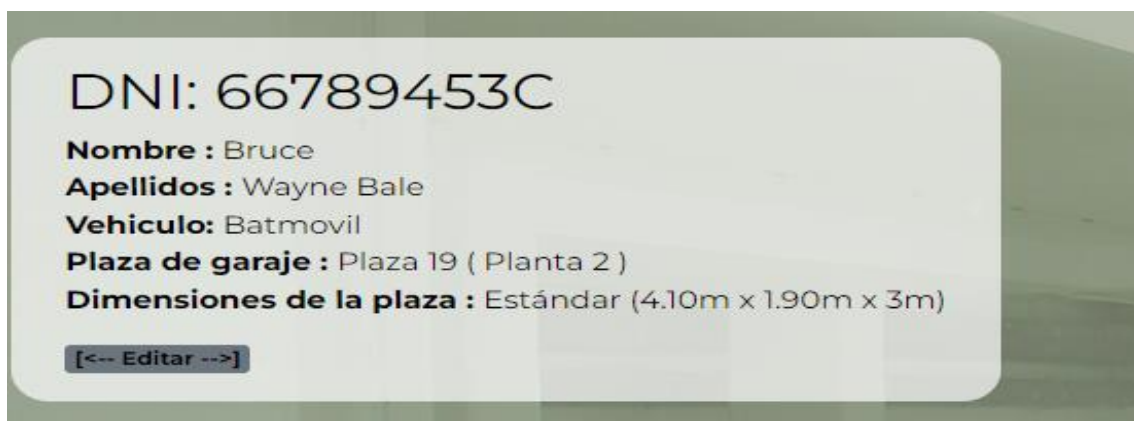
Los clientes sin plaza deberán verse de distinta manera que los que tienen plaza



El admin tiene accesos directos para edición que los usuarios no pueden ver



Datos del Cliente – Usuario vs *Admin*



Buscar Clientes:

Búsqueda por nombre con resultados

Búsqueda de Clientes

Búsqueda por apellido:

Búsqueda por nombre:

- Juan Manuel Rovira Lago -- Renault Megane
- Manuela Barrachina Pariente -- Peugeot 308



Búsqueda por nombre sin resultados

Búsqueda de Clientes

Búsqueda por apellido:

Búsqueda por nombre:

Clientes no encontrados :(



Búsqueda por apellidos con resultados

Búsqueda de Clientes

Búsqueda por apellido:

Búsqueda por nombre:

- Bruce Wayne Bale -- Batmovil




Búsqueda por apellido sin resultados

Búsqueda de Clientes

Búsqueda por apellido:

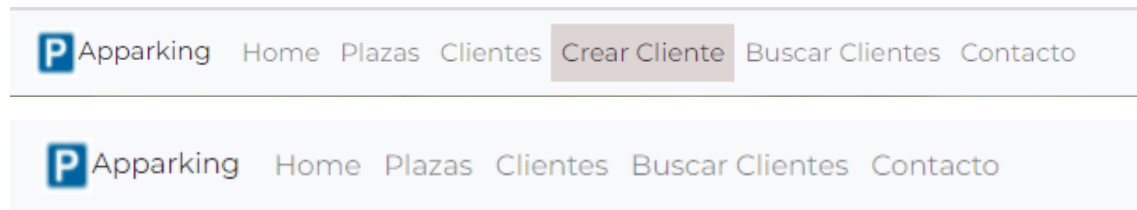
Búsqueda por nombre:

Clientes no encontrados :(

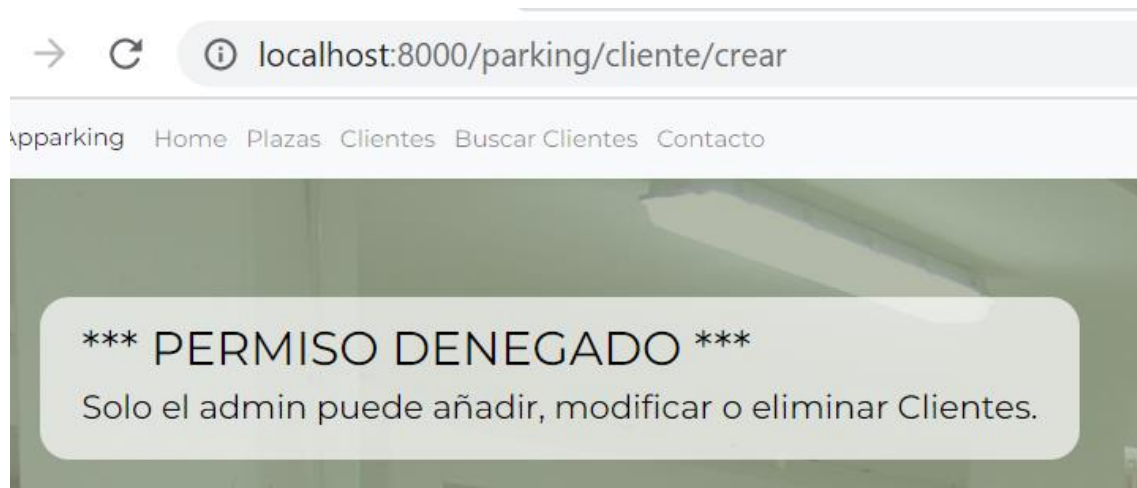


Crear Cliente:

El admin tiene un acceso directo en el Navbar que el usuario no puede ver



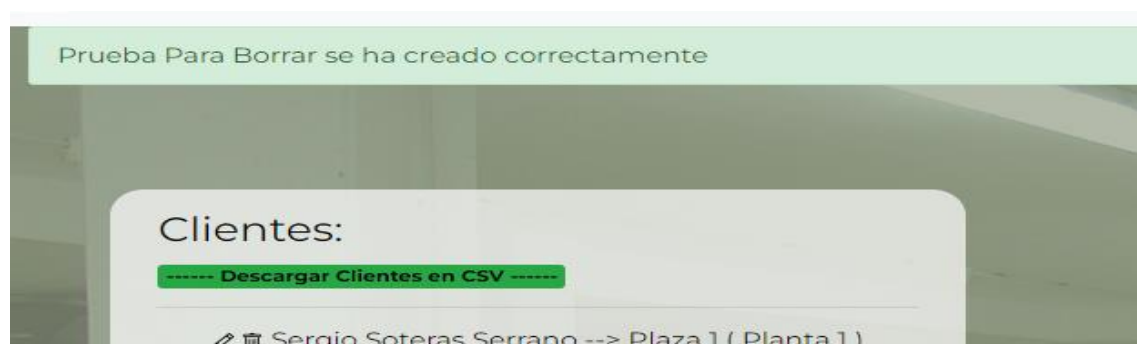
Si el usuario introduce la url, se mostrará un aviso de que no tiene acceso

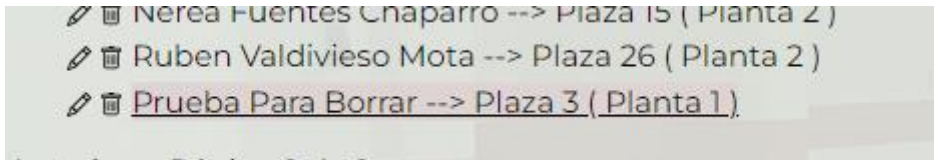


Validación del DNI existente y de Plaza Disponible



Mensaje de éxito al crear cliente

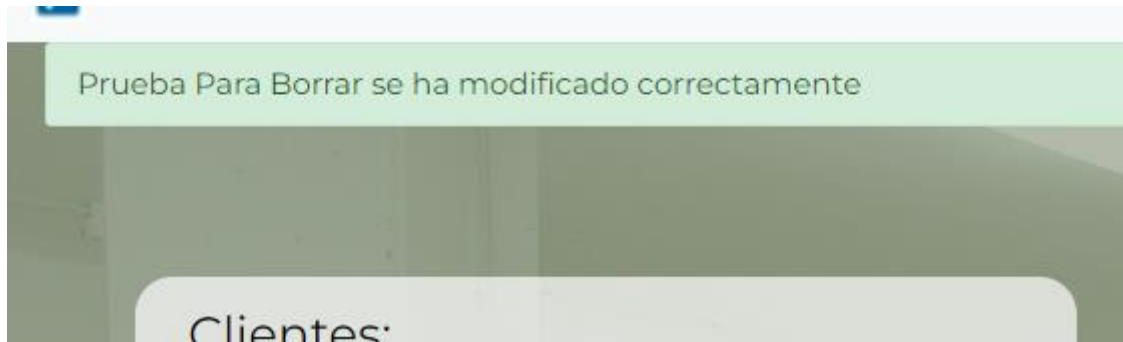




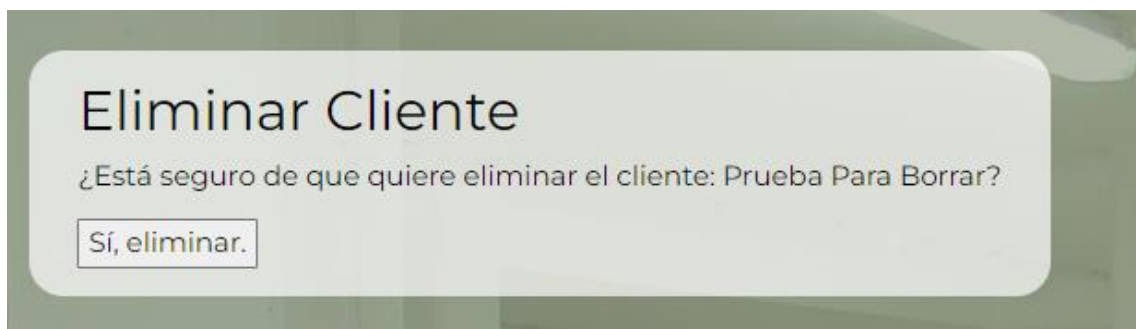
Modificar Cliente:

Utiliza el mismo formulario que Crear Cliente, asique las validaciones son las mismas.

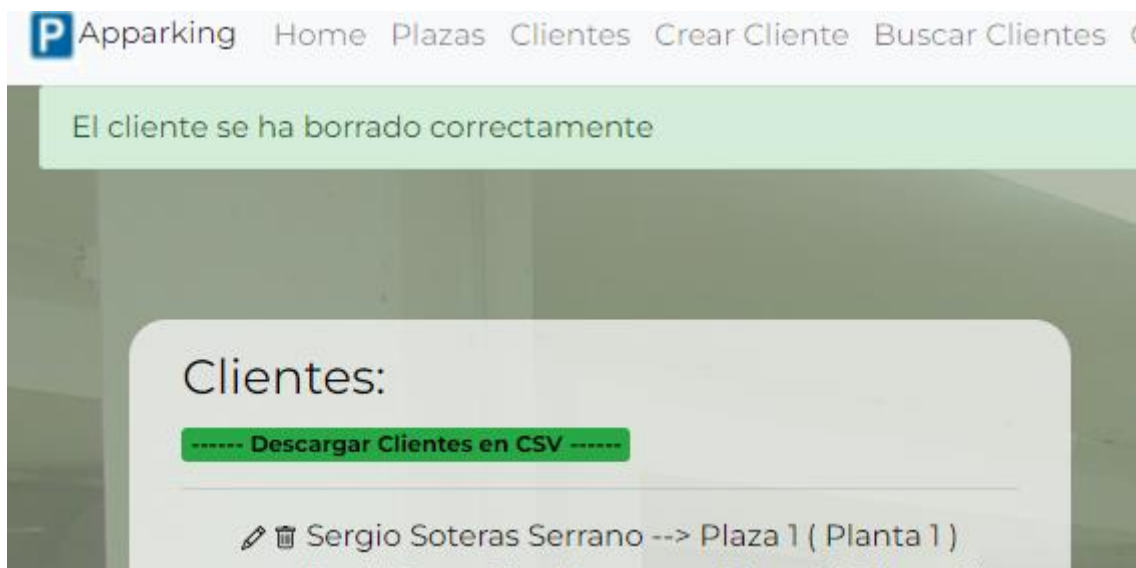
Mensaje de éxito



Eliminar Cliente:



Mensaje de éxito



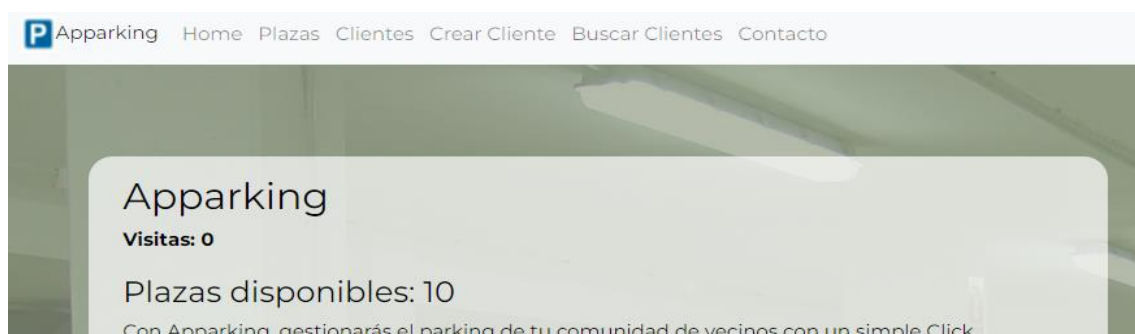
Plazas:

Visualizar el plano de las plazas correctamente, pintando de verde las disponibles y de rojo las ocupadas. Si echamos un ojo a la lista de clientes, las únicas plazas que no tienen cliente y deberán pintarse de verde son la 3, 5, 8, 9, 17, 20, 22, 23, 25 y 27.



Contador del Home de plazas disponibles:

Con una función, identifico cuantas plazas no tienen cliente y lo muestro, en este caso, fijándonos en el plano de arriba, hay 10 disponibles.



5. DOCUMENTACIÓN

5.1. MANUAL DE INSTALACIÓN

1. Instalación de Python

Descargaremos la versión Python 3.10 o superior

<https://www.python.org/downloads>

2. Creamos un Entorno Virtual

- Linux:
`sudo apt-get install python3-venv`
`python3 -m venv env`
- macOS:
`python3 -m venv env`
- Windows:
`python -m venv env`

3. Activar el Entorno Virtual

- Linux / Mac:
`source env/bin/activate`
- Windows:
`env\Scripts\activate.bat`

4. Instalar Django y dependencias

```
pip install Django  
pip install -r requirements.txt
```

5. Crear *superuser*

```
py manage.py createsuperuser  
Es importante que el username sea admin
```

6. `py manage.py runserver`

5.2. MANUAL DE USUARIO






Este manual va dirigido al admin y trata sobre el manejo del panel de control, al que se accede añadiendo “/admin” a la url de la aplicación, debido a que la navegación por la aplicación es muy intuitiva y sencilla de usar y no hace falta manual.

Para acceder al panel de control, nos pedirá un usuario y contraseña si no hemos iniciado sesión aún y, por supuesto, solo el superuser admin tendrá acceso a él.

Lo primero que veremos serán las tablas de la aplicación divididas en 2 grupos, una para Autenticación y Autorización o Permisos, y otra para la gestión del Parking con Clientes, Plazas y Dimensiones.

Administración de Django

Sitio administrativo

| AUTENTICACIÓN Y AUTORIZACIÓN | | |
|------------------------------|----------|---|
| Grupos | + Añadir |  Modificar |
| Usuarios | + Añadir |  Modificar |
| PARKING | | |
| Clientes | + Añadir |  Modificar |
| Dimensiones | + Añadir |  Modificar |
| Plazas | + Añadir |  Modificar |

5.2.1. Grupos

En el panel de control de grupos, introduciremos el nombre del grupo que deseemos crear y observaremos dos columnas. La primera con los permisos disponibles y la segunda con los permisos elegidos para el grupo en creación.

Para añadir los permisos de una columna a la otra tenemos dos opciones: doble clic o seleccionar una y hacer clic en los botones flecha que se encuentran entre ambas columnas.

Añadir grupo

Nombre:

Permisos:

permisos Disponibles ?

- admin | entrada de registro | Can add log entry
- admin | entrada de registro | Can change log entry
- admin | entrada de registro | Can delete log entry
- admin | entrada de registro | Can view log entry
- auth | grupo | Can add group
- auth | grupo | Can change group
- auth | grupo | Can delete group
- auth | grupo | Can view group
- auth | permiso | Can add permission
- auth | permiso | Can change permission
- auth | permiso | Can delete permission
- auth | permiso | Can view permission
- auth | usuario | Can add user

Selecciona todos ?

permisos elegidos ?
Eliminar todos

Mantenga presionado "Control" o "Comando" en una Mac, para seleccionar más de uno.

5.2.2. Usuarios

Cuando un invitado se registre en el sistema, automáticamente se añadirá a la tabla de usuarios con la contraseña cifrada por seguridad. El admin puede crear usuarios desde el panel de control por si le interesa tener alguna cuenta más con acceso al panel de control o cualquier otro motivo.

Administración de Django
Inicio > Autenticación y autorización > Usuarios > Añadir usuario

AUTENTICACIÓN Y AUTORIZACIÓN
Grupos + Añadir
Usuarios + Añadir

PARKING
Clientes + Añadir
Dimensioness + Añadir
Plazas + Añadir

Añadir usuario

Primero, ingrese un nombre de usuario y contraseña. Luego, podrá editar más opciones del usuario.

Nombre de usuario:
Requerido. 150 caracteres como máximo. Únicamente letras, dígitos y @/./+/_/

Contraseña:
Su contraseña no puede asemejarse tanto a su otra información personal.
Su contraseña debe contener al menos 8 caracteres.
Su contraseña no puede ser una clave utilizada comúnmente.
Su contraseña no puede ser completamente numérica.

Contraseña (confirmación):
Para verificar, introduzca la misma contraseña anterior.

Para modificar un usuario, ya sea para darle permisos de *staff* o hacerle superuser, al seleccionar usuarios se mostrará una lista de todos los usuarios, haciendo clic sobre uno de ellos nos llevará a modificar dicho usuario, en el que se encuentran los siguientes apartados:

Información del Usuario

Modificar usuario

usuario2

Nombre de usuario:

usuario2

Requerido. 150 caracteres como máximo. Únicamente letras, dígitos y @/./+/-/_

Contraseña:

algoritmo: pbkdf2_sha256 **iteraciones:** 260000 **salto:** 8tx9V***** **función resumen:** 6ed+eZ*****

Las contraseñas no se almacenan en bruto, así que no hay manera de ver la contraseña del usuario, pero se puede cambiar la contraseña mediante este formulario.

Información personal

Nombre:

Nombre de user2

Apellidos:

Apellido de user2

Dirección de correo electrónico:

usuario2@hotmail.com

Permisos

Permisos

☒ Activo
Indica si el usuario debe ser tratado como activo. Desmarque esta opción en lugar de borrar la cuenta.

☐ Es staff
Indica si el usuario puede entrar en este sitio de administración.

☐ Estado de superusuario
Indica que este usuario tiene todos los permisos sin asignárselos explícitamente.

Grupos:

grupos Disponibles

Cientes del Parking

Selecciona todos

grupos elegidos

Eliminar todos

Permisos de usuario:

permisos de usuario Disponibles

admin | entrada de registro | Can add log entry

admin | entrada de registro | Can change log entry

admin | entrada de registro | Can delete log entry

admin | entrada de registro | Can view log entry

auth | grupo | Can add group

auth | grupo | Can change group

auth | grupo | Can delete group

auth | grupo | Can view group

auth | permiso | Can add permission

auth | permiso | Can change permission

auth | permiso | Can delete permission

auth | permiso | Can view permission

auth | usuario | Can add user

permisos de usuario elegidos

Eliminar todos

Esta es la parte más importante a la hora de modificar un Usuario. En la primera parte se selecciona el tipo de usuario (Activo, Staff o Superusuario), seguido de los grupos a los que pertenecerá con sus respectivos permisos y, finalmente, para añadirle permisos extras al usuario.

42

Fechas importantes

Fechas importantes

Último inicio de sesión:

Fecha: Hoy | 

Hora: Ahora | 

Nota: Usted va 2 horas por delante de la hora del servidor.

Fecha de alta:

Fecha: Hoy | 

Hora: Ahora | 

Nota: Usted va 2 horas por delante de la hora del servidor.

Proporciona información sobre la actividad del usuario y la fecha en la que se registró en el sistema.

5.2.3. Tablas de Parking

Puesto que todas las tablas de este apartado funcionan de la misma manera, se explicará solo una de ellas, pudiéndose aplicar al resto de tablas.

Se hará uso de la tabla Dimensiones para la tabla Plazas, en la que se pueden encontrar 3 tipos:

| <input type="checkbox"/> TIPO | ANCHURA | LARGURA |
|-----------------------------------|---------|---------|
| <input type="checkbox"/> Pequeña | 1,80 | 3,80 |
| <input type="checkbox"/> Grande | 2,50 | 4,80 |
| <input type="checkbox"/> Estándar | 1,90 | 4,10 |

Para añadir o modificar una Plaza, se hará clic en la plaza o en el botón de Añadir Plaza.

Seleccione plaza a modificar

AÑADIR PLAZA +

Acción: ▼

seleccionados 0 de 27

| <input type="checkbox"/> PLANTA | NUMERO |
|---------------------------------|--------|
| <input type="checkbox"/> 2 | 27 |
| <input type="checkbox"/> 2 | 26 |
| <input type="checkbox"/> 2 | 25 |

FILTRO

Por planta

Todo

1




2

Completar los campos, y en Dimensiones, seleccionar el tipo, que se habrán configurado anteriormente, y estarán conectadas por una *foreign key*.

Añadir plaza

Planta:




Numero:

Dimensiones:   

Pequeña (3.80m x 1.80m x 3m)

Estándar (4.10m x 1.90m x 3m)

Grande (4.80m x 2.50m x 3m)

 Guardar y continuar editando  Guardar y continuar editando  GUARDAR

6. CONCLUSIONES

Me satisface decir que Apparking es una aplicación que en su primera versión cumple con todas las funciones demandadas por el cliente que se encuentra satisfecho con el resultado.

Como punto negativo de esta versión, decir que sólo funciona de manera local, instalada en un ordenador público, al que tendrán acceso los miembros de la comunidad y el presidente, pero no podrán acceder desde el ordenador personal a través de una web con dominio.

En la segunda versión de la aplicación se procederá a desplegar la aplicación y el traspaso de la base de datos SQLite a PostgreSQL, así como elegir un *hosting* y dominio, y se valorará la posibilidad de poder reservar una plaza siendo Usuario.

En mi opinión, se trata de una aplicación con bastante visión de futuro, muy personal puesto que cada aparcamiento tiene un plano diferente debido a la disposición de las plazas, y ofrece al parking la posibilidad de negocio sin intermediarios como son *Parkapp* u *Onepark*, ya que se puede implementar una pasarela de pago que permita a los clientes pagar su reserva directamente desde la web.

El trabajo duro y la ayuda del profesorado ha resultado un factor clave para el desarrollo del proyecto y para el éxito de los resultados. Por eso mismo quiero agradecer a mi tutor Luis Miguel Morillas y a todo el departamento de informática por los conocimientos y el apoyo que me han brindado a lo largo de estos dos años.

Gracias.