SMART BUILDING BOOKING

Gestión de Comunidades

Autor: Nerea Rodríguez Martín

TUTOR: JOSÉ PIÑERO CENTRO: IES FRANCISCO AYALA, DPTO: DESARROLLO APLICACIONES WEB

ÍNDICE

1. RESUMEN DEL PROYECTO……...2

1.1. RESUMEN DEL PROYECTO EN INGLÉS……..2

2. ENUNCIADO DEL PROBLEMA…….3

3. TECNOLOGÍAS APLICADAS…….4

4. REQUERIMIENTOS HARDWARE Y SOFTWARE……...5

5. ANÁLISIS Y DISEÑO……...6

6. IMPLEMENTACIÓN…….7

7. RELACIÓN DETALLADA DE CADA UNO DE LOS FICHEROS QUE SE INCLUYEN……7

8. EVALUACIÓN Y PRUEBA……14

9. MANUAL DE USUARIO……15

10. GUÍA DE INSTALACIÓN…….21

11. SOFTWARE UTILIZADO…….21

12. MEJORAS POSIBLES Y APORTACIONES……21

1. RESUMEN DEL PROYECTO

Sobway (Smart Open Building), es un grupo de profesionales del sector de la tecnología de comunicaciones y automatización con amplia experiencia en Real Estate y Property Managment. Realizan un diagnóstico de las posibilidades de domotización que posee la edificación, y aportan la solución a medida y personalizada que convertirá el residencial en SmartBuilding. Han querido ampliar su campo de actuación añadiendo reservas a las instalaciones de los edificios, haciéndolos más eficientes. La aplicación web a rasgos generales, permite la gestión de dichas reservas, tanto de los propietarios como de personas ajenas a la comunidad haciendo una compensación de gastos.

En la página principal he incluido cuatro secciones en el menú, las cuales incluyen el *Inicio, Servicios, Nosotros, Clientes*, Además, también cuenta con un apartado de registro y de login. Una vez logeado, se tiene acceso a la parte privada donde se muestran todas las opciones de reserva que tiene ese usuario y puede acceder a las instalaciones con las que contamos, observar sus detalles, como su ubicación, galería de fotos, valoraciones, breve descripción… y de esta forma, realizar la reserva que considere oportuna, pudiendo elegir el número de pases, turno, etc.

La aplicación consta de varias interfaces principales, cuyo aspecto y funcionalidad dependerán del rol del usuario registrado. La aplicación se desarrolla en web, de forma que los usuarios podrán acceder a ella sin necesidad de instalar software en su equipo informático y con el único requisito de tener un navegador web y una conexión a internet. Algo que a día del desarrollo de esta aplicación está muy extendido en nuestro país.

La razón de la elaboración de este proyecto viene dada por la necesidad que tienen los propietarios de hacer su edificio eficiente, ya que con esta ampliación podrán compensar los gastos de comunidad ya que al contar con tantas instalaciones tienen un gasto mensual muy elevado. Pueden estar informados en todo momento y en tiempo real de los datos propios de las personas que hagan uso de las mismas, así como aforo, horario, etc.

Se han usado varias tecnologías: Angular y Nodejs, la base de datos será SQL y para todo lo que tenga que ver con usuarios se usará Cognito de AWS(AmazonWebService).

En cuanto a funcionalidad se busca conseguir un gestor del estilo de Airbnb o incluso BlaBlacar, etc. pero para instalaciones, por ejemplo, poder reservar una pista de tenis de una comunidad por determinado número de horas o una piscina en cierto tramo horario… En cuanto al diseño de las páginas, he utilizado MaterializeCSS.

Hay 2 perfiles de usuarios, que son los siguientes:

- Admin: usuario que maneja toda la aplicación.
- Usuario: cada usuario que accede a reservar instalaciones.

1.1. TRADUCCIÓN DEL RESUMEN AL INGLÉS

Sobway (Smart Open Building) is a group of professionals in the communications and automation technology sector with extensive experience in Real Estate and Property Management. They carry out a diagnosis of the possibilities of home automation that the building has, and provide a customised and personalised solution that will convert the residential building into a SmartBuilding. They wanted to broaden their field of action by adding reservations to the building installations, making them more efficient. The web application, in general terms, allows the management of these reservations, both for the owners and for people outside the community, by compensating expenses.

On the main page I've included four sections in the menu, which include the Home, Services, About Us, Clients, In addition, it also has a registration and login section. Once logged in, you have access to the private area where all the booking options that the user has are shown and you can access the facilities we have, observe their details, such as their location, photo gallery, ratings, brief description... and thus, make the reservation you consider appropriate, choosing the number of passes, shift, etc...

The application consists of several main interfaces, whose appearance and functionality will depend on the role of the registered user. The application is developed on the web, so that users can access it without the need to install software on their computer and with the only requirement of having a web browser and an internet connection. This is something that is very widespread in our country at the time of the development of this application.

The reason for the development of this project is given by the need of the owners to make their building efficient, as with this extension they will be able to compensate the community expenses, as they have a very high monthly expenditure due to the fact that they have so many installations. They can be informed at all times and in real time of the data of the people who make use of them, as well as the capacity, timetable, etc.

Several technologies have been used: Angular and Nodejs, the database will be SQL and for everything that has to do with users, Cognito from AWS (AmazonWebService) will be used.

In terms of functionality, we are looking for a manager in the style of Airbnb or even BlaBlacar, etc. but for facilities, for example, to be able to reserve a tennis court in a community for a certain number of hours or a swimming pool in a certain time slot... As for the design of the pages, I have used MaterializeCSS.

There are 2 user profiles, which are as follows:

- Admin: administrator of communities with facilities.

- User: each user who accesses to book facilities.

2. ENUNCIADO DEL PROBLEMA

Disponemos de varias urbanizaciones modernas que desean probar esta tecnología innovadora e introducir a sus comunidades de vecinos la posibilidad de realizar reservas de sus instalaciones por parte de personas ajenas al vecindario, de forma que su cuota de mantenimiento de la comunidad sea menos elevada, o incluso obtengan un beneficio para posibles mejoras de sus instalaciones.

Cada comunidad de vecinos cuenta con una o varias instalaciones, como podría ser una piscina, una pista de tenis o pádel, un gimnasio, y las más lujosas hasta ofrecen su spa.

Por cada comunidad queremos almacenar su nombre, la ciudad en la que está ubicada, y su referencia en el mapa, para que al cliente le sea más fácil el acceso.

Además de cada instalación se quiere almacenar la comunidad a la que pertenece, su hora de apertura y su hora de cierre, el aforo permitido, el tipo de instalación (si es una piscina, una sala de reuniones, un gimnasio…). También el precio que se piensa cobrar al cliente por el uso de la instalación, la calificación que se le da a diferentes aspectos de la instalación (como, por ejemplo, su limpieza, su ubicación, si está bien comunicado, y la

calidad de las instalaciones). Estos últimos datos, se obtendrán en una encuesta y no se contempla en el desarrollo de la app su actualización por la valoración de los clientes.

Se añadirán 3 atractivas imágenes por cada instalación, para que el usuario pueda ver detalladamente la instalación.

Además, cada instalación cuenta con la posibilidad de reservar en un turno determinado, por ejemplo, en las piscinas puede haber turno de mañana y turno de tarde.

Para poder realizar una reserva, el usuario deberá estar logueado previamente.

En cada reserva se podrán adquirir uno o más "pases", de acceso a la instalación deseada. El precio a abonar de dicha reserva se calculará mediante el producto del precio por turno de la instalación seleccionada, por la cantidad de pases a adquirir.

Una vez que se realiza la reserva, se añaden sus campos a la base de datos y se le muestra la reserva al administrador en el panel de administración. Al efectuar la reserva puede seleccionar el pago online a través de su cuenta de PayPal o con tarjeta de crédito o débito. También tiene la posibilidad de realizar el pago en el establecimiento, de forma que para cada reserva, existirá un campo de pago, que se pondrá a true o false dependiendo de si se efectúa o no el pago de la reserva.

Este usuario administrador, también podrá editar los campos de las instalaciones que ofrece, e incorporar al portal del usuario nuevas instalaciones.

Por otra parte, también se controlarán las instalaciones favoritas de cada cliente. Cada usuario puede tener 0 o varias instalaciones añadidas a sus favoritos, de forma que le sea más fácil encontrarlo para las siguientes reservas.

Por ello, la aplicación también contará con una sección de Área personal, donde le aparecerán sus instalaciones favoritas, por una parte, con la posibilidad de eliminarlas de su lista si lo desea. Por otra parte, también podrá acceder a todas las reservas que haya efectuado en la página web, por si quisiera efectuar la reserva nuevamente, el proceso sea más ágil.

3. TECNOLOGÍAS APLICADAS

- **Framework de desarrollo en el cliente: Angular**

Como Framework de desarrollo en el cliente, he utilizado Angular, framework opensource desarrollado por Google para facilitar la creación y programación de aplicaciones web de una sola página, las webs SPA (Single Page Application).

He decidido usar esta herramienta, ya que evita escribir código repetitivo, al separar completamente el frontend del backend y lo mantiene todo organizado gracias a su patrón MVC (Modelo-Vista-Controlador) asegurando los desarrollos con rapidez, a la vez que posibilita modificaciones y actualizaciones.

Además, es muy eficiente con la velocidad de carga, ya que, aunque puede resultar un poco lenta la primera vez que se abre, navegar después es totalmente instantáneo, ya que se ha cargado toda la página web de golpe. Otro factor a favor, es que, por su programación reactiva, la vista se actualiza automáticamente tras realizar los cambios.

- **Framework de desarrollo en el servidor: Node.js**

Node es un entorno JavaScript que nos permite ejecutar en el servidor, de manera asíncrona, sin necesidad de aprender ningún otro lenguaje de programación de entorno servidor como php.

Proporciona muchos beneficios a la web, como su alta velocidad de ejecución, y el más importante, dispone del Bucle de Eventos (Event Loop), que permitirá gestionar enormes cantidades de clientes de forma asíncrona. En vez de generar un nuevo hilo E/S para cada cliente, como se ha hecho tradicionalmente, cada conexión dispara la ejecución de un evento dentro del proceso del motor de Node. De este modo, Node permite que un solo servidor que lo ejecute pueda soportar decenas de miles de conexiones.

- **Base de datos: MySQL**

He elegido una base de datos relacional, como es MySQL. Es una herramienta muy útil y extendida, ya que es flexible y fácil de usar, tiene un alto rendimiento, ya que posibilita el almacenamiento de enormes cantidades de datos, sin problemas y con una velocidad óptima. Además, las industrias han estado usando MySQL durante años, lo que significa que hay abundantes recursos para desarrolladores calificados. Y lo más importante para una aplicación que gestiona datos, establece un alto estándar de seguridad.

- **Interfaz**

Para el aspecto de Interfaz de la página web, he utilizado algunas herramientas como:

- Bootstrap v4.6.0, ya que es un framework CSS muy cómodo para desarrollar aplicaciones que se adaptan a cualquier dispositivo.
- Materialize Css v1.0, también es un framework CSS, que aporta la posibilidad de crear aplicaciones web con las guías de Material Design.
- Fontawesome, para todos los iconos de la web, que la hacen más intuitiva y mejoran la experiencia de usuario.

- **Otras herramientas utilizadas:**
  - Amplify AWS: facilita el registro y control de usuarios de la aplicación web, a través del servicio de autentificación de Amazon, Cognito.
  - Express, elegido porque es el Framework más popular de Node, proporciona muchos mecanismos y facilita el desarrollo.
  - S3 de AWS. Es el sistema de almacenamiento en la nube de Amazon Web Services. Nos aporta el beneficio del alojamiento en la web y evita la sobrecarga innecesaria de la aplicación, haciéndola más ligera.

4. REQUERIMIENTOS HARDWARE Y SOFTWARE

El objetivo de la página web es que tenga el máximo alcance posible, por los que los requerimientos de hardware y software serán los mínimos. Bastará con un dispositivo que tenga conexión a internet.

- Navegadores soportados es móviles

	Chrome	Firefox	Safari
Android	X	X	-
iOS	X	X	X

- Navegadores soportados en Pc

	Chrome	Firefox	Safari	Opera
Mac	X	X	X	X
Windows	X	X	-	X
Linux	X	X	-	X

5. ANÁLISIS Y DISEÑO

Para solucionar el almacenamiento y gestión de los elementos del proyecto, se ha utilizado una base de datos relacional, MySQL concretamente.

La base de datos cuenta con 6 tablas: instalación, turno, turno_instalacion (cuya funcionalidad es guardar los turnos que tiene disponible una instalacion), reserva, likes, y comunidad.

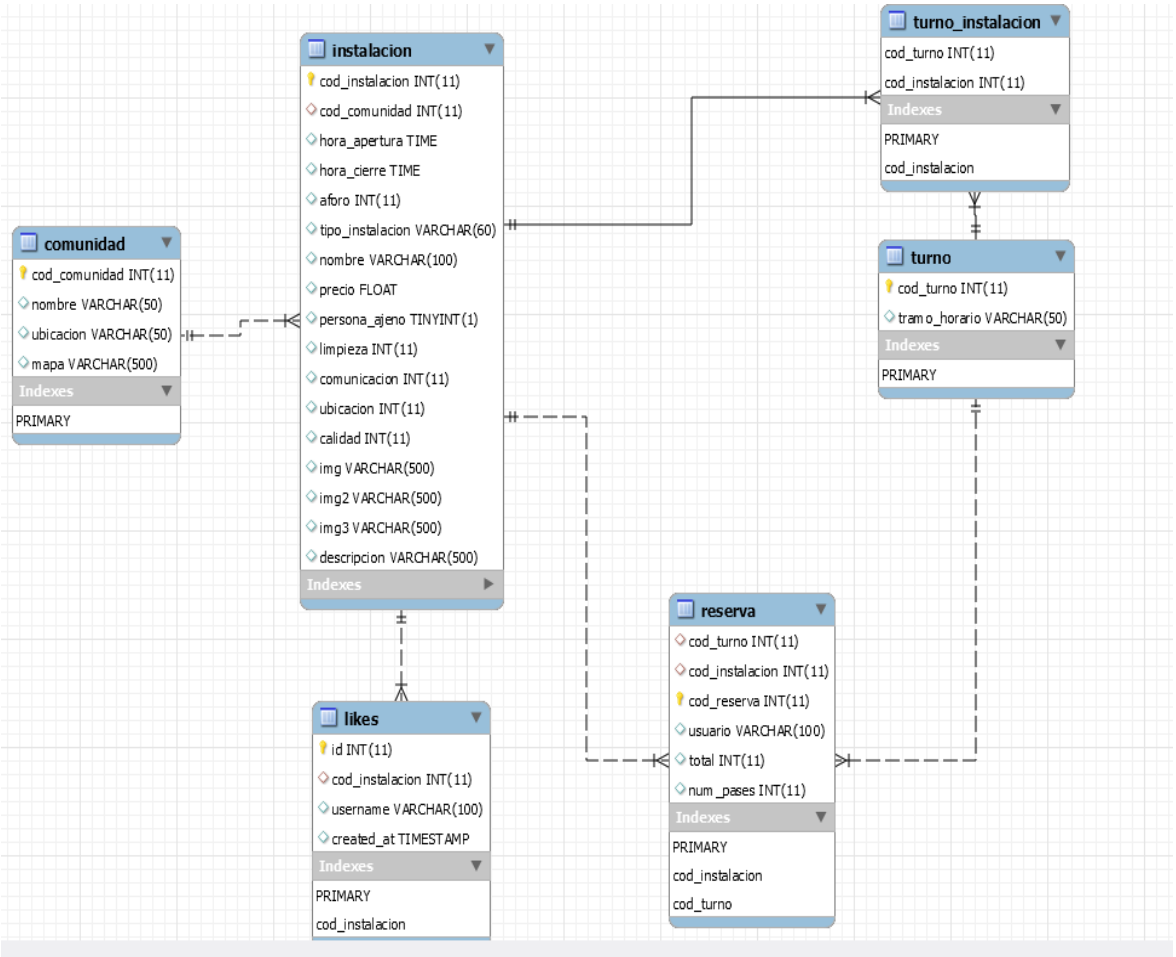

Ilustración 1 Base de Datos MySQL

6. IMPLEMENTACIÓN

Para la conexión de la base de datos y el servidor con el cliente utilizamos una API que contiene las rutas y los controladores necesarios para las solicitudes requeridas por el cliente, en esta parte se ha usado Express.

Para cada base de datos se ha creado una interfaz donde se guardan en el cliente sus respectivos datos.

Se usa un formulario para el registro y otro para la actualización y creación de nuevas instalaciones de las comunidades y se comprueba como se demuestra en el ejemplo del punto 8 que se han introducido bien los datos y que todos los campos están completos.

7. RELACIÓN DETALLADA DE CADA UNO DE LOS FICHEROS QUE SE INCLUYEN

A continuación, adjunto cada uno de los ficheros del proyecto, generados por la herramienta Docusaurus.

```
proyecto
├── client
│   ├── amplify
│   ├── angular.json
│   ├── dist
│   │   ├── 404.html
│   │   ├── index.html
│   │   ├── out-tsc
│   │   │   ├── client
│   │   │   │   ├── e2e
│   │   │   │   │   └── src
│   │   │   │   │       ├── app.e2e-spec.js
│   │   │   │   │       ├── app.e2e-spec.js.map
│   │   │   │   │       ├── app.po.js
│   │   │   │   │       └── app.po.js.map
│   │   │   │   └── src
│   │   │   │       ├── app
│   │   │   │       │   ├── app-routing.module.js
│   │   │   │       │   ├── app-routing.module.js.map
│   │   │   │       │   ├── app.component.js
│   │   │   │       │   ├── app.component.js.map
│   │   │   │       │   ├── app.component.spec.js
│   │   │   │       │   ├── app.component.spec.js.map
│   │   │   │       │   ├── app.module.js
│   │   │   │       │   ├── app.module.js.map
│   │   │   │       │   └── views
│   │   │   │       │       ├── activacion
│   │   │   │       │       │   ├── activacion.component.js
│   │   │   │       │       │   ├── activacion.component.js.map
│   │   │   │       │       │   ├── activacion.component.spec.js
│   │   │   │       │       │   └──
activacion.component.spec.js.map
│   │   │   │       │       ├── body
│   │   │   │       │       │   ├── body.component.js
│   │   │   │       │       │   ├── body.component.js.map
│   │   │   │       │       │   ├── body.component.spec.js
│   │   │   │       │       │   └── body.component.spec.js.map
```

```
│   │   │   │   │   ├── cabecer
│   │   │   │   │   │   ├── cabecer.component.js
│   │   │   │   │   │   ├── cabecer.component.js.map
│   │   │   │   │   │   ├── cabecer.component.spec.js
│   │   │   │   │   │   └── cabecer.component.spec.js.map
│   │   │   │   │   ├── clientes
│   │   │   │   │   │   ├── clientes.component.js
│   │   │   │   │   │   ├── clientes.component.js.map
│   │   │   │   │   │   ├── clientes.component.spec.js
│   │   │   │   │   │   └── clientes.component.spec.js.map
│   │   │   │   │   ├── contrasena
│   │   │   │   │   │   ├── contrasena.component.js
│   │   │   │   │   │   ├── contrasena.component.js.map
│   │   │   │   │   │   ├── contrasena.component.spec.js
│   │   │   │   │   │   └──
contrasena.component.spec.js.map
│   │   │   │   │   ├── cuerpo
│   │   │   │   │   │   ├── cuerpo.component.js
│   │   │   │   │   │   ├── cuerpo.component.js.map
│   │   │   │   │   │   ├── cuerpo.component.spec.js
│   │   │   │   │   │   └── cuerpo.component.spec.js.map
│   │   │   │   │   ├── footer
│   │   │   │   │   │   ├── footer.component.js
│   │   │   │   │   │   ├── footer.component.js.map
│   │   │   │   │   │   ├── footer.component.spec.js
│   │   │   │   │   │   └── footer.component.spec.js.map
│   │   │   │   │   ├── login
│   │   │   │   │   │   ├── login.component.js
│   │   │   │   │   │   ├── login.component.js.map
│   │   │   │   │   │   ├── login.component.spec.js
│   │   │   │   │   │   └── login.component.spec.js.map
│   │   │   │   │   ├── nosotros
│   │   │   │   │   │   ├── nosotros.component.js
│   │   │   │   │   │   ├── nosotros.component.js.map
│   │   │   │   │   │   ├── nosotros.component.spec.js
│   │   │   │   │   │   └── nosotros.component.spec.js.map
│   │   │   │   │   ├── recupera-codigo
│   │   │   │   │   │   ├── recupera-codigo.component.js
│   │   │   │   │   │   ├── recupera-
codigo.component.js.map
│   │   │   │   │   │   ├── recupera-
codigo.component.spec.js
│   │   │   │   │   │   └── recupera-
codigo.component.spec.js.map
│   │   │   │   │   ├── registro
│   │   │   │   │   │   ├── registro.component.js
│   │   │   │   │   │   ├── registro.component.js.map
│   │   │   │   │   │   ├── registro.component.spec.js
│   │   │   │   │   │   └── registro.component.spec.js.map
│   │   │   │   │   └── servicios
│   │   │   │   │       ├── servicios.component.js
│   │   │   │   │       ├── servicios.component.js.map
│   │   │   │   │       ├── servicios.component.spec.js
│   │   │   │   │       └── servicios.component.spec.js.map
│   │   │   │   ├── environments
│   │   │   │   │   ├── environment.js
│   │   │   │   │   ├── environment.js.map
│   │   │   │   │   ├── environment.prod.js
```

```
│   │   │   │   │   └── environment.prod.js.map
│   │   │   │   ├── main.js
│   │   │   │   ├── main.js.map
│   │   │   │   ├── polyfills.js
│   │   │   │   ├── polyfills.js.map
│   │   │   │   ├── test.js
│   │   │   │   └── test.js.map
│   │   │   └── server
│   │   │       └── src
│   │   │           ├── index.js
│   │   │           └── index.js.map
│   │   └── proyecto
│   │       ├── assets
│   │       │   ├── img
│   │       │   │   ├── edificio.jpg
│   │       │   │   ├── empresas.jpg
│   │       │   │   ├── equipo-gestión-proyectos.jpg
│   │       │   │   ├── icon blanco.png
│   │       │   │   ├── iconoTransparente.png
│   │       │   │   ├── inquilino.jpg
│   │       │   │   ├── promotores.jpg
│   │       │   │   ├── property-management.jpg
│   │       │   │   └── soluciones.jpg
│   │       │   └── scripts
│   │       │       └── Scrollify-master
│   │       │           ├── bower.json
│   │       │           ├── jquery.scrollify.js
│   │       │           ├── LICENSE
│   │       │           ├── package.json
│   │       │           ├── README.md
│   │       │           └── scrollify.jquery.json
│   │       ├── common.js
│   │       ├── common.js.map
│   │       ├── empresas.jpg
│   │       ├── favicon.ico
│   │       ├── index.html
│   │       ├── inquilino.jpg
│   │       ├── main.js
│   │       ├── main.js.map
│   │       ├── owl.video.play.png
│   │       ├── polyfills-core-js.js
│   │       ├── polyfills-core-js.js.map
│   │       ├── polyfills-css-shim.js
│   │       ├── polyfills-css-shim.js.map
│   │       ├── polyfills-dom.js
│   │       ├── polyfills-dom.js.map
│   │       ├── polyfills.js
│   │       ├── polyfills.js.map
│   │       ├── promotores.jpg
│   │       ├── property-management.jpg
│   │       ├── runtime.js
│   │       ├── runtime.js.map
│   │       ├── scripts.js
│   │       ├── scripts.js.map
│   │       ├── shadow-css-360bb6da-js.js
│   │       ├── shadow-css-360bb6da-js.js.map
│   │       ├── styles.css
│   │       ├── styles.css.map
```

```
|   ├── e2e
|   |   ├── protractor.conf.js
|   |   ├── src
|   |   |   ├── app.e2e-spec.ts
|   |   |   └── app.po.ts
|   |   └── tsconfig.json
|   ├── firebase.json
|   ├── karma.conf.js
|   ├── package-lock.json
|   ├── package.json
|   ├── README.md
|   ├── src
|   |   ├── amplify
|   |   |   ├── #current-cloud-backend
|   |   |   |   ├── amplify-meta.json
|   |   |   |   ├── auth
|   |   |   |   |   └── proyectoe209f3ab
|   |   |   |   |       ├── parameters.json
|   |   |   |   |       └── proyectoe209f3ab-cloudformation-
template.yml
|   |   |   |   ├── backend-config.json
|   |   |   |   ├── hosting
|   |   |   |   |   └── amplifyhosting
|   |   |   |   |       └── amplifyhosting-template.json
|   |   |   |   └── tags.json
|   |   |   ├── backend
|   |   |   |   ├── amplify-meta.json
|   |   |   |   ├── auth
|   |   |   |   |   └── proyectoe209f3ab
|   |   |   |   |       ├── parameters.json
|   |   |   |   |       └── proyectoe209f3ab-cloudformation-
template.yml
|   |   |   |   ├── awscloudformation
|   |   |   |   |   └── nested-cloudformation-stack.yml
|   |   |   |   ├── backend-config.json
|   |   |   |   ├── hosting
|   |   |   |   |   └── amplifyhosting
|   |   |   |   |       └── amplifyhosting-template.json
|   |   |   |   └── tags.json
|   |   |   ├── cli.json
|   |   |   ├── README.md
|   |   |   └── team-provider-info.json
|   |   ├── app
|   |   |   ├── admin
|   |   |   ├── app-routing.module.ts
|   |   |   ├── app.component.html
|   |   |   ├── app.component.scss
|   |   |   ├── app.component.spec.ts
|   |   |   ├── app.component.ts
|   |   |   ├── app.module.ts
|   |   |   ├── area-personal
|   |   |   ├── db.ts
|   |   |   ├── guards
|   |   |   |   ├── login.guard.spec.ts
|   |   |   |   └── login.guard.ts
|   |   |   ├── models
|   |   |   |   ├── Comunidad.ts
|   |   |   |   ├── horario.ts
```

```
│   │   │   │   ├── Instalacion.ts
│   │   │   │   ├── Like.ts
│   │   │   │   ├── reserva.ts
│   │   │   │   └── user.ts
│   │   │   ├── services
│   │   │   │   ├── auth-service.service.spec.ts
│   │   │   │   ├── auth-service.service.ts
│   │   │   │   ├── comunidad.service.spec.ts
│   │   │   │   ├── comunidad.service.ts
│   │   │   │   ├── horario.service.spec.ts
│   │   │   │   ├── horario.service.ts
│   │   │   │   ├── instalaciones.service.spec.ts
│   │   │   │   ├── instalaciones.service.ts
│   │   │   │   ├── likes.service.spec.ts
│   │   │   │   ├── likes.service.ts
│   │   │   │   ├── login.service.spec.ts
│   │   │   │   ├── login.service.ts
│   │   │   │   ├── reserva.service.spec.ts
│   │   │   │   └── reserva.service.ts
│   │   │   └── views
│   │   │       ├── activacion
│   │   │       │   ├── activacion.component.html
│   │   │       │   ├── activacion.component.scss
│   │   │       │   ├── activacion.component.spec.ts
│   │   │       │   └── activacion.component.ts
│   │   │       ├── admin
│   │   │       │   ├── admin.component.html
│   │   │       │   ├── admin.component.scss
│   │   │       │   ├── admin.component.spec.ts
│   │   │       │   └── admin.component.ts
│   │   │       ├── area-personal
│   │   │       │   ├── area-personal.component.html
│   │   │       │   ├── area-personal.component.scss
│   │   │       │   ├── area-personal.component.spec.ts
│   │   │       │   └── area-personal.component.ts
│   │   │       ├── body
│   │   │       │   ├── body.component.html
│   │   │       │   ├── body.component.scss
│   │   │       │   ├── body.component.spec.ts
│   │   │       │   └── body.component.ts
│   │   │       ├── cabecer
│   │   │       │   ├── cabecer.component.html
│   │   │       │   ├── cabecer.component.scss
│   │   │       │   ├── cabecer.component.spec.ts
│   │   │       │   └── cabecer.component.ts
│   │   │       ├── clientes
│   │   │       │   ├── clientes.component.html
│   │   │       │   ├── clientes.component.scss
│   │   │       │   ├── clientes.component.spec.ts
│   │   │       │   └── clientes.component.ts
│   │   │       ├── confirmacion
│   │   │       │   ├── confirmacion.component.html
│   │   │       │   ├── confirmacion.component.scss
│   │   │       │   ├── confirmacion.component.spec.ts
│   │   │       │   └── confirmacion.component.ts
│   │   │       ├── contrasena
│   │   │       │   ├── contrasena.component.html
│   │   │       │   ├── contrasena.component.scss
```

```
│   │   │   │   │   ├── contrasena.component.spec.ts
│   │   │   │   │   └── contrasena.component.ts
│   │   │   │   ├── cuerpo
│   │   │   │   │   ├── cuerpo.component.html
│   │   │   │   │   ├── cuerpo.component.scss
│   │   │   │   │   ├── cuerpo.component.spec.ts
│   │   │   │   │   └── cuerpo.component.ts
│   │   │   │   ├── detalle-instalacion
│   │   │   │   │   ├── detalle-instalacion.component.html
│   │   │   │   │   ├── detalle-instalacion.component.scss
│   │   │   │   │   ├── detalle-instalacion.component.spec.ts
│   │   │   │   │   └── detalle-instalacion.component.ts
│   │   │   │   ├── footer
│   │   │   │   │   ├── footer.component.html
│   │   │   │   │   ├── footer.component.scss
│   │   │   │   │   ├── footer.component.spec.ts
│   │   │   │   │   └── footer.component.ts
│   │   │   │   ├── instalaciones-form
│   │   │   │   │   ├── instalaciones-form.component.html
│   │   │   │   │   ├── instalaciones-form.component.scss
│   │   │   │   │   ├── instalaciones-form.component.spec.ts
│   │   │   │   │   └── instalaciones-form.component.ts
│   │   │   │   ├── instalaciones-list
│   │   │   │   │   ├── instalaciones-list.component.html
│   │   │   │   │   ├── instalaciones-list.component.scss
│   │   │   │   │   ├── instalaciones-list.component.spec.ts
│   │   │   │   │   └── instalaciones-list.component.ts
│   │   │   │   ├── login
│   │   │   │   │   ├── login.component.html
│   │   │   │   │   ├── login.component.scss
│   │   │   │   │   ├── login.component.spec.ts
│   │   │   │   │   └── login.component.ts
│   │   │   │   ├── nf
│   │   │   │   │   ├── nf.component.html
│   │   │   │   │   ├── nf.component.scss
│   │   │   │   │   ├── nf.component.spec.ts
│   │   │   │   │   └── nf.component.ts
│   │   │   │   ├── nosotros
│   │   │   │   │   ├── nosotros.component.html
│   │   │   │   │   ├── nosotros.component.scss
│   │   │   │   │   ├── nosotros.component.spec.ts
│   │   │   │   │   └── nosotros.component.ts
│   │   │   │   ├── recupera-codigo
│   │   │   │   │   ├── recupera-codigo.component.html
│   │   │   │   │   ├── recupera-codigo.component.scss
│   │   │   │   │   ├── recupera-codigo.component.spec.ts
│   │   │   │   │   └── recupera-codigo.component.ts
│   │   │   │   ├── registro
│   │   │   │   │   ├── registro.component.html
│   │   │   │   │   ├── registro.component.scss
│   │   │   │   │   ├── registro.component.spec.ts
│   │   │   │   │   └── registro.component.ts
│   │   │   │   └── servicios
│   │   │   │       ├── servicios.component.html
│   │   │   │       ├── servicios.component.scss
│   │   │   │       ├── servicios.component.spec.ts
│   │   │   │       └── servicios.component.ts
│   │   │   ├── assets
```

```
│   │   │   ├── img
│   │   │   │   ├── edificio.jpg
│   │   │   │   ├── empresas.jpg
│   │   │   │   ├── equipo-gestión-proyectos.jpg
│   │   │   │   ├── icon blanco.png
│   │   │   │   ├── iconoTransparente.png
│   │   │   │   ├── inquilino.jpg
│   │   │   │   ├── promotores.jpg
│   │   │   │   ├── property-management.jpg
│   │   │   │   └── soluciones.jpg
│   │   │   └── scripts
│   │   │       └── Scrollify-master
│   │   │           ├── bower.json
│   │   │           ├── jquery.scrollify.js
│   │   │           ├── LICENSE
│   │   │           ├── package.json
│   │   │           ├── README.md
│   │   │           └── scrollify.jquery.json
│   │   ├── aws-exports.js
│   │   ├── environments
│   │   │   ├── environment.prod.ts
│   │   │   └── environment.ts
│   │   ├── favicon.ico
│   │   ├── index.html
│   │   ├── main.ts
│   │   ├── polyfills.ts
│   │   ├── src
│   │   │   └── aws-exports.js
│   │   ├── styles.scss
│   │   └── test.ts
│   ├── tsconfig.app.json
│   ├── tsconfig.json
│   ├── tsconfig.spec.json
│   └── tslint.json
├── database
│   └── database.sql
├── docker-compose.yml
├── Dockerfile
├── docs
│   ├── doc1.md
│   ├── doc2.md
│   ├── doc3.md
│   ├── exampledoc4.md
│   └── exampledoc5.md
├── README.md
├── server
│   ├── build
│   │   ├── controllers
│   │   │   ├── comunidadController.js
│   │   │   ├── horarioController.js
│   │   │   ├── indexControllers.js
│   │   │   ├── instalacionesControllers.js
│   │   │   ├── likesController.js
│   │   │   └── reservasController.js
│   │   ├── database.js
│   │   ├── index.js
│   │   ├── keys.js
│   │   └── routes
```

```
│   │   │   ├── comunidadRoutes.js
│   │   │   ├── horarioRoutes.js
│   │   │   ├── indexRoutes.js
│   │   │   ├── instalacionRoutes.js
│   │   │   ├── likesRoutes.js
│   │   │   └── reservasRoutes.js
│   ├── package-lock.json
│   ├── package.json
│   ├── src
│   │   ├── controllers
│   │   │   ├── comunidadController.ts
│   │   │   ├── horarioController.ts
│   │   │   ├── indexControllers.ts
│   │   │   ├── instalacionesControllers.ts
│   │   │   ├── likesController.ts
│   │   │   └── reservasController.ts
│   │   ├── database.ts
│   │   ├── index.ts
│   │   ├── keys.ts
│   │   └── routes
│   │   │   ├── comunidadRoutes.ts
│   │   │   ├── horarioRoutes.ts
│   │   │   ├── indexRoutes.ts
│   │   │   ├── instalacionRoutes.ts
│   │   │   ├── likesRoutes.ts
│   │   │   └── reservasRoutes.ts
```

8. EVALUACIÓN Y PRUEBA

Si no rellenas todos los campos salen mensajes Notify recordando al usuario que rellene cada campo requerido y así se evitan fallos del sistema. Se añade captura de comprobación para visualizarlo.

Por otro lado, para probar el correcto funcionamiento de la API el servidor en NODE.js y la base de datos, se ha utilizado la herramienta Postman que tras introducir una URL muestra la respuesta dependiendo de la clase de petición.

9. MANUAL DE USUARIO

- Con un dispositivo electrónico, acceder a un navegador web (Google, Mozilla Firefox…)
- Buscar la web (no se puede poner el enlace porque no se ha podido poner en producción)
- Dentro de la página, el usuario puede acceder a la sección que prefiera para informarse de quiénes son, datos de contacto, etc. A la derecha se encuentra un botón llamado LOGIN, clicar ahí y facilitar los datos para tener un acceso completo.

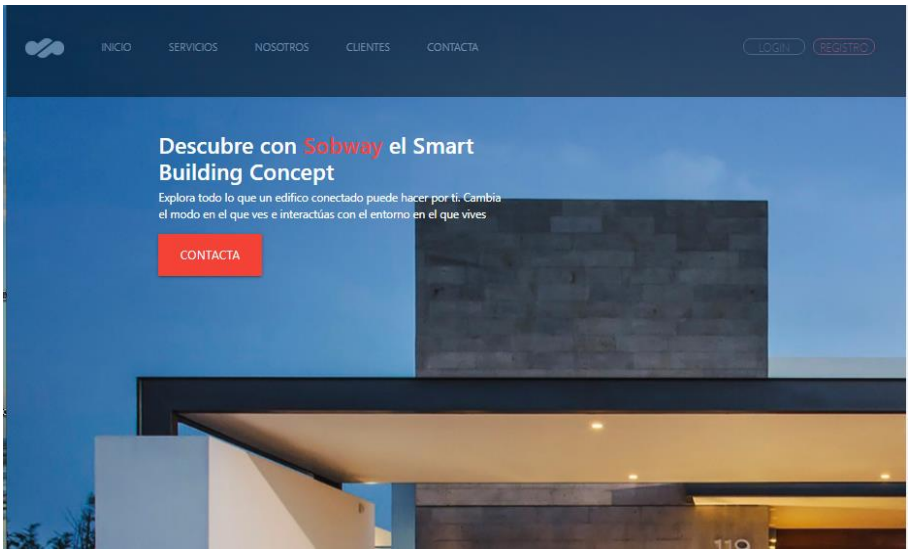

Ilustración 2 Página de inicio

- En el Footer se muestran los enlaces a las redes sociales, y otros enlaces de interés.

Ilustración 3 Footer

Si se pulsa el botón AWS se puede registrar con otros social providers.

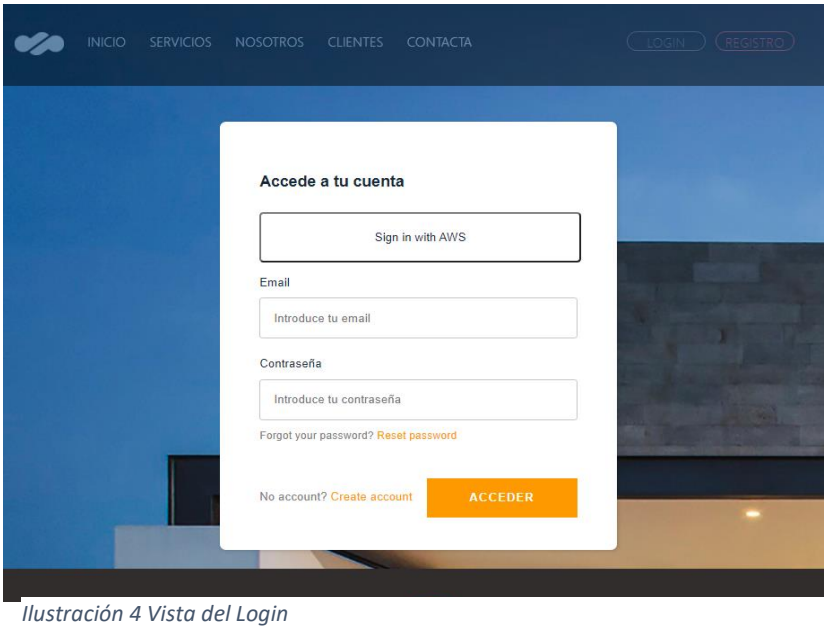

Ilustración 4 Vista del Login

- Si pulsamos en crear cuenta, nos aparecerá el siguiente formulario de registro.

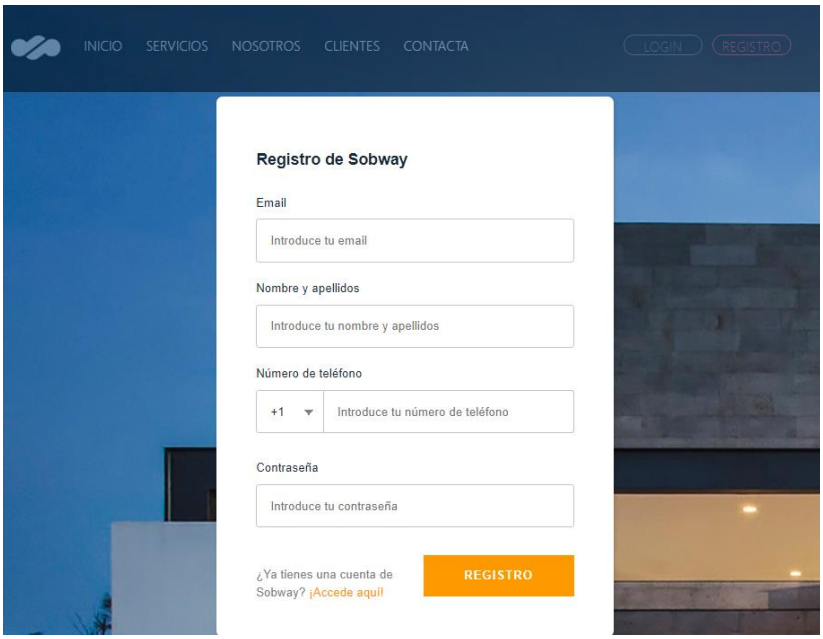

Ilustración 5 Formulario de registro

- Recibirá un correo electrónico a la cuenta de correo que haya facilitado con un código de verificación para completar el LOGIN.

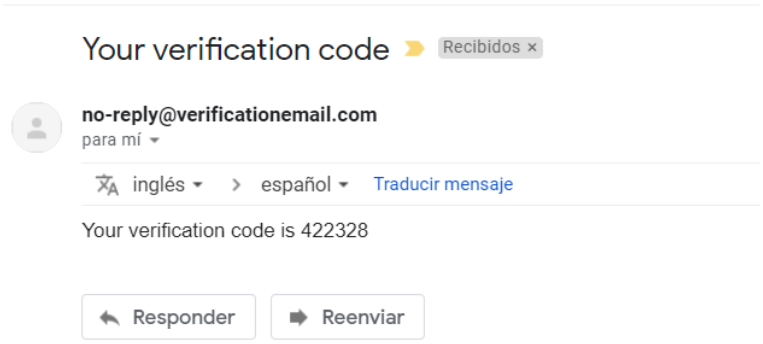

Ilustración 6 Código de verificación

- Una vez logeado, acceder a las instalaciones y visualizar todas las facilidades que ofrece cada comunidad, accediendo a cualquiera de ellas, se encuentran las valoraciones, fotografías, nombre, ubicación de la misma, precio pase, turnos… y por tanto indicando el número de pases y horario escogido, procederá a realizar la reserva clicando en el botón correspondiente.

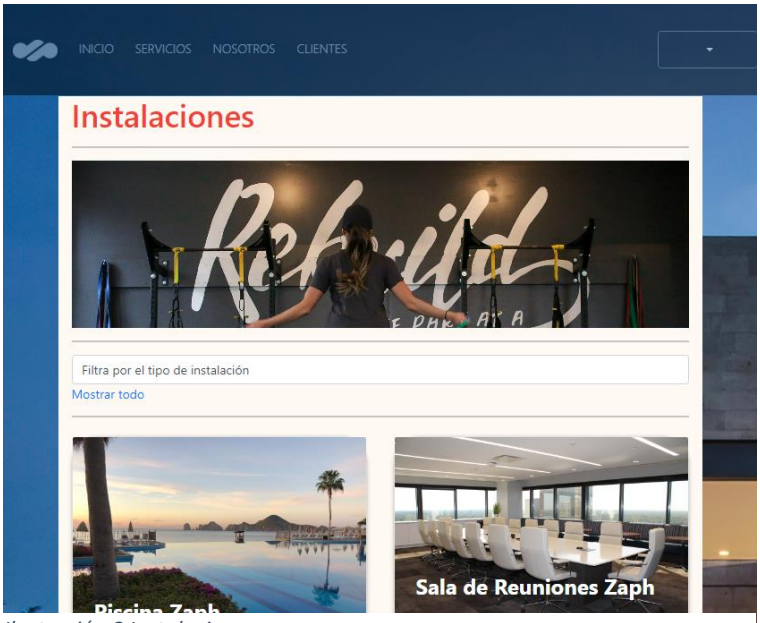

Ilustración 8 Instalaciones

Ilustración 7Instalación

- En las instalaciones nos aparece este select con el que podemos filtrar las instalaciones según el tipo de instalación que estamos buscando.

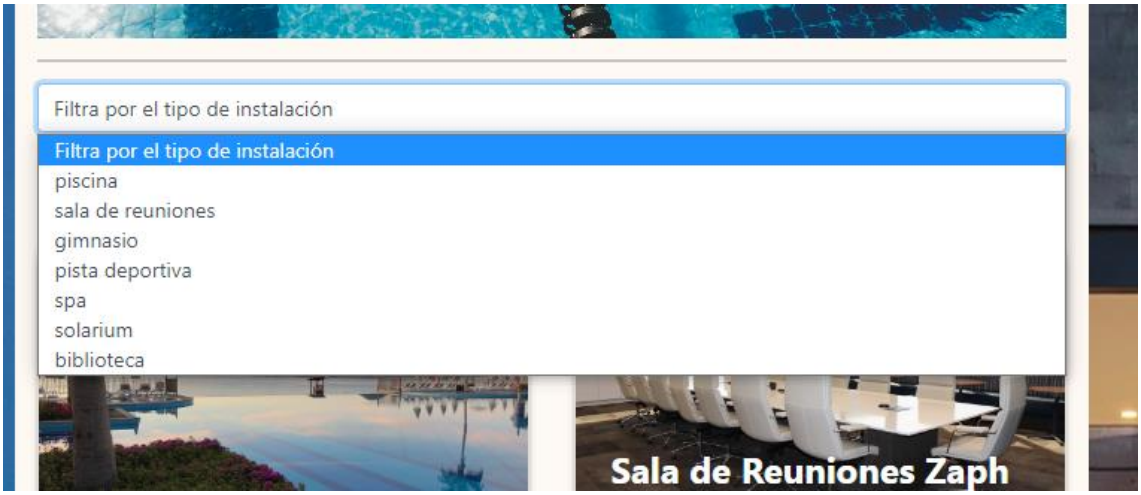

Ilustración 10 Select

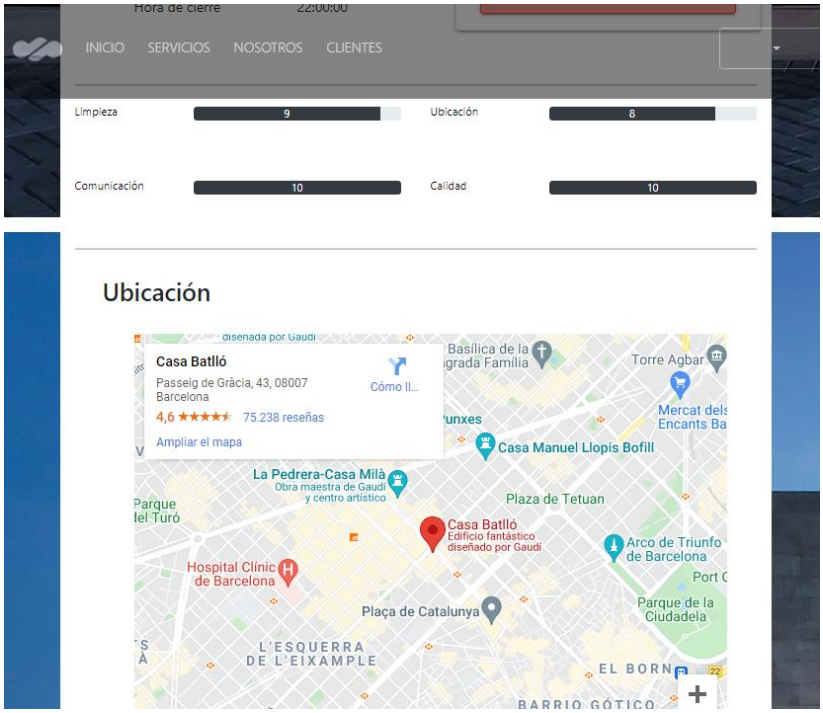

Ilustración 9 Detalles de la instalación

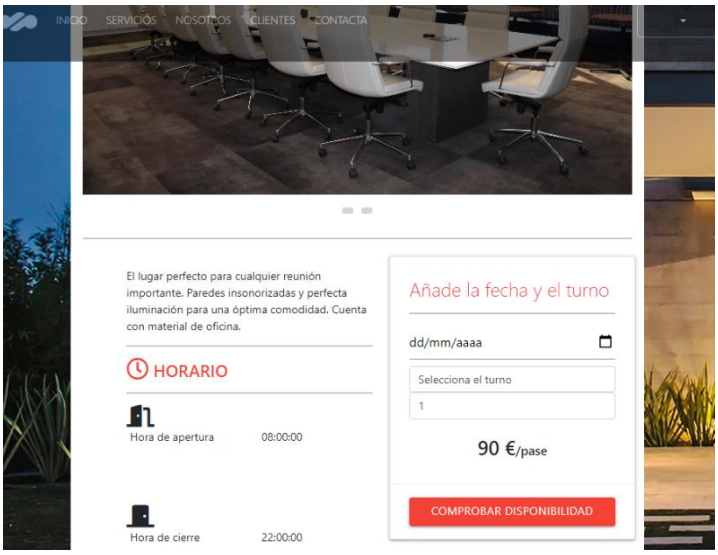

Ilustración 12 Detalles de la instalación

Para comprobar disponibilidad, aparece la siguiente pantalla:

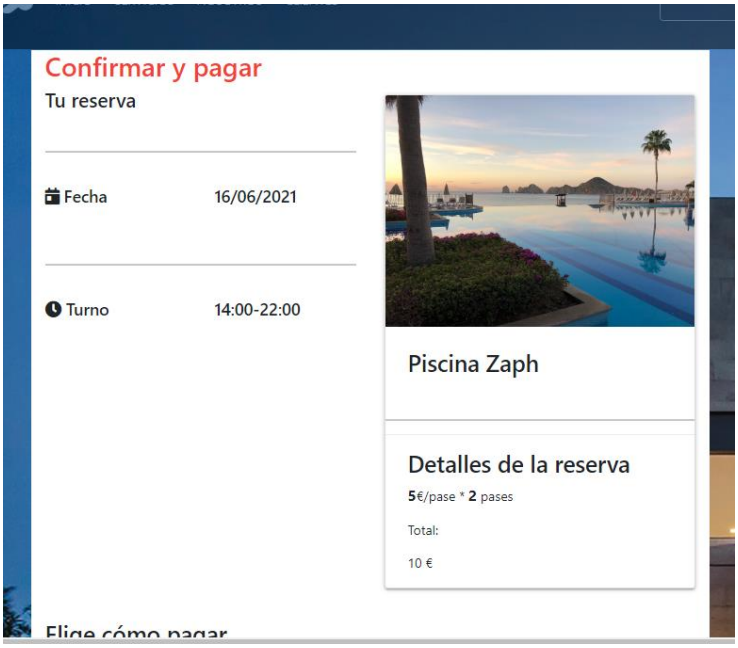

Ilustración 11 Confirmación

- Para proceder al pago, puede utilizar la plataforma Paypal, pagar con tarjeta de crédito o débito, o escoger pagar en el establecimiento.

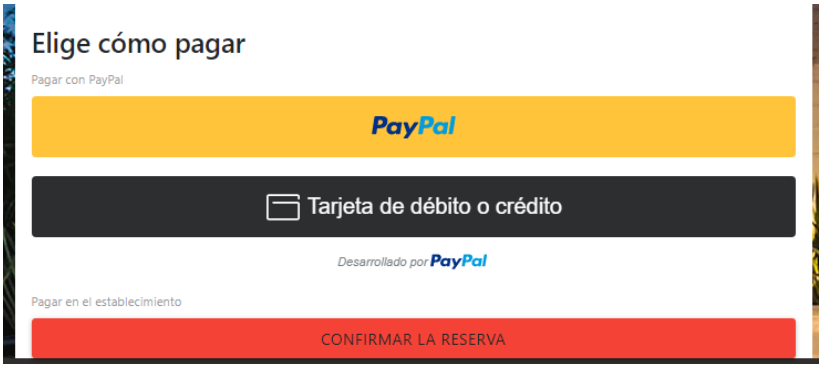

Ilustración 13 Pago

- Tras este paso, en su área personal aparece la reserva que ha realizado. Puede reservar tantas veces y en tantas comunidades como desee.
- Cada usuario cuenta con su Área personal, donde tendrá un menú que le permitirá acceder a sus instalaciones guardadas en favoritos, y a las reservas que ha realizado.

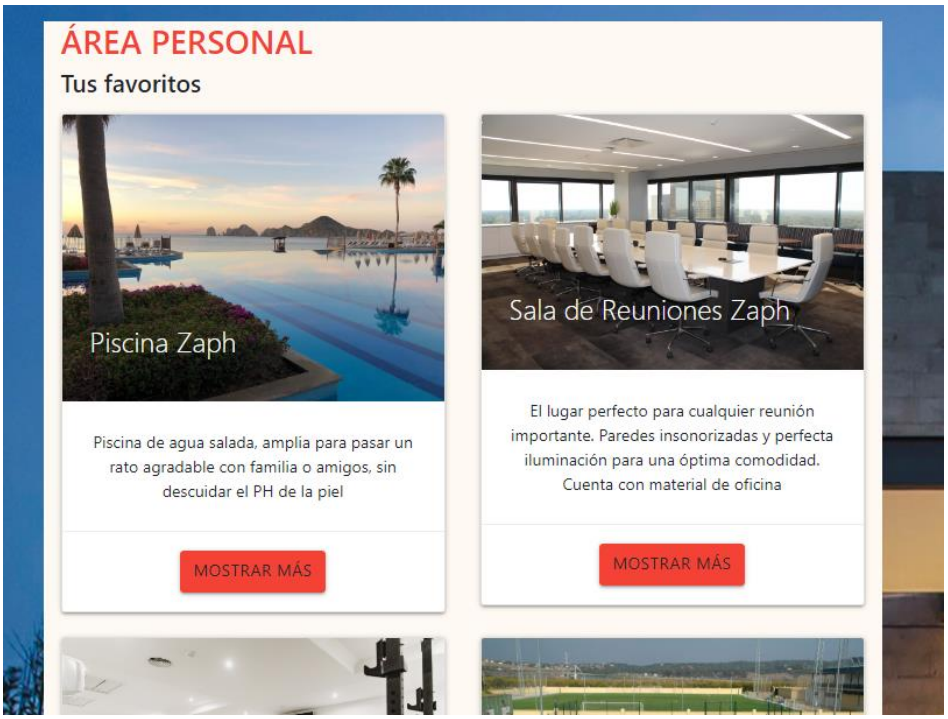

Ilustración 14 Área Personal

- *Por parte de la empresa:* Si el usuario que inicia sesión es el administrador, puede acceder al área de administración y editar las instalaciones.

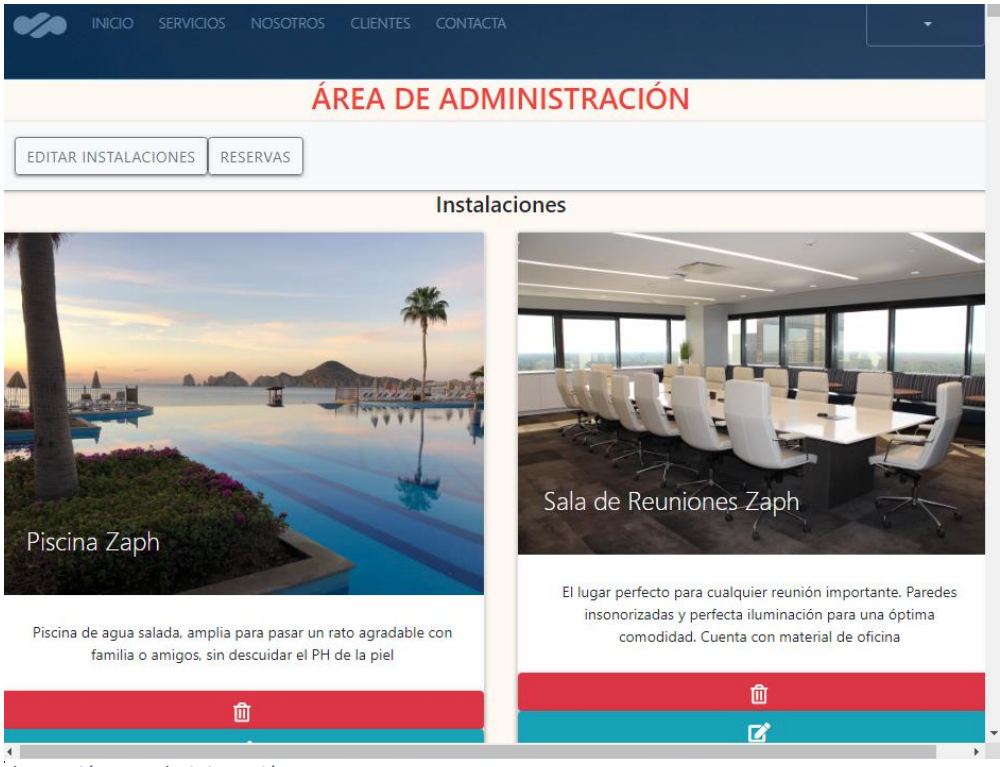

Ilustración 15 Administración

- Además de gestionar las reservas.

Ilustración 16 reservas

10. GUÍA DE INSTALACIÓN

No requiere la instalación de ningún programa en específico en el dispositivo electrónico, con acceso a internet y un navegador web, es suficiente.

11. SOFTWARE UTILIZADO

Se han utilizado varios servicios de Amazon Web Service:

- S3: Almacenamiento en la nube
- Cognito para registro de usuarios junto con Amplify AWS

Bootstrap para el diseño, todas las demás tecnologías se desarrollan en el punto 3.

12. MEJORAS POSIBLES Y APORTACIONES

- Una posible mejora sería la posibilidad por parte de los usuarios de quieran reservar cualquier instalación de las que se ofrecen, de añadir la valoración que le dan esos clientes a los aspectos de la instalación contemplados (limpieza, calidad, comunicación, ubicación), e incluso la posibilidad de añadir comentarios y fotos sobre su experiencia.
- Otra mejora sería añadir un usuario administrador en cada comunidad, que fuese gestionado por un miembro de la dicha urbanización y controlase directamente las reservas de sus instalaciones.

- Añadir un chat en el que el cliente pueda ponerse en contacto con el administrador de la instalación en la que está interesado en reservar, y así facilitar la comunicación.