



**Escuela Superior  
de Ingeniería y Tecnología**  
Universidad de La Laguna

# Trabajo de Fin de Grado

Grado en Ingeniería Informática

---

## FixAdvisor: Buscador de Servicios de Reparaciones del Hogar

*FixAdvisor: Home Repair Services Finder*

Óscar Moreira Estévez

---

La Laguna, 8 de *julio* de 2022

D. **Iván Castilla Rodríguez**, con N.I.F. 78.565.451-G profesor Contratado Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

## **CERTIFICA**

Que la presente memoria titulada:

*“FixAdvisor: Buscador de Servicios de Reparaciones del Hogar”*

Ha sido realizada bajo su dirección por D. **Óscar Moreira Estévez**, con N.I.F. 78.644.512-Z.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 8 de julio de 2022

# Agradecimientos

A mi familia, pareja y amigos por estar siempre a mi lado, apoyándome en los momentos más duros de la carrera. Sin ellos nada de esto hubiera sido posible, siempre confiaron en mí, incluso en los momentos en los que ni yo mismo lo hacía. Les quiero mucho.

A Marta y Vanessa, compañeras de momentos muy difíciles, momentos duros pero también de muchos momentos de alegrías. Sin duda un pilar fundamental para mí en este paso por la universidad y espero que continúe así.

A Nayara, mi manager del trabajo y proyecto en el que me encuentro. Gracias por comprender la situación de estrés, tranquilizarme y ayudarme a no estar más agobiado.

Gracias a todas y todos los compañeros con los que me he encontrado y que de una u otra forma han influido a lo largo de estos años haciéndome tener una mejor experiencia.

Por último pero no menos importante, a mi tutor Iván, por siempre estar ahí para mis dudas, debates técnicos y de ética y siempre estar con una sonrisa en nuestras reuniones. No podía haber tenido mejor tutor.

Muchas gracias a todos. Siempre les estaré agradecido por haber sido parte una parte fundamental de mi etapa universitaria.

# Licencia



© Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional.

## **Resumen**

Actualmente, cuando necesitamos cualquier cosa, donde primero acudimos es a Internet. Surgen así los buscadores de servicios, donde podemos encontrar información relevante y también reseñas con puntuaciones. Sin embargo, aún existen servicios que no cuentan con plataformas específicas. La finalidad de este trabajo ha sido crear una aplicación que agrupe a los profesionales de servicios como la fontanería, albañilería, carpintería y servicios similares donde estos puedan ofrecer sus trabajos y los potenciales clientes puedan buscar información sobre ellos y también puntuarlos. También será una aplicación donde los usuarios puedan guardar servicios como favoritos permitiéndoles consultarlos en futuras ocasiones. De esta manera, este proyecto podrá cubrir una necesidad actual.

**Palabras clave:** servicios, aplicación, desarrollo, buscador

## **Abstract**

Nowadays, when we need anything, the first thing we turn to is the Internet. This is how service search engines arise, where we can find relevant information and also reviews with scores of these. However, there are still services that do not have specific platforms. The purpose of this work has been to create an application that brings together service professionals such as plumbing, brickwork, carpentry and similar services where they can offer their work and potential clients can search for information about them or also rate them. It will also be an application where users can save services as favorites, allowing them to consult the services on future occasions. In this way, this project will be able to cover a current need.

**Keywords:** services, application, development, finder

# Índice general

<b>1. Introducción</b>	<b>11</b>
1.1 Motivación	12
1.2 Antecedentes	13
1.3 Objetivos	14
<b>2. Desarrollo</b>	<b>15</b>
2.1 Metodología de trabajo	15
2.2 Requisitos de la aplicación	16
2.3 Arquitectura de la aplicación y tecnologías	18
2.3.1 Base de datos	19
2.3.2 Autenticación de usuarios	20
2.3.3 Backend	20
2.3.4 Frontend	21
2.4 Implementación	24
2.4.1 Base de datos	24
2.4.2 Backend	26
2.4.3 Frontend	29
2.5 Despliegue	38
<b>3. Conclusiones y líneas futuras</b>	<b>40</b>
<b>4. Conclusions and future works</b>	<b>42</b>
<b>5. Presupuesto</b>	<b>44</b>
<b>Bibliografía</b>	<b>45</b>

# Índice de figuras

Figura 2.1. Arquitectura general de la aplicación	19
Figura 2.2. Barra menú sin iniciar sesión	30
Figura 2.3. Barra menú sin iniciar sesión	30
Figura 2.4. Barra menú sin iniciar sesión	30
Figura 2.5. Página de inicio de sesión	31
Figura 2.6. Página de registro de un cliente	31
Figura 2.7. Página de registro de proveedor	32
Figura 2.8. Página creación de servicio	33
Figura 2.9. Página principal	33
Figura 2.10. Página principal de un usuario cliente	34
Figura 2.11. Página de servicios favoritos	34
Figura 2.12. Página detalles de un servicio	35
Figura 2.13. Página detalles de un servicio - opiniones	35
Figura 2.14. Crear opinión sobre servicio	36
Figura 2.15. Perfil usuario proveedor	36
Figura 2.16. Perfil usuario cliente	37
Figura 2.17 Ejecución GitHub Action	39



# Índice de tablas

Tabla 2.1. Requisitos específicos para los usuarios	17
Tabla 2.2. Endpoints API REST	26
Tabla 5.1: Presupuesto por trabajo realizado	43

# Capítulo 1

## Introducción

Hoy en día se pueden encontrar muchas plataformas donde los usuarios pueden dar su opinión sobre diferentes tipos de servicios. Los más comunes suelen ser restaurantes, bares, hoteles y tiendas. Por otro lado, todavía hay sectores que no tienen presencia en este tipo de plataformas, como es el caso de profesionales de la fontanería, carpintería y albañilería, entre otros.

Cuando se estropea algo en casa, si no tenemos a un familiar o un conocido que pueda hacerlo intentamos hacerlo nosotros mismos, lo que puede causar más problemas. En el momento en el que queramos contratar a un profesional para que solucione el problema, nos daremos cuenta de que no existe una plataforma concreta donde buscar los servicios de estas personas y opiniones de otros clientes respecto a los servicios contratados.

Estos profesionales normalmente ofrecen sus servicios a través del tradicional “boca a boca” o mediante las redes sociales en sus perfiles personales, lo que ocasiona problemas a las personas que buscan este tipo de servicios ya que no pueden conocer las opiniones de otros clientes o acceder a estos profesionales sin necesidad de buscar una persona en común.

Muchos profesionales de estos sectores prefieren seguir anunciándose de una manera más tradicional y es probable que muchos de ellos no se hayan planteado la necesidad de estar presente en una plataforma donde poder ofrecer sus servicios. Una de las razones de esto es un posible miedo a recibir opiniones negativas de los clientes ya que se puede crear una mala imagen del servicio o empresa y no volver a ser contratados debido a las reseñas en este tipo de plataformas.

Por otro lado, la existencia de este tipo de aplicaciones también hace que las empresas y los profesionales intenten ofrecer un mejor servicio tanto a nivel profesional como personal con los clientes para así no recibir malas críticas.

A pesar de las problemáticas comentadas anteriormente para los profesionales y empresas que quieran ofrecer sus servicios en estas plataformas, son cada vez más las que existen teniendo un gran éxito para los usuarios que las usan antes de reservar o contratar un servicio, gracias a que evitarán malas experiencias viendo las opiniones de otros clientes. Por otro lado, los profesionales o empresas que se promocionen en estas plataformas y tengan buenas críticas tendrán una gran publicidad y conseguirán más clientes gracias a esto.

## 1.1 Motivación

La idea de este proyecto surge de la necesidad de buscar profesionales de la carpintería para hacer reformas en una casa. Tras realizar la búsqueda surgió un problema: no se encontraron plataformas donde poder hallar a los profesionales que se necesitaban con información, opiniones y puntuación sobre los servicios que realizan.

Haciendo un análisis más exhaustivo se llegó a la conclusión de que existen plataformas que permiten valorar otros servicios pero no los que se estaban buscando para cubrir esta necesidad. Como se comentó anteriormente, la mayoría de estas plataformas se basan en sectores como la restauración, hostelería, etc.

A raíz de esto, surge la idea de crear una plataforma que agrupe a los profesionales de la fontanería, albañilería, carpintería y los relacionados con la construcción o arreglos en los hogares.

La búsqueda de una solución a esta problemática ha desembocado en el desarrollo de este proyecto que tiene en cuenta tanto a las personas que buscan estos servicios como a los profesionales que los ofertan.

## 1.2 Antecedentes

Como se ha comentado anteriormente ya existen aplicaciones que incluyen la oferta de estos una gran variedad de servicios pero no en los que se enfoca este proyecto. Algunas de ellas que analizaremos en profundidad son *TripAdvisor* [1], *Google Reviews* [2] y *Yelp* [3].

*TripAdvisor* [1] es una de las plataformas más conocidas y usadas en la actualidad. Esta te ofrece una gran variedad de servicios para que puedas consultar su información, ver y crear opiniones y puntuar servicios, entre otras. Esta aplicación cada año ofrece premios a los lugares o servicios mejor valorados por sus usuarios, un distintivo muy valorado por los clientes y los dueños de los servicios.

Sin embargo, se ha detectado un aspecto negativo en dicha aplicación: cualquier usuario puede crear el perfil de un servicio no existente hasta el momento en la plataforma, para opinar sobre este sin el conocimiento ni el consentimiento del propietario. Se ha valorado como algo negativo ya que quien ofrece el servicio no tiene control sobre lo que se publica sobre su servicio ni puede responder a las reseñas que se hagan sobre este. Esto implica que no puede solucionar los aspectos negativos debido a que no sabe de su existencia. También es posible que este propietario simplemente no quiera aparecer en estas plataformas.

Debido a este problema, se ha decidido que en este proyecto sólo podrán crear servicios aquellas personas que se registren como profesionales para evitar que se publiquen servicios sin el consentimiento de los dueños.

Otra herramienta muy usada es *Google Reviews* [2] que da información sobre servicios, lugares, etc. y que permite consultar y hacer comentarios y valoraciones sobre estos. Sin embargo no es una plataforma como pueden ser las otras ya que esta está integrada en las búsquedas de google.

Por último, *Yelp* [3] es una plataforma similar a *TripAdvisor* [1] ya que contiene varios tipos de servicios, con información sobre estos, opiniones, etc. Esta es la única plataforma que se ha encontrado que contiene servicios como los que motivaron este trabajo como son la fontanería, carpintería, albañilería, etc. El problema de esta aplicación es que no es específica de estos servicios, lo que implica que muchas personas, tanto usuarias como profesionales, no sepan de esta característica de la aplicación.

El análisis de estas herramientas han llevado a la elaboración de una serie de características que se creen necesarias en una aplicación específica como la de este proyecto para agrupar servicios como la carpintería, albañilería, fontanería, etc.

## 1.3 Objetivos

Este proyecto consiste en crear una aplicación web donde usuarios como fontaneros, albañiles, carpinteros y similares puedan ofrecer sus servicios en ella y a su vez que usuarios clientes puedan acceder a ellos.

Teniendo esto en cuenta, la aplicación tendrá un *Backend* para poder crear usuarios, publicar fotos, añadir valoraciones y demás. Por otra parte tendrá un *Frontend* usando tecnologías web con una interfaz de usuario sencilla para que ningún usuario tenga complicaciones para usarla.

Objetivos:

- Análisis de herramientas similares
- Identificar requisitos de este tipo de herramientas
- Diseño de la arquitectura de la aplicación.
- Definición del modelo de base de datos e implementación de la misma.
- Implementar autenticación de usuarios
- Definición e implementación de la API REST
- Definición e implementación del despliegue de la aplicación.
- Diseños e implementación del *Frontend* de la aplicación
- Implementar subida de imágenes a un servicio *cloud*.
- Documentación

# Capítulo 2

## Desarrollo

En este capítulo se explicará cómo ha sido todo el desarrollo de la aplicación, explicando la metodología de trabajo usada, arquitecturas del software creado, tecnologías seleccionadas para este proyecto y justificación de las mismas. Este proyecto al ser una aplicación *Fullstack* tiene una gran variedad de partes como puede ser la base de datos, autenticación de usuarios el *Backend* con la API REST, *Frontend* de la aplicación, *testing* y un entorno creado para los despliegues de la aplicación con *continuous integration* y *continuous delivery/continuous deployment (CI/CD)*. Todo esto ha hecho que sea un proyecto muy complejo a nivel de desarrollo y de tecnologías.

### 2.1 Metodología de trabajo

Una de las partes más importantes para realizar un proyecto es contar con una buena metodología de trabajo ya que hará una mejor experiencia de desarrollo y eficaz. Una buena metodología debe contar con diferentes características importantes como pueden ser la comunicación, la gestión de las tareas, estimaciones, etc.

Para la realización del proyecto se ha usado una metodología ágil donde se realizaban reuniones cada 2 semanas para ver los avances del proyecto y proponer las siguientes funcionalidades a realizar.

Primero, antes de comenzar con el desarrollo, se pasaron por varias etapas. La primera de ellas fue un análisis de mercado sobre aplicaciones similares, luego, teniendo en cuenta esta información, se debatió sobre qué funcionalidades tiene que tener el proyecto y los objetivos, por último, teniendo claro los objetivos, se empezó con el desarrollo.

Por otra parte, se ha usado *GitHub* [6] no solo como plataforma para alojar el código, sino que también se han aprovechado las diferentes herramientas que nos proporciona. Una de estas herramientas es la funcionalidad de proyectos que se encuentra en el repositorio donde se pueden gestionar tareas mediante un tablero *Kanban* para crear las tareas a realizar y a su vez tener un flujo del estado de cada una de ellas basado en columnas pasando por diferentes estados como son: las tareas a realizar, tareas en progreso y por último tareas ya realizadas.

Repositorio Frontend: <https://github.com/omorest/FixAdvisor-FrontEnd>

Repositorio Backend: <https://github.com/omorest/FixAdvisor-BackEnd>

Luego, se ha usado también la funcionalidad de *issues* de *GitHub* [6] junto con las *pull requests*, de manera que todas las tareas creadas en el tablero comentado anteriormente

se conviertan en *issues* a resolver del proyecto y se cierren cuando hagamos *pull requests* donde resolvamos estas tareas.

Toda esta metodología realizada se ha hecho para hacer el proyecto más cercano a la realidad de cómo se trabaja en proyectos en una empresa y de esta manera no solo desarrollar la aplicación si no también aprender a gestionar un proyecto, separar tareas, estimarlas, etc.

## 2.2 Requisitos de la aplicación

Este trabajo comenzó a desarrollarse con un objetivo claro: crear un producto que permitiese buscar proveedores de servicios de arreglos en el hogar, como puede ser el caso de profesionales de la fontanería, albañilería y carpintería entre otros.

En esta aplicación podemos encontrar tres tipos diferentes de usuarios: los usuarios no registrados, los usuarios registrados, también llamados clientes y, por último, los proveedores de servicios. Debido a estas diferencias se han desarrollado diversos objetivos generales, además de objetivos específicos para cada tipo de usuario.

Requisitos generales:

- Se podrán registrar usuarios como clientes o como proveedores de servicios.
- Se podrá iniciar sesión o cerrar sesión para cualquier tipo de usuario registrado.
- Existirá una página donde se muestren todos los servicios.
- Se podrá ordenar los servicios por nombre, tipo, etc.
- Se podrá filtrar por tipo de servicio.
- Los usuarios podrán editar su información.
- Existirá una página de detalles para cada servicio.
- La información sobre servicios de la aplicación la podrá consultar cualquier tipo de usuario de la aplicación, incluyendo los no registrados.
- Existirá un buscador de servicios.

Los requisitos específicos para usuarios se pueden ver en la Tabla 2.1.

**Tabla 2.1. Requisitos específicos para los usuarios**

	No registrado	Registrado	Proveedor
Opinar sobre un servicio	✗	✓	✗
Responder opiniones	✗	✗	✓
Puntuar un servicio	✗	✓	✗
Añadir servicio a lista de favoritos	✗	✓	✗
Crear uno o varios servicios	✗	✗	✓
Consultar información sobre los servicios	✓	✓	✓
Publicar fotos sobre un servicio	✗	✗	✓
Eliminar un servicio	✗	✗	✓
Editar información de un servicio	✗	✗	✓



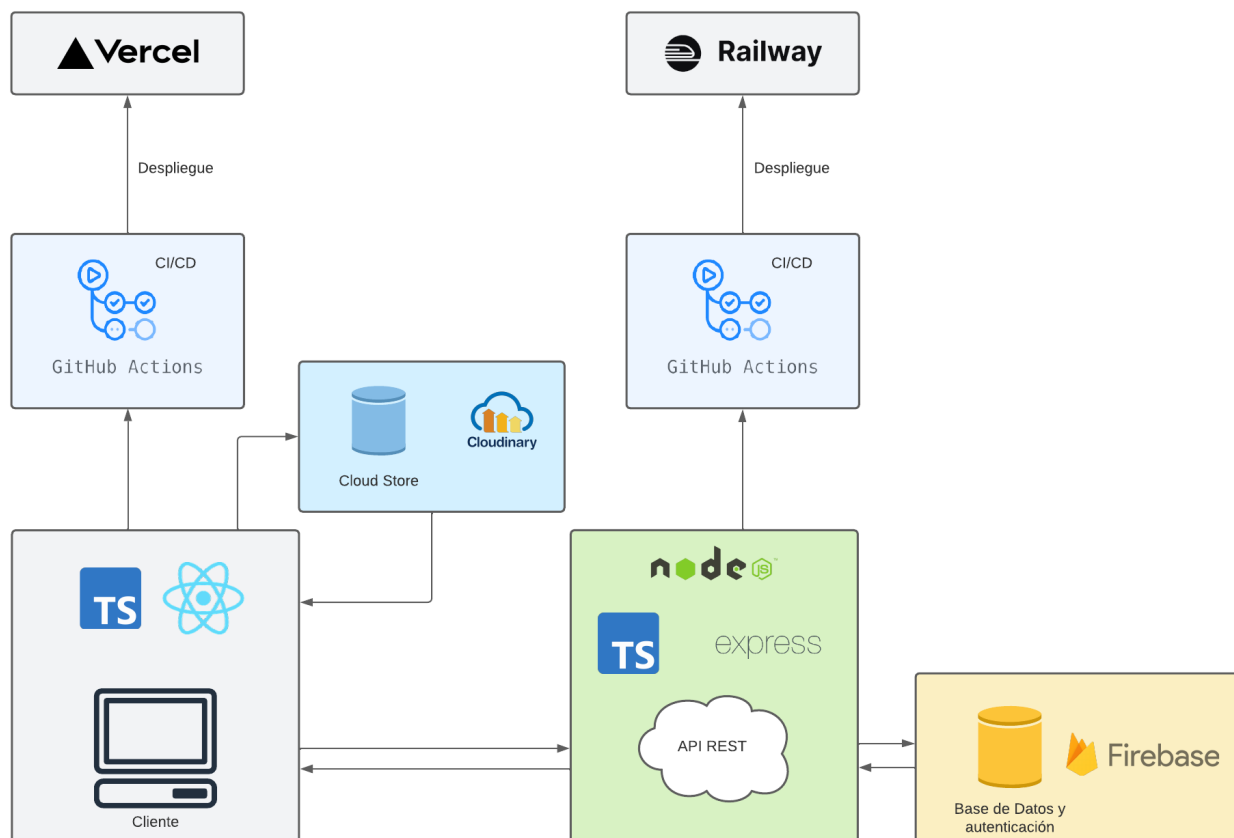
## 2.3 Arquitectura de la aplicación y tecnologías

Antes de empezar a comentar cada una de las partes que componen la aplicación se procederá a explicar la arquitectura general creada, cuyo esquema general puede verse en la Figura 2.1.

Las partes principales y más importantes de la arquitectura son la parte del cliente (*Frontend*), el *Backend* con la API REST y la base de datos. Para empezar el *Frontend* será el que cree toda la interfaz gráfica para el consumo de los usuarios, hará peticiones a la API REST creada, esta será la encargada de resolver esas peticiones solicitadas por el cliente para buscar información en la base de datos y devolvérsela al usuario, también puede crear, modificar o eliminar información. La base de datos será la encargada de guardar toda esta información de manera persistente.

Por otra parte, tenemos en el *Frontend* una conexión a un *cloud store* que se usará para subir las imágenes que añadan los usuarios de la aplicación. Esta conexión funciona de manera que el usuario sube sus imágenes y desde el *Frontend* se envían a la plataforma *cloud* y esta nos devolverá una *URL* de cada una de las imágenes para poder consumirlas de manera sencilla. Estas *URLs* se le enviarán luego a la API REST para guardarlas en la información del usuario o de los servicios en la base de datos.

Finalmente, se encuentran los despliegues tanto del *Frontend* como del *Backend*, lo que servirá para poder hacer pública nuestra web. Los dos despliegues contarán con su *continuous integration* y *continuous delivery/continuous deployment (CI/CD)* correspondiente antes de poder sacar a producción y de esta manera tener un mejor control de posibles errores. Esto es gracias a los tests realizados en ambas partes y si estos tests se pasan correctamente podrá ser desplegado en su plataforma correspondiente.



**Figura 2.1. Arquitectura general de la aplicación**

### 2.3.1 Base de datos

Esta es una de las partes más importantes de la aplicación ya que la base de datos será donde se guarde la información de los usuarios, servicios, opiniones, etc.

Para la base de datos se han barajado diferentes opciones entre las relacionales basadas en tablas y no relacionales basadas en colecciones y documentos. Se ha elegido esta última por el tipo de información que va a manejar la aplicación y la agilidad que nos aporta en el desarrollo.

Algunas de las ventajas que podemos encontrar en las bases de datos no relacionales pueden ser [5]:

- La administración de este tipo de bases de datos es menor y menos compleja.
- Son versátiles, de modo que te permiten hacer cambios de una manera sencilla
- Son escalables ya que se puede aumentar el número de nodos sin interrumpir la usabilidad de la misma.
- Son más económicas debido a que no necesitan grandes recursos.

Respecto a las desventajas en las bases de datos no relacionales se pueden encontrar las siguientes [5]:

- En comparación a las bases de datos relacionales, estas no son tan consistentes y no cumplen con la propiedad de atomicidad.
- No hay un estándar debido a que son bases de datos relativamente nuevas y aún se encuentran en un estado de madurez.

Dentro de las diferentes bases de datos no relacionales se ha usado *Firestore* [9] por varios motivos: su facilidad de uso, las funcionalidades que te proporciona para acceder, eliminar, modificar o crear los datos de una manera sencilla y la posibilidad de poder acceder a través de una interfaz gráfica para así poder ver toda la información y poder gestionarla, lo que hace que puedas cambiar datos rápidamente para los momentos de pruebas en desarrollo.

Otras de las bases de datos no relacionales usadas hoy en día pueden ser *MongoDB*, *DynamoDB*, *Redis*, etc. No se ha usado ninguna de estas ya que, al contrario de *Firestore* [9], requerían una complejidad mayor de uso.

### 2.3.2 Autenticación de usuarios

Los usuarios de la plataforma necesitarán registrarse, iniciar sesión, cerrar sesión, etc, y para esto se usará la herramienta *Authentication* [10] que proporciona *Firebase* [8]. Se usa esta herramienta respecto a otras como *Auth0* debido a que ya se está usando la plataforma *Firebase* [8] para la base de datos y tenemos una mejor integración. También, proporciona muchas funcionalidades para facilitar el desarrollo como pueden ser métodos para la creación, eliminación, inicio de sesión y cierre de sesión de los usuarios.

Por otra parte, como en el caso de *Firestore*, también hay una interfaz gráfica donde se pueden ver todos los usuarios registrados y se puede gestionarlos desde esta misma interfaz.

En este proyecto para la autenticación solo se ha permitido que los usuarios accedan con usuario y contraseña sin necesidad de tener que vincular cuentas de google, apple u otras plataformas a la aplicación.

### 2.3.3 Backend

Para el *Backend* se ha usado una arquitectura *REST* para la comunicación entre cliente y servidor y así recibir, crear, modificar o eliminar información de la base de datos. De esta manera nuestro *Backend* creará una API REST y esta se encargará de hacer de intermediaria entre el cliente y la base de datos.

En el *Backend* se han usado diferentes tecnologías que hoy en día se consideran modernas y con una alta demanda en el mercado.

*NodeJs* [11] se ha usado como entorno de ejecución de *JavaScript* para el *Backend* en vez de otros como pueden ser *Java*, *Python*, etc. y así tener un desarrollo más ágil junto con el *Frontend* y que el cambio de contexto a la hora del desarrollo sea mucho menor.

*TypeScript* como lenguaje de programación con el objetivo de hacer un *software* más robusto a la hora del desarrollo en vez de *JavaScript* ya que es fuertemente tipado, tiene interfaces, etc.

Una vez elegido el lenguaje del *Backend* se ha elegido el *Framework* para desarrollar la API REST, en este caso *ExpressJs* [12] ya que es el más usado actualmente en *NodeJs* [11], con una baja curva de aprendizaje y con una experiencia muy satisfactoria para desarrollar una API REST. Existe otro bastante usado actualmente también como es *NestJs* [13] pero requería un aprendizaje mucho mayor y complejidad de desarrollo.

Para los tests del *Backend* se usa *Jest* [14] debido a que se integra de una manera muy fácil en el desarrollo y tiene una baja curva de aprendizaje. Es un *framework* de testing moderno y en comparación a otros como puede ser *Mocha* junto con *Chai*, *Jest* te ofrece lo mismo y más funcionalidades en un solo *framework*.

### **2.3.4 Frontend**

Esta es una de las partes más importantes hoy en día si quieres vender un producto ya que consiste en la parte visual de nuestro producto, una buena interfaz de usuario y experiencia son imprescindibles para los usuarios. Debe ser una interfaz simple y que sea sencilla de usar ya que a los usuarios cuanto más información les aparezca más difícil les será comprenderla y pueden llegar sobrecargarse o perderse entre tanta información.

Por todo esto se ha optado por una interfaz sencilla y clara donde se podrá ver lo que se necesita a simple vista, ya sea un usuario normal que visita la página web, un cliente que quiera guardar sus servicios favoritos o un proveedor de servicios que quiera añadir nuevos servicios a la plataforma.

Como lenguaje de programación para el desarrollo del *Frontend* se ha usado *TypeScript* [17] en vez de *JavaScript* [18] ya que hará el *software* más robusto gracias a sus funcionalidades como los tipados, interfaces, etc. Este lenguaje de programación es un *superset* de *JavaScript* [18] por lo que tendremos también todas las funcionalidades que nos aporta este.

Actualmente, para el desarrollo *Frontend* lo más común es usar *frameworks* o librerías para tener un desarrollo más ágil, con mejor arquitectura e incluso con mejores prácticas que si no lo usamos.

Algunos de los *framework* más utilizados hoy en día son *React* [19], *Angular* [20] o *Vue* [21], hay muchos más pero estos tres son los principales en la actualidad.

Para este trabajo se ha seleccionado *React* [19] ya que de los comentados anteriormente es el más demandado por las empresas según la encuesta de *Stack Overflow* [4]. Además, ofrece flexibilidad a la hora de usarlo y la curva de aprendizaje es baja, lo cual favorece un desarrollo más rápido del *software* a crear.

Por otro lado, se han usado librerías para complementar el desarrollo en el *Frontend* con *React* [19], algunas de estas librerías son:

- **React Router** [22]: librería para manejar las rutas que creemos de la aplicación y también la navegación entre ellas.
- **React Hook Form** [23]: librería para manejar la información de los formularios, validaciones y errores.

A la hora de pensar en el diseño de la aplicación, sus estilos, colores, etc. suele complicarse para los desarrolladores ya que estos no cuentan con mucho conocimiento sobre diseño. Por esto, existen también las llamadas librerías de componentes que te ofrecen componentes con estilos predefinidos y a su vez con algunas funcionalidades implementadas.

Hoy en día, hay una gran variedad de librerías de componentes y las más conocidas suelen tener las mismas funcionalidades y componentes y la decisión para elegir una respecto a otras suele ser debido a gustos: si los estilos te parecen atractivos, su facilidad en el desarrollo, si son fáciles de configurar o no, etc.

Algunas de las librerías de componentes más usadas hoy en día pueden ser: *MaterialUI* [24], *ChakraUI* [25] y *Bootstrap* [26]. Todas son muy similares entre ellas, ofrecen prácticamente lo mismo y lo que cambia es a la hora de desarrollar con ella y sus estilos.

En este proyecto se ha usado *ChakraUI* [25] ya que había conocimiento previo en esta librería y por gustos se creyó que tenía mejores estilos y diseño de componentes que las otras nombradas anteriormente.

Por otra parte, en el *Frontend* se crearán componentes propios de la aplicación, o componentes básicos personalizados, ya que las librerías de componentes son más generales. Para este caso, hay que darle estilos a estos y se barajaron diferentes opciones como pueden ser CSS nativo o *Tailwind* [27].

Finalmente, se eligió usar *Tailwind* [27] ya que puedes desarrollar de una forma más rápida, no existe la necesidad de tener que crear ficheros para el CSS y también es una herramienta muy usada en la actualidad. Tiene la ventaja de que si tienes conocimientos en CSS será muy sencillo adaptarte ya que simplemente funciona añadiendo estilos en línea a las etiquetas *HTML* con su propia nomenclatura.

Cuando se quieran subir imágenes, estas se subirán a una plataforma *cloud* y así de esta

manera tener el contenido multimedia desacoplado de la base de datos. Para esto se ha usado la herramienta *Cloudinary* [28] ya que provee una gran variedad de funcionalidades para poder modificar las imágenes, se adapta a múltiples lenguajes de programación y permite usarla gratis hasta un límite de 20GB.

Por otra parte, tanto en el *Frontend* como en el *Backend* es importante realizar tests para que nuestro *software* sea menos susceptible a errores. De esta manera, cuando hagamos cambios en el código estos tests determinarán si hay algún error o no. También esto nos servirá para los despliegues de la aplicación y así hacer comprobaciones antes de su despliegue.

Para los test se ha usado la herramienta más conocida hoy en día y más demandada que es *Jest* [14]. Esta herramienta tiene una fácil integración en el proyecto y tiene una curva de aprendizaje muy suave. A su vez esta nos permite hacer una gran cantidad de test gracias a que tienen mucha variedad de aserciones. Otras de las herramientas que se barajaron fueron el framework *Mocha* [15] junto con la librería *Chai* [16] pero se escogió *Jest* [14] debido a que en una solo tecnología se tienen las mismas funcionalidades que las anteriores juntas y más funcionalidades, esto es algo realmente importante ya que no será necesario añadir más dependencias al proyecto.

## 2.4 Implementación

### 2.4.1 Base de datos

Para el diseño de la base de datos de esta aplicación se han creado diferentes colecciones para poder guardar toda la información necesaria de los usuarios, servicios, etc.

Estas colecciones se componen de documentos en los que cada uno de ellos tendrá una estructura y campos fijos. Las colecciones que se han creado son las siguientes:

- **Clients**

Guarda toda la información de los usuarios registrados como clientes ya que es diferente a la de los proveedores de servicios.

Esta colección se compone de documentos que tienen la siguiente estructura y campos:

- **id:** identificador del cliente.
- **name:** nombre del cliente.
- **email:** correo electrónico del cliente.
- **type:** tipo de usuario, en este caso *Client*.
- **FavouriteServices:** *array* con los identificadores de los servicios favoritos.

- **Providers**

Guarda toda la información de los usuarios registrados como proveedores que, en este caso, contendrán más información que en el caso de los clientes.

Esta colección se compone de documentos que tienen la siguiente estructura y campos:

- **id:** identificador del proveedor.
- **name:** nombre del proveedor.
- **company:** nombre de la compañía del proveedor.
- **email:** correo electrónico del proveedor.
- **location:** localización del proveedor
- **type:** tipo de usuario, en este caso *Provider*
- **phoneNumber:** número de teléfono del proveedor.

- **website:** página web del proveedor.
- **servicesIds:** identificadores de los servicios que ha creado el proveedor.
- **urlLogoProvider:** *URL* de la imagen del perfil del proveedor

- **Services**

Guarda la información de los servicios creados por los usuarios proveedores de servicios.

Esta colección se compone de documentos que tienen las siguientes estructuras y campos:

- **id:** identificador del servicio.
- **providerId:** identificador del proveedor que creó el servicio.
- **nameService:** nombre del servicio.
- **description:** descripción del servicio.
- **typeService:** tipo de servicio.
- **urlImagesService:** URLs de las imágenes que el proveedor quiera mostrar sobre su servicio.
- **rate:** valoración general del servicio.
- **totalReviews:** total de opiniones del servicio.

- **Reviews**

Colección donde se guardará toda la información relacionada con las opiniones, valoraciones, etc, sobre los servicios.

Esta colección se compone de documentos que tienen las siguientes estructura y campos:

- **id:** identificador del servicio al que corresponden las opiniones.
- **reviews:** *array* con cada una de las *reviews* que ha hecho un cliente sobre ese servicio. Estas *reviews* individuales tienen los siguientes campos:
  - **id:** identificador de una *review* específica
  - **opinion:** opinión del cliente sobre el servicio.



- **nameClient:** nombre del cliente que realiza la opinión.
- **responseProvider:** respuesta del proveedor al cliente.
- **companyProvider:** nombre de la compañía del proveedor.
- **rate:** puntuación del cliente al servicio.

## 2.4.2 Backend

En el *Backend* de la aplicación se han creado bastantes rutas diferentes para los diferentes tipos de peticiones que se quiera hacer desde el cliente, ya sea para obtener información, crear, etc.

Las rutas tienen una estructura que se compone de dos partes: la primera parte de las *URLs* para las peticiones sería la *URL* base, que puede ser la de nuestro servidor funcionando en local con *http://localhost:4000*, o la *URL* que nos da el servicio donde se despliega añadiendo */api*. Por otro lado, la segunda parte de la *URL* de la petición se compone de los diferentes endpoints, listados en la Tabla 2.2.

**Tabla 2.2. Endpoints API REST**

Endpoint	Tipo de petición	Descripción
<b>Usuarios</b>		
<b>/users/{id}</b>	<b>GET</b>	Recibe un identificador de un cliente o de un proveedor y comprobará si existe este usuario en la plataforma. Un caso de uso es en el inicio de sesión.
<b>Clientes</b>		
<b>/clients/new-client</b>	<b>POST</b>	Se le pasará por el <i>body</i> un objeto con la información del cliente.
<b>/clients</b>	<b>GET</b>	Devolverá a todos los clientes existentes.
<b>/favourites/{idClient}</b>	<b>GET</b>	Se le pasará como parámetro el identificador del cliente y devolverá todos los servicios favoritos de este.

Endpoint	Tipo de petición	Descripción
<b>/favourites/{idClient}-{idService}</b>	<b>POST</b>	Recibirá por parámetros el identificador del cliente y el identificador del servicio. Luego comprobará si el cliente tiene ese servicio como favorito o no, si lo tiene lo eliminará de sus favoritos y si no lo tiene lo añadirá.
<b>Proveedores</b>		
<b>/providers/new-provider</b>	<b>POST</b>	Se le pasará por el body un objeto con la información del proveedor.
<b>/providers</b>	<b>GET</b>	Devolverá a todos los proveedores existentes.
<b>Servicios</b>		
<b>/services</b>	<b>GET</b>	Devolverá todos los servicios creados por los proveedores.
<b>/services/{idService}</b>	<b>GET</b>	Recibe un identificador del servicio como parámetro y este será el que devuelva.
<b>/services/{idService}</b>	<b>DELETE</b>	Recibe un identificador del servicio como parámetro y eliminará este servicio
<b>/services/search/{input}</b>	<b>GET</b>	Recibe un texto como parámetro y buscará si hay servicios que contengan este texto en el nombre y los devolverá.
<b>/services/provider/{idProvider}</b>	<b>GET</b>	Recibe un identificador de un proveedor como parámetro y buscará todos los servicios creados por este proveedor y los devolverá.
<b>/services/new-service</b>	<b>POST</b>	Recibe un objeto con la información de un servicio por el body de la petición y lo creará.
<b>Opiniones y valoraciones</b>		
<b>/reviews/{serviceId}</b>	<b>GET</b>	Recibe un identificador de un servicio y devuelve todas las reviews que se han creado para este.

Endpoint	Tipo de petición	Descripción
<b>/reviews/newReview/{service Id}</b>	<b>POST</b>	Recibe un identificador de un servicio y un objeto con la información de una nueva opinión por el body de la petición. Esta comprobará si existe, si es así se modificará la información y si no se creará la nueva opinión.

Después de ver todas las rutas creadas para la API REST en el *Backend*, un ejemplo de *URL* completa para una petición desde un cliente trabajando en local podría ser la siguiente:

- <http://localhost:4000/api/clients/new-client>

Se puede ver cómo sería una petición desde el *Frontend* hacia esta ruta. Esta función recibe una información con los datos del cliente y será lo que se envíe por el *body* de la petición.

```
const fetchPostNewClient = (data: Client) => {
  fetch("http://localhost:4000/api/clients/new-client", {
    method: 'POST',
    body: JSON.stringify(data),
    headers: { 'Content-type': 'application/json; charset=UTF-8' }
  })
  .then(response => response.json())
  .then(json => console.log(json))
  .catch(err => console.log(err))
}
```

Un ejemplo de una petición con una respuesta interesante podría ser la siguiente:

- <http://localhost:4000/api/clients/VfldWYULkBQP7wdIIWLXiR8q43o1>

Esta petición nos devolverá la información de un cliente en concreto como podemos ver en la siguiente imagen ya que le hemos pasado el identificador de este.

```
{
  "favouriteServices": [
    "l4jui7rrtghsr1egex",
    "l4jtfk7gmmichabf1n"
  ],
  "name": "vane",
  "email": "vane@fixadvisor.com",
  "id": "VfIdWYULkBQP7wd1lWLXiR8q43o1",
  "type": "Client"
}
```

### 2.4.3 Frontend

La estructura de ficheros para esta parte del proyecto se basa en una arquitectura limpia de separar en capas. De esta manera tendremos separada las diferentes partes del *Frontend* como pueden ser las peticiones, componentes de la aplicación, páginas, modelos de los datos, etc.

La estructura final de carpetas dentro del *Frontend* es la siguiente:

- .github/workflows: directorio para la configuración de *GitHub Actions* [7].
- src/models: se encontrarán los modelos de los datos que vamos a utilizar en la aplicación a lo largo de su desarrollo.
- src/componentes: directorio donde se crearán todos los componentes de la aplicación.
- src/pages: en este directorio se encontrarán las páginas por las que navegaremos en la aplicación.
- src/context: directorio para crear los contextos de la aplicación como puede ser el contexto del usuario.
- src/firebase: configuración de *Firebase* [8] para poder comunicarse y usar las funcionalidades necesarias.
- src/cloudinary: configuración de *Cloudinary* [28] para poder comunicarse y usar las funcionalidades necesarias.
- src/services: directorio donde se están las funciones que realizan las peticiones a la API REST.
- src/utills: directorio con funcionalidades comunes y útiles en todo el proyecto.

- `src/tests`: directorio donde se encuentran los tests de la aplicación.

## Gestión del contenido multimedia

Para el contenido multimedia como se ha comentado anteriormente, se usa la plataforma *Cloudinary* [28] como servicio *cloud* para subir las imágenes que los usuarios quieran subir.

En este caso cuando un usuario suba una imagen desde el *Frontend* se hará una petición a esta plataforma para subir el contenido. Una vez se suba, la petición nos devolverá una *URL* para cada imagen. Una vez se tengan estas *URLs* se juntará con la información del usuario o del servicio y se hará la petición correspondiente a la API REST para que esta introduzca toda la información en la base de datos.

Finalmente cuando se quieran usar estas imágenes, simplemente usando estas *URLs* que obtendremos al hacer una petición ya podremos verlas en nuestra interfaz.

## Vistas principales de la aplicación

A lo largo de este apartado se verán todas las vistas creadas para la aplicación y cómo funciona cada una de ellas según cada usuario.

El menú de navegación se verá que cambia si es usuario cliente o proveedor de servicios. En el caso de que no se haya iniciado sesión tendremos un menú solo con la opción de *login*. Figura 2.2.



**Figura 2.2. Barra menú sin iniciar sesión**

Si se ha iniciado sesión como usuario cliente tendremos un menú como se ve en la Figura 2.3. y como usuario proveedor se verá como demuestra la Figura 2.4.



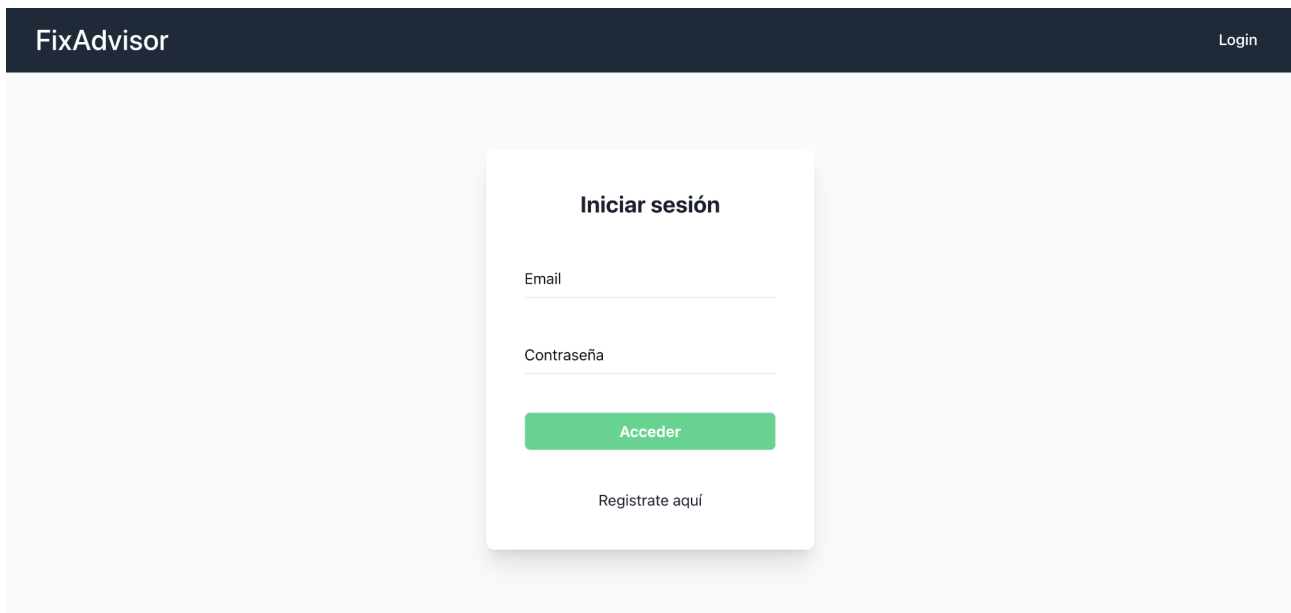
**Figura 2.3. Barra menú sin iniciar sesión**



**Figura 2.4. Barra menú sin iniciar sesión**

Esto es debido a que cada tipo de usuario tendrá funcionalidades diferentes en la aplicación ya que su objetivo dentro de ella no es el mismo, los clientes quieren consultar servicios y los proveedores crearlos.

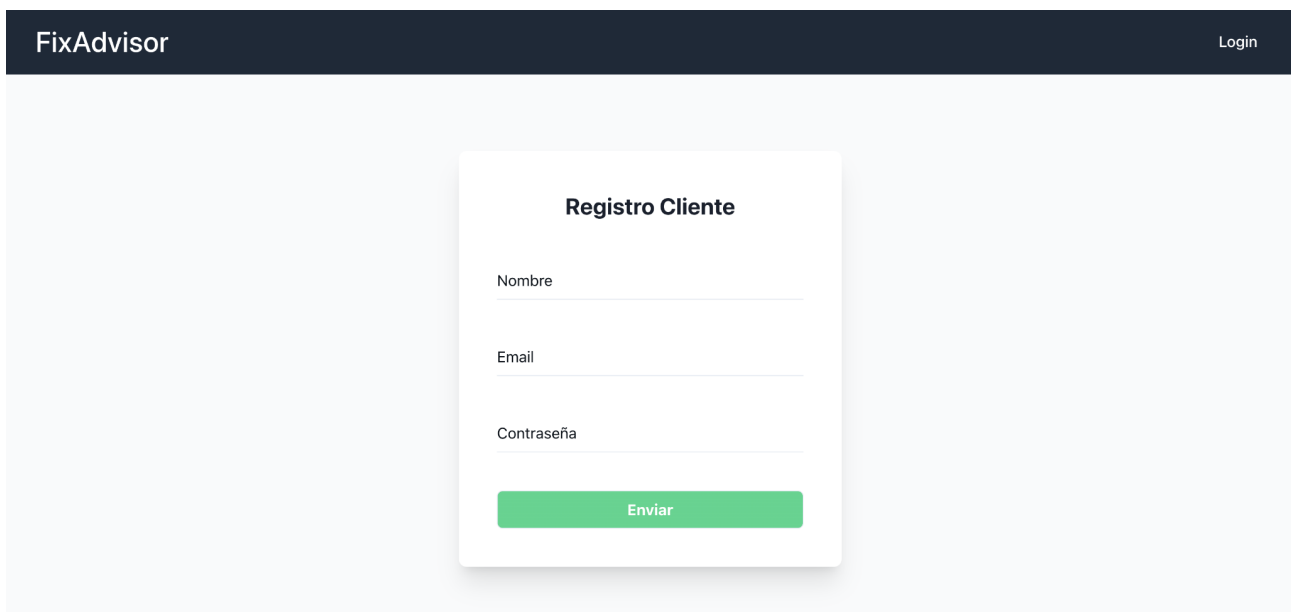
Si un usuario quiere iniciar sesión, tendrá un formulario donde rellenará con usuario y contraseña con la que se haya registrado en su momento. Figura 2.5.



**Figura 2.5. Página de inicio de sesión**

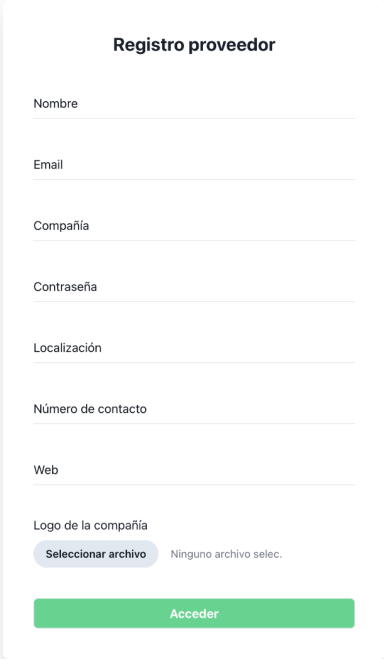
En caso de que no se hayan registrado en la plataforma tendrán la opción de registrarse, donde les solicitará que elijan si quieren registrarse como cliente o como proveedores.

Si se elige usuario cliente les llevará a un formulario en el que se les pedirá el nombre, email y una contraseña para poder acceder. Figura 2.6.



**Figura 2.6. Página de registro de un cliente**

Si el usuario quiere registrarse como proveedor de servicios, esto abrirá un formulario un poco más extenso donde tendrá que rellenar varios campos sobre su compañía. Figura 2.7.



**Registro proveedor**

Nombre

Email

Compañía

Contraseña

Localización

Número de contacto

Web

Logo de la compañía

Seleccionar archivo Ninguno archivo selec.

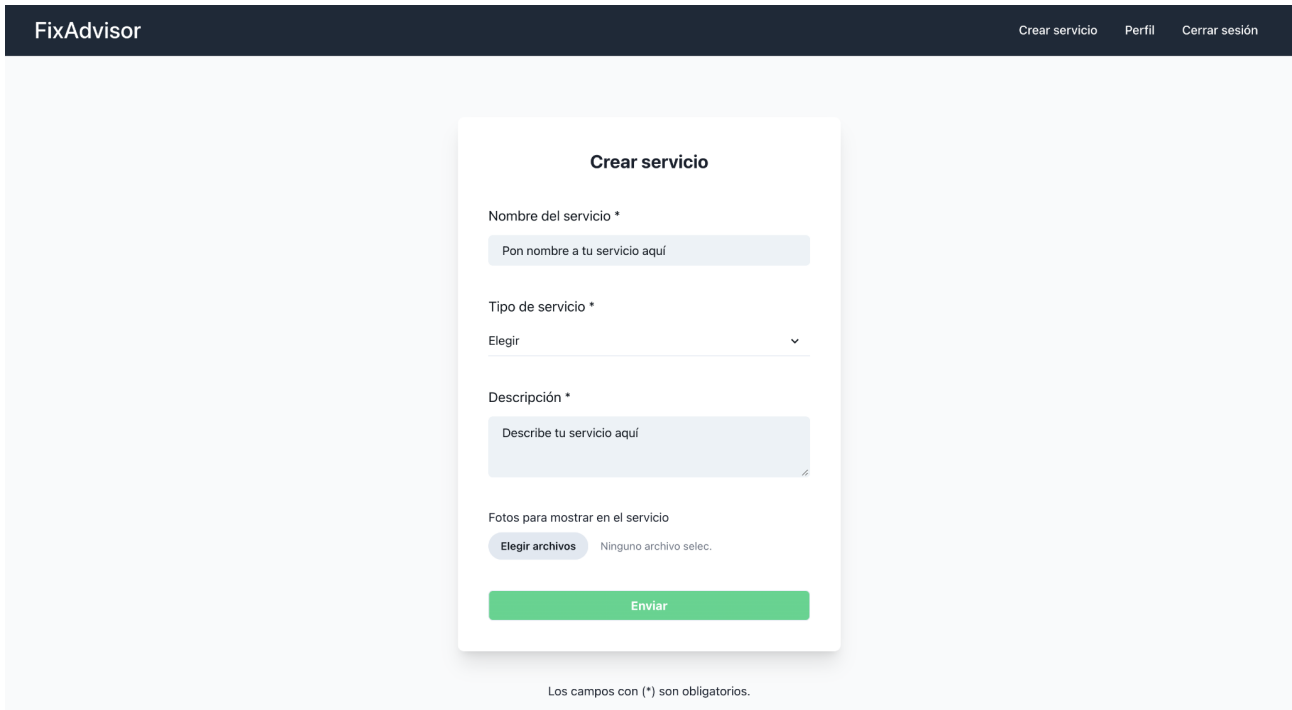
Acceder

**Figura 2.7. Página de registro de proveedor**

En el caso del registro como proveedor rellena varios campos como pueden ser el nombre, email, nombre de la compañía, contraseña para poder acceder, localización para informar al usuario, número de contacto, página web y por último una foto de un logo o imagen de la compañía.

Toda esta información es solamente información del proveedor, algunos datos para simplemente el uso de la aplicación e información sobre la compañía y otros datos para poder darle más información al usuario sobre este proveedor.

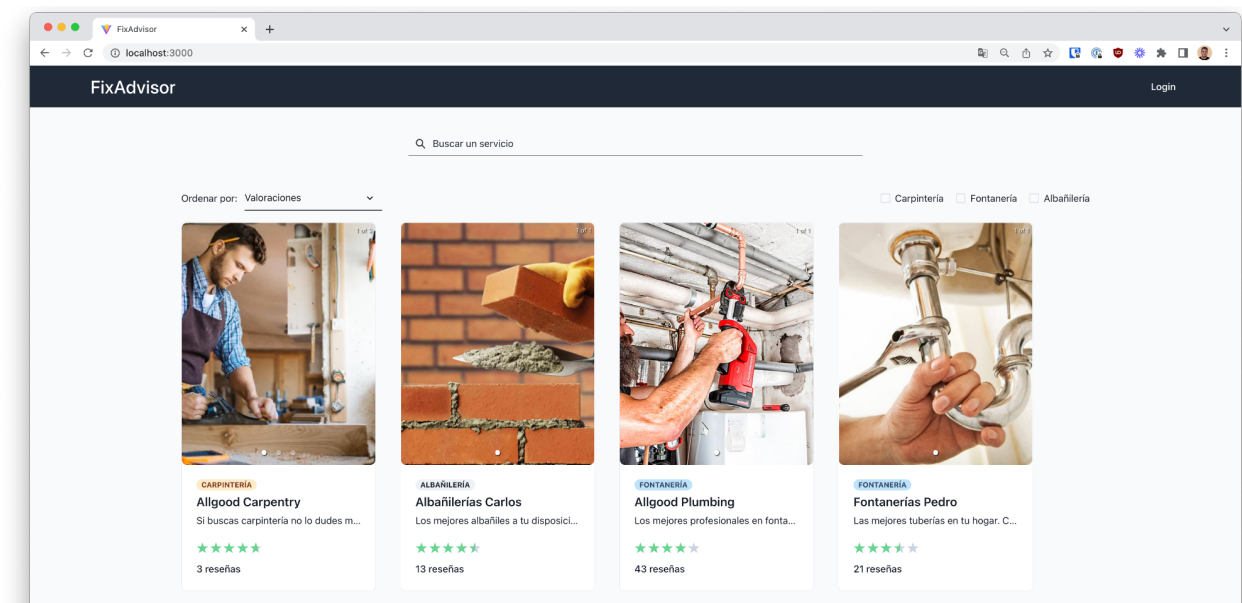
Si un proveedor quiere crear un servicio, accederá a una página con un formulario en concreto para la creación de este. En este se le preguntará sobre el nombre del servicio, tipo, una descripción y que pueda subir fotos sobre servicios que haya realizado y quiera que los usuarios vean. Figura 2.8.



**Figura 2.8. Página creación de servicio**

En la vista principal de la aplicación se podrán ver todos los servicios a modo de tarjetas tanto los usuarios registrados como los que no lo están. A primera vista tendrás un buscador de manera que se puede buscar por nombre de los servicios y así se podrá encontrar directamente el que más interese.

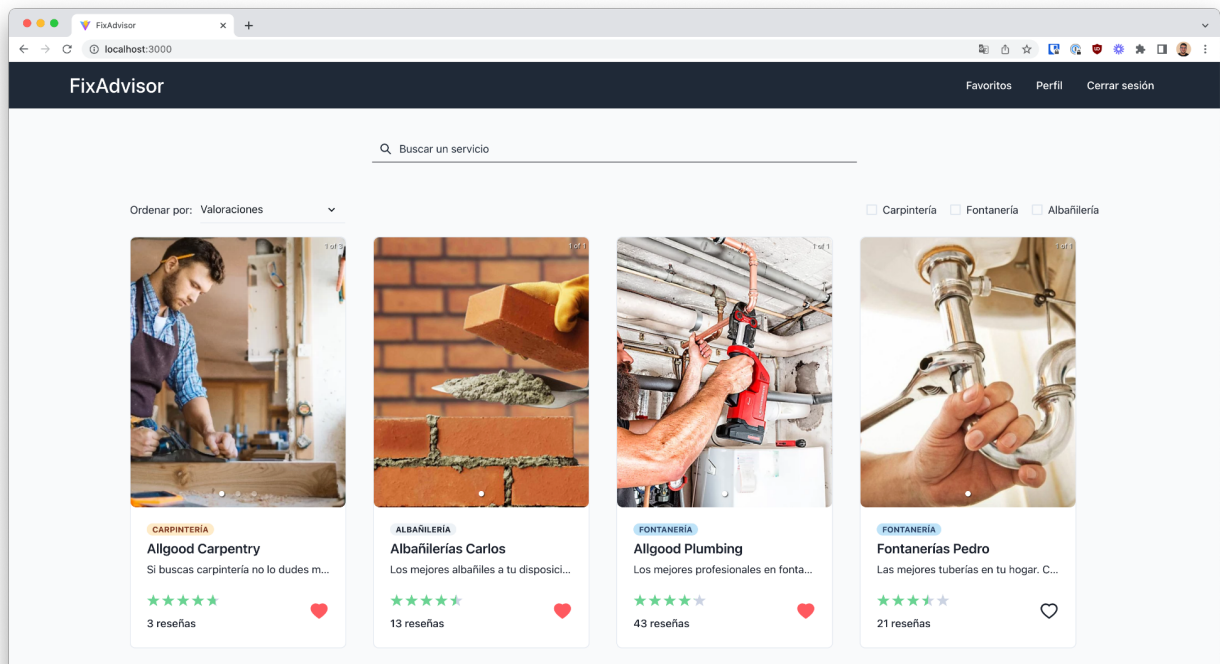
Por otra parte se podrán ordenar estos servicios por nombre, valoraciones o tipo de servicios. También la aplicación te ofrece un filtrado de los servicios por si solamente estás interesado en los servicios de un tipo en concreto. Figura 2.9.



**Figura 2.9. Página principal**

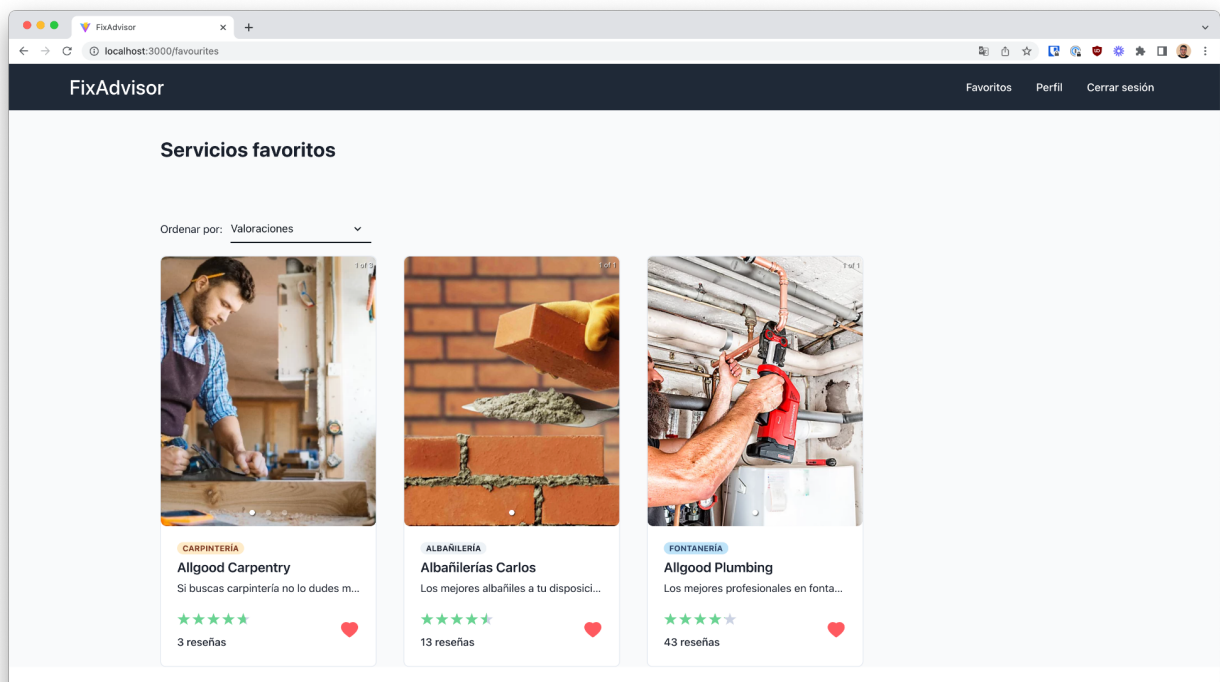


En el caso de que un usuario cliente haya iniciado sesión tendrá la misma vista pero estas tarjetas tendrán la opción de marcar como favorito el servicio y así añadir a la sección de favoritos del cliente. Figura 2.10.



**Figura 2.10. Página principal de un usuario cliente**

Los clientes tendrán la opción de favoritos en el menú de navegación, en este podrán ver todos los servicios a los que le han dado favorito como se comentó anteriormente. Figura 2.11.



**Figura 2.11. Página de servicios favoritos**

Si entramos en uno de los servicios se podrá ver en más detalle su información. Esta página contendrá información sobre la valoración de este servicio, descripción, las fotos de servicios ya realizados que haya subido el proveedor, información del proveedor como puede ser el nombre, contacto y página web. Figura 2.12.

También se podrán ver las opiniones que han hecho otros usuarios clientes sobre este servicio y a su vez las respuestas del proveedor a estas reseñas. Solamente los usuarios clientes son los que pueden dar su opinión sobre un servicio.

Dentro de las opiniones tenemos una opción que es para añadir un comentario que por defecto aparece cerrada. Figura 2.13.

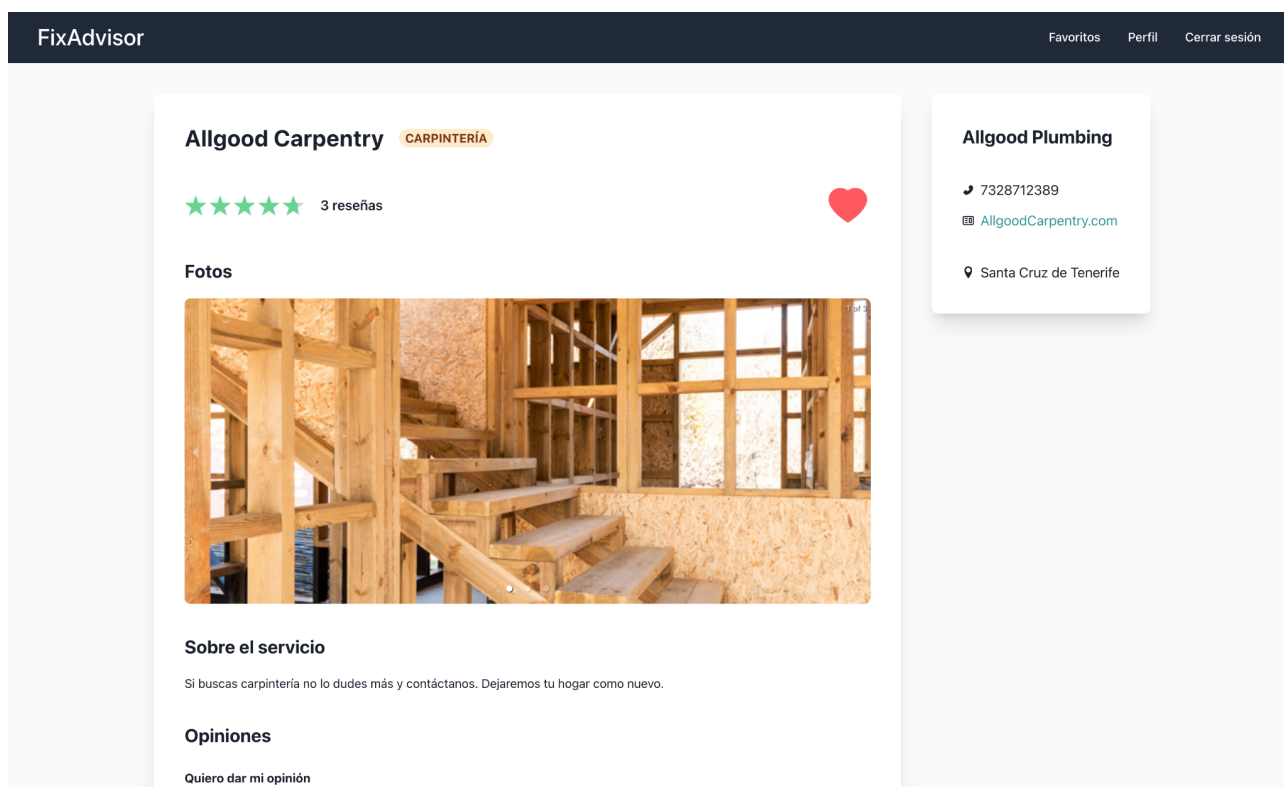


Figura 2.12. Página detalles de un servicio

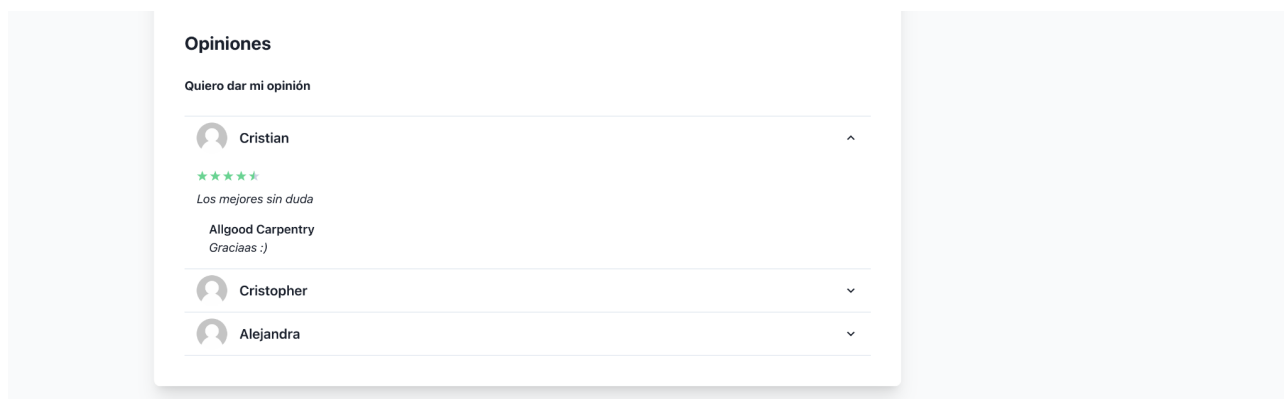
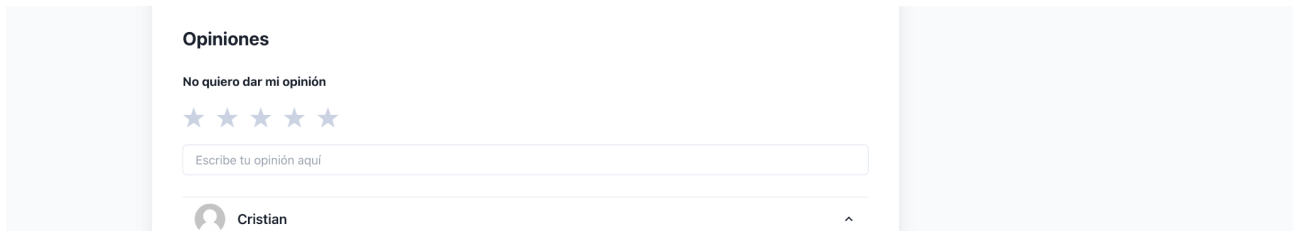


Figura 2.13. Página detalles de un servicio - opiniones

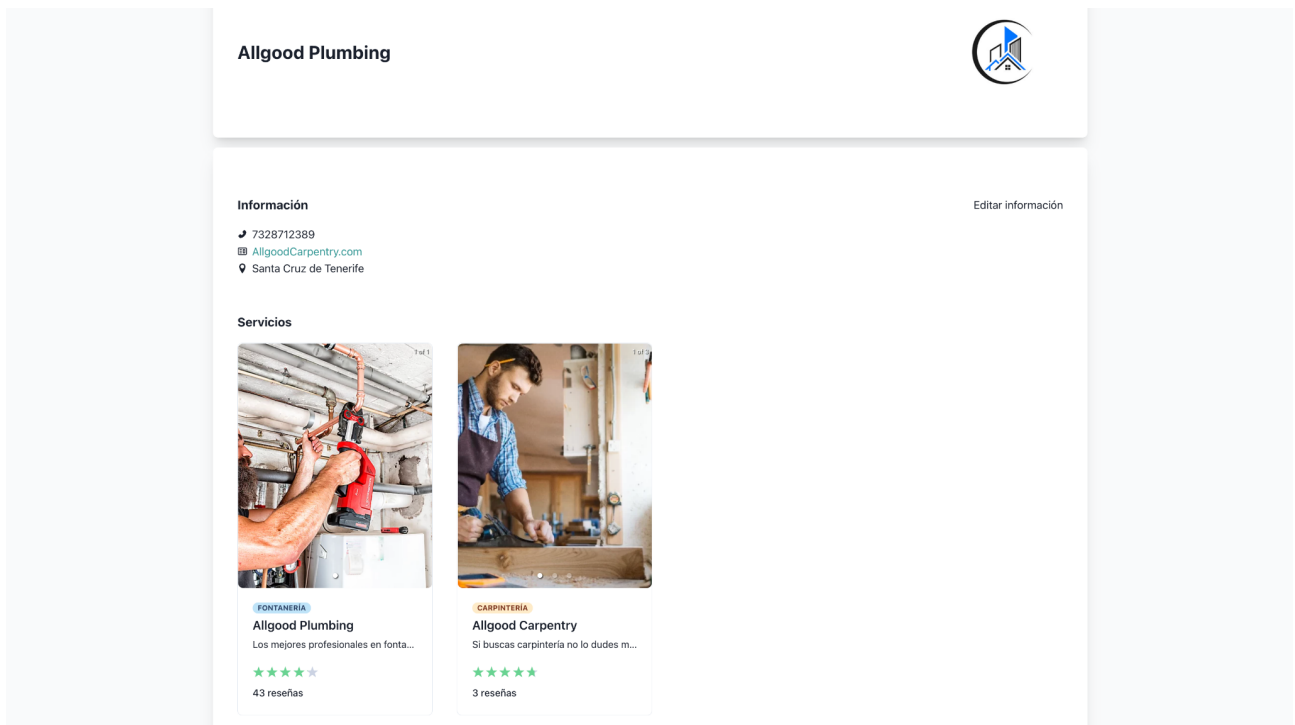
Si el usuario quiere dar su opinión solo tendrá que abrir la opción de “quiero dar mi opinión” y se abrirá esta opción que tendrá un apartado para escribir y también puntuar el servicio mediante las estrellas. En caso de arrepentirse también se podrá cerrar. Figura 2.14.



**Figura 2.14. Crear opinión sobre servicio**

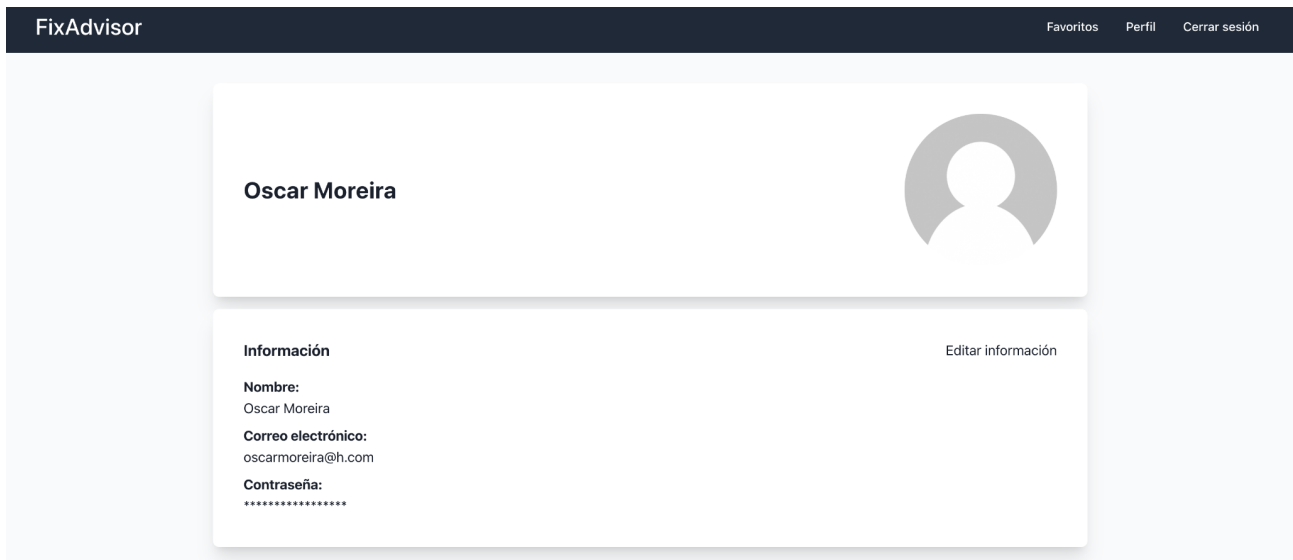
Por otra parte tenemos también la página del perfil de los usuarios donde podrán ver su información o editarla. Esta será adaptada según si es un usuario cliente o proveedor.

En el caso del proveedor se le mostrará su información y la opción de editarla. Por otra parte también se muestran sus servicios creados. Figura 2.15.



**Figura 2.15. Perfil usuario proveedor**

Para el usuario cliente le mostrará su información también para poder cambiarla si lo necesita aunque sea menos que la del proveedor. Figura 2.16.



**Figura 2.16. Perfil usuario cliente**

## 2.5 Despliegue

Hoy en día, en un proyecto de *software* es muy importante implementar prácticas de integración continua y automatizar procesos. En los despliegues de las aplicaciones es donde más se pueden ver este tipo de prácticas.

En el caso de este trabajo se hace tanto para el *Backend* como para el *Frontend* y de la misma manera. Para esto se está usando *GitHub* [6] como repositorio de código y así poder tener un copia de seguridad del código donde podemos ver el control de versiones que hemos hecho.

*GitHub* [6] nos provee una herramienta llamada *GitHub Actions* [7] que nos permite gestionar estos despliegues y poder ejecutar acciones cuando se le indique. Estas *actions* se pueden ejecutar cuando se haga un suba un cambio al repositorio, cuando se haga una *pull request*, o con una gran variedad de opciones más pero estas son las más relevantes para el desarrollo.

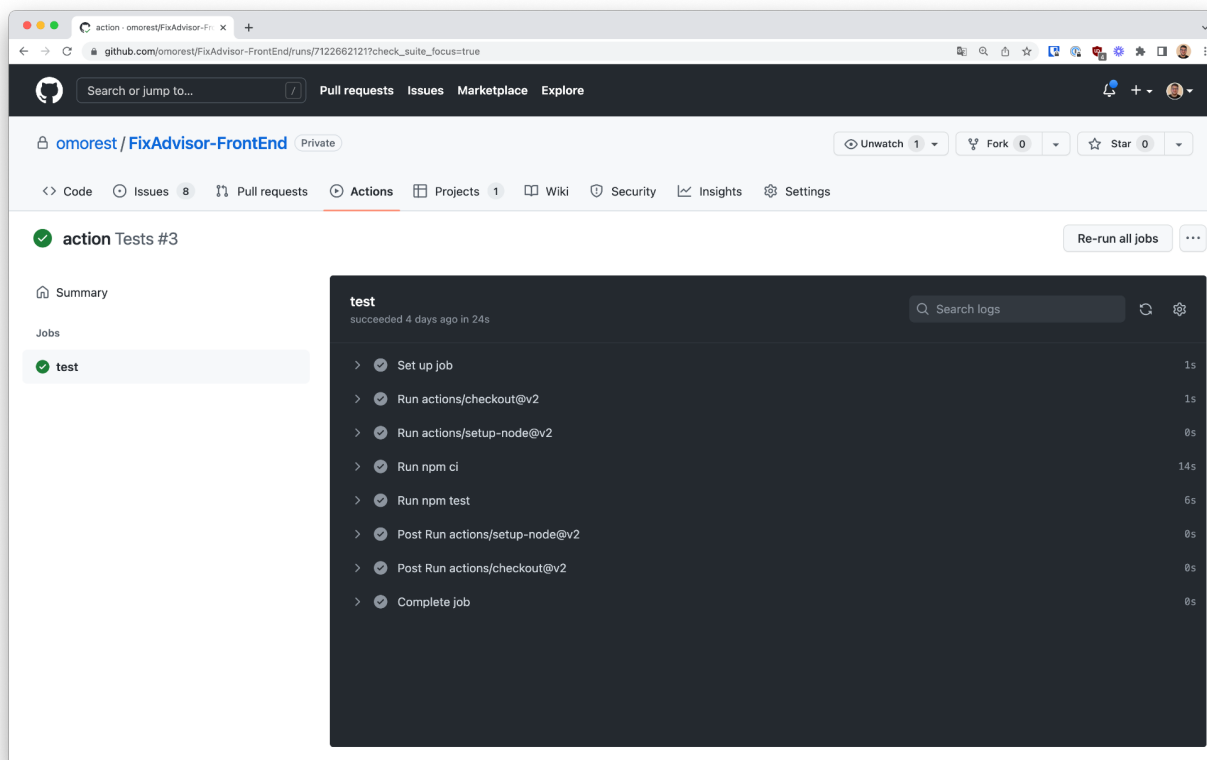
En el proyecto se usa esta herramienta para que siempre que se suba un cambio a la rama *dev* o una *pull request* a la rama *main* se ejecute una *GitHub Action* [7] que ejecutará los tests del código que se han creado, si esta se ejecuta correctamente se podrá subir el código o mezclar en el caso de la *pull request*.

Para crear estas *actions* en nuestro proyecto debemos se ha creado una directorio *.github/workflows/* y dentro de este se ha creado un fichero *action.yml* que contiene la configuración de la action. Este fichero contiene el código.

```
name: Tests
on:
  push:
    branches: [ main, dev ]
  pull_request:
    branches: [ main ]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: actions/setup-node@v2
        with:
          node-version: '14'
      - run: npm ci
      - run: npm test
```

Podemos ver el ejemplo de una *GitHub Action* [7] ejecutada correctamente en *GitHub* [6] en la Figura 2.17



**Figura 2.17 Ejecución GitHub Action**

Como se ha comentado anteriormente, se ha hecho lo mismo tanto para el *Backend* como para el *Frontend* ya que aunque sean áreas de desarrollo distintas los procesos de despliegue y automatización de tests son similares.

Tanto el *Frontend* como el *Backend* serán desplegados en sus plataformas correspondientes para así poder acceder a la página desde cualquier sitio.

El *Frontend* se despliega en una plataforma llamada Vercel [30], ya que nos facilita la integración con nuestro proyecto pudiendo seleccionar el repositorio de GitHub [6] que se quiera desplegar. Esta herramienta sólo permite actualmente subir proyectos con tecnologías de Frontend. También nos permite añadir variables de entornos para la configuración del proyecto.

Por otra parte el *Backend* se despliega en *Railway* [29], esta plataforma es similar a Vercel [30] pero para proyectos de *Backend* donde te provee un servidor. También cuenta con integración con GitHub [6] y configuración de variables de entorno para el proyecto.

- URL del proyecto desplegado: <https://fixadvisor.vercel.app/>

# Capítulo 3

## Conclusiones y líneas futuras

Tal y como se ha podido comprobar en este proyecto, las plataformas para buscar información sobre servicios y empresas se han convertido en una herramienta imprescindible a la hora de elegir dónde viajar, dónde comer o qué servicios contratar.

Este trabajo se ha centrado en este último punto, más concretamente en los servicios prestados por profesionales de la albañilería, carpintería, fontanería, etc. Se ha desarrollado una aplicación que agrupe a estos profesionales para que puedan ofertar sus servicios y los potenciales clientes puedan acceder a estos, conocer información y dejar reseñas.

A la hora de elaborar este trabajo se han encontrado muchos aspectos positivos y también se han encontrado algunas dificultades. Entre los positivos se encuentra la oportunidad de poder desarrollar un proyecto *Fullstack* completo desde cero. También el aprender y utilizar tecnologías modernas muy demandadas actualmente, como las ya mencionadas anteriormente. Otro aspecto ha sido aprender a gestionar un proyecto de desarrollo, estimando la dificultad y el tiempo dedicado a cada tarea, entregas, prioridades, etc. Finalmente, se ha aprendido a ofrecer una solución que no se ha encontrado aún en el mercado actualmente.

Por otro lado, se han encontrado varias dificultades, entre ellas elaborar este proyecto como un Trabajo de Fin de Grado ya que es un trabajo en el que constantemente van surgiendo nuevas ideas para añadir a la plataforma, lo que puede provocar que se retrasen los tiempos estimados desde un principio. Otra complicación es la preparación previa antes del desarrollo del proyecto, es decir, se han tenido que implementar tecnologías hasta ahora desconocidas, lo que implica que haya que dedicar tiempo para su aprendizaje.

Como se ha comentado anteriormente, este es un proyecto que permite agregar nuevas características continuamente. Algunas de estas podrían ser:

- Tener la posibilidad de que un usuario suba fotos del servicio que ha contratado a las reseñas que haga.
- Los usuarios pueden indicar si un comentario ha sido útil.
- Permitir que los clientes hagan listas en su sección de favoritos para tener los servicios organizados.
- Implementar mapas en la aplicación con dos objetivos:
  - Los proveedores pueden poner la ubicación de sus servicios.

- El usuario puede ver en el mapa los servicios cercanos.
- Que los proveedores puedan añadir un rango máximo para definir su zona de trabajo.
- Posibilidad de modificar la información de los servicios creados por el proveedor.
- Permitir a los proveedores eliminar sus servicios
- Añadir más tipos de servicios de reparaciones o arreglos en el hogar.
- Soporte para diferentes idiomas.
- Sistema recomendador cuando la aplicación crezca a nivel de usuarios y tenga más información.



# Capítulo 4

## Conclusions and future works

As this project has shown, platforms to search for information on services and companies have become an essential tool when choosing where to travel, where to eat or what services to hire.

This work has focused on this last point, more specifically on the services provided by professionals in masonry, carpentry, plumbing, etc. An application has been developed that groups these professionals so that they can offer their services and potential clients can access them, find out information and leave reviews.

At the time of elaborating this work, many positive aspects have been found and some difficulties have also been found. Among the positives is the opportunity to develop a complete Fullstack project from scratch. Also learning and using modern technologies that are currently in high demand, such as those already mentioned above. Another aspect has been learning to manage a development project, estimating the difficulty and time spent on each task, deliverables, priorities, etc. Finally, it has learned to offer a solution that has not yet been found on the market today.

On the other hand, several difficulties have been encountered, including preparing this project as a Final Degree Project since it is a project in which new ideas are constantly emerging to add to the platform, which can cause time delays estimated from the beginning. Another complication is the prior preparation before the development of the project, ergo, until now unknown technologies have had to be implemented, which implies that time must be dedicated to learning them.

As mentioned before, this is a project that allows new features to be added continuously. Some of these could be:

- Having the possibility for a user to upload photos of the service they have contracted to the reviews they make.
- Users can indicate whether a comment has been helpful.
- Allow customers to make lists in their favorites section to have the
- Implement maps in the application with two objectives:
  - Providers can put the location of their services.
  - The user can see nearby services on the map.
- Providers can add a maximum range to define their work area.

- Possibility of modifying the information of the services created by the provider.
- Allow providers to remove their services
- Add more types of home repair or home improvement services.
- Support for different languages.
- Recommender system when the application grows at the user level and has more information.

# Capítulo 5

## Presupuesto

El presupuesto del proyecto se basa en las horas dedicadas al análisis y las diferentes partes del desarrollo.

**Tabla 5.1: Presupuesto por trabajo realizado**

Tarea	Horas	Coste (35€/h)
Análisis del proyecto	24	840 €
Arquitectura aplicación	16	560 €
Modelado Base de datos	50	1750 €
Backend	150	5250 €
Frontend	180	6300 €
Despliegue	16	560 €
Documentación	16	560 €
<b>Total</b>	<b>452</b>	<b>15820 €</b>

# Bibliografía

- [1] Página web de TripAdvisor. Disponible en: <https://www.tripadvisor.es/>. Accedido el 28/01/2022.
- [2] Documentación Google Reviews. Disponible en: <https://support.google.com/maps/answer/6230175?hl=en&co=GENIE.Platform%3DDesktop>. Accedido el 28/01/2022.
- [3] Página web de Yelp. Disponible en: <https://www.yelp.es>. Accedido el 28/01/2022.
- [4] Stack Overflow Survey. Most popular technologies web frameworks. Disponible en: <https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks>. Accedido el 3/07/2022.
- [5] Las 5 principales ventajas y desventajas de bases de datos relacionales y no relacionales: NoSQL vs SQL. Autor: Guido Cutipa. Disponible en: <https://guidocutipa.blog.bo/principales-ventajas-desventajas-bases-de-datos-relacionales-no-relacionales-nosql-vs-sql/>. Accedido el 03/07/2022
- [6] Página web de Github. Disponible en <https://github.com/>. Accedido el 2/02/2022.
- [7] Página web Github Actions. Disponible en: <https://github.com/features/actions>. Accedido el 30/06/2022.
- [8] Página web de Firebase. Disponible en: <https://firebase.google.com/>. Accedido el 10/02/2022.
- [9] Documentación Firestore. Disponible en: <https://firebase.google.com/docs/firestore>. Accedido el 10/02/2022.
- [10] Documentación Firebase Authentication. Disponible en: <https://firebase.google.com/docs/auth>. Accedido el 10/02/2022.
- [11] Página NodeJs. Disponible en: <https://nodejs.dev/>. Accedido el 7/02/2022.
- [12] Página ExpressJs. Disponible en: <https://expressjs.com/>. Accedido el 7/02/2022.
- [13] NestJs. Disponible en: <https://nestjs.com/>. Accedido el 6/02/2022.
- [14] Página Jest. Disponible en: <https://jestjs.io/>. Accedido el 15/04/2022.
- [15] Página Mocha . Disponible en: <https://mochajs.org/>. Accedido el 15/04/2022.
- [16] Página Chai . Disponible en: <https://www.chaijs.com/>. Accedido el 15/04/2022.

- [17] Página TypeScript. Disponible en: <https://www.typescriptlang.org/>. Accedido el 7/02/2022.
- [18] Página JavaScript. Disponible en: <https://www.javascript.com/>. Accedido el 7/02/2022.
- [19] Página React. Disponible en: <https://reactjs.org/>. Accedido el 11/02/2022.
- [20] Página Angular. Disponible en: <https://angular.io/>. Accedido el 11/02/2022.
- [21] Página Vue. Disponible en: <https://vuejs.org/>. Accedido el 11/02/2022.
- [22] Página React Router. Disponible en: <https://reactrouter.com/>. Accedido el 20/02/2022.
- [23] Página React Hook Form. Disponible en: <https://react-hook-form.com/>. Accedido el 18/03/2022.
- [24] Página MaterialUI . Disponible en: <https://mui.com/>. Accedido el 13/02/2022.
- [25] Página ChakraUI. Disponible en: <https://chakra-ui.com/>. Accedido el 13/02/2022.
- [26] Página Bootstrap . Disponible en: <https://getbootstrap.com/>. Accedido el 13/02/2022.
- [27] Página Tailwind. Disponible en: <https://tailwindcss.com/>. Accedido el 13/02/2022.
- [28] Página Cloudinary . Disponible en: <https://cloudinary.com/documentation>. Accedido el 29/04/2022.
- [29] Página Railway . Disponible en: <https://railway.app/>. Accedido el 15/05/2022.
- [30] Página Vercel . Disponible en: <https://vercel.com/>. Accedido el 15/05/2022.