INSTITUTO TECNOLÓGICO DE BUENOS AIRES – ITBA

ESCUELA DE INGENIERÍA Y GESTIÓN

HomeFix

AUTORES: Lifschitz, Gastón (Leg. Nº 58225)

Banfi, Micaela (Leg. Nº 57293)

TUTOR: Debrouvier, Hemilse

TRABAJO FINAL PRESENTADO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN INFORMÁTICA

Lugar: Buenos Aires
Fecha: 9-11-2022

Índice

Abstract

En el proceso de contratar proveedores de servicios hogareños como jardineros, plomeros o electricistas, suele existir incertidumbre e inseguridad sobre la calidad y confianza del proveedor. Debido a esto, suele optarse por seguir recomendaciones de conocidos. Este proyecto describe a **HomeFix**, una aplicación web destinada a consorcios, comunidades y grupos de vecinos para facilitar la búsqueda de proveedores de servicios para el hogar, por medio de reseñas basadas en experiencias previas que brinden confianza y tranquilidad a quienes contratan un servicio para su hogar, y mayor posibilidad de concretar contrataciones a proveedores con buena calidad de servicio.

1 Introducción

Cuando uno necesita contratar un servicio de mantenimiento para solucionar problemas del hogar, realiza una exhaustiva búsqueda hasta encontrar un proveedor de confianza. En la gran mayoría de los casos, el proveedor contratado es recomendado por conocidos a través de una red de comunicación informal como grupos de WhatsApp o el famoso "boca en boca".

HomeFix busca conectar a los proveedores de servicios hogareños con quienes lo necesitan y no saben dónde encontrarlo. La misión de esta propuesta es mejorar el problema de la confianza en la calidad de los servicios hogareños brindados utilizando experiencias previas de otros usuarios.

La aplicación proveerá, por un lado, seguridad al usuario final a la hora de contratar servicios para su hogar y, por el otro, más presencia en internet a los proveedores para facilitar nuevas contrataciones, fomentando a los proveedores a ser más competitivos, para obtener mejores reseñas.

2 Estado del arte

2.1 Problema a resolver

Hoy en día, la referencia para servicios del hogar se realiza principalmente por medio del "boca en boca", que se basa en recomendaciones de familiares, amigos o conocidos. El beneficio de este método es la obtención de información de la calidad del servicio previo a la contratación.

Si bien existen diferentes alternativas que intentan fomentar estas conexiones, no existe una solución que funcione como una red social para personas de una misma comunidad que tenga como fin la promoción de servicios para el hogar. Al momento de dejar ingresar un proveedor al hogar, las personas suelen preferir la recomendación de un vecino, más que la de un completo desconocido en internet, ya que podría ser falsa o poco precisa.

2.2 Alternativas existentes

Si bien el proyecto en cuestión es diferente a sus posibles alternativas debido a la implementación del concepto de *grupos de vecinos* (en la sección Alcance del proyecto se explicará este término), hoy en día hay varias alternativas para conseguir mano de obra en distintas plataformas.

- Zolvers [13]
- Timbrit [20]
- Homesolution [11]
- Opcionempleo [8]

Vale la pena destacar que estas soluciones son simplemente un marketplace para conectar trabajadores con gente que necesita de ciertos servicios, pero carecen de redes de confianza que den credibilidad a las recomendaciones.

3 Alcance del proyecto

Como se describió en la sección anterior, el objetivo principal de este proyecto es lograr conectar vecinos con proveedores de servicios. Dentro de la aplicación, debe ser fácil encontrar estos trabajadores, en base a las necesidades del vecino, y brindar una manera simple e intuitiva de contacto con el proveedor, considerando, también, que la edad media de quienes contratan trabajos para el hogar supera los 30 años.

Una de las funcionalidades más importantes de la aplicación será puntuar el servicio provisto, para que otros vecinos puedan verlo junto con reseñas y sirvan de referencia al decidir contactar al trabajador.

Durante la investigación para el proyecto, notamos que hoy en día los vecinos de una misma zona que se conocen, mantienen una lista informal de trabajadores de confianza en grupos informales. Esto presenta muchas limitaciones, no son flexibles ni existen reseñas disponibles y accesibles. Por ello la aplicación permitirá crear “Grupos de vecinos”, en donde los usuarios puedan ver qué vecinos son parte y quiénes lo administran. Estos grupos son los que permitirán que el usuario lea reseñas confiables, con nombre de sus vecinos, y no anónimas. A la hora de buscar trabajadores, el usuario podrá filtrar servicios por estos grupos y encontrar proveedores y sus reseñas respectivas.

Dentro de los Grupos de Vecinos, se brindará la posibilidad de hacer anuncios importantes para todos los integrantes del grupo, por ejemplo un aviso de seguridad para el vecindario. También, existirá la posibilidad de denunciar usuarios, ya sea porque no pertenecen a la comunidad o porque no cumplen con las políticas del grupo, y denunciar reseñas por ser ofensivas o utilizar vocabulario inadecuado. A su vez, habrá una lista negra de trabajadores, que por ejemplo, cobraron por un trabajo que no fue realizado.

Por otro lado, los trabajadores podrán crear su perfil, incluyendo la información que les parezca adecuada, como información de contacto, fotografías de trabajos realizados, presupuestos aproximados, referencias, etc.

Cuando un usuario vecino quiera contactarse con un trabajador, para preguntarle sobre el servicio que desea, cuál sería el presupuesto, los horarios disponibles, etc, no será necesario brindar información privada, como el número de teléfono, ya que se podrá utilizar el chat interno de la aplicación. De esta manera se brinda más seguridad a los usuarios, ya que no es necesario dar datos de contacto cuando aún no se tiene una confianza establecida con el trabajador. También, este chat interno permite monitorear y moderar la comunicación entre los usuarios.

4 Producto desarrollado

Se comienza esta sección explicando el flujo general de HomeFix, para una mejor comprensión del producto desarrollado. Se continuará mostrando los requerimientos, tanto funcionales como no y los de los diferentes usuarios. Se explica también la arquitectura de HomeFix, tanto del backend como del frontend, diagramas de bajo y alto nivel, las tecnologías utilizadas en cada uno y las limitaciones que presenta.

4.1 Flujo General

Previo a exponer los requisitos funcionales, ya que todos los usuarios presentan muchas funcionalidades, se ilustra un flujo básico de la aplicación, para que a la hora de mencionar los requisitos funciones, se entienda cuál es el objetivo y el alcance de cada requisito.

Se adjunta el diagrama del flujo general al informe y a continuación se lo explica.

Al entrar a HomeFix, si uno no está registrado, puede crear una nueva cuenta. La plataforma ofrece opciones para diferentes tipos de usuarios, tanto como vecino, administrador de vecinos, o trabajador. Se debe confirmar el email, y luego se podrá iniciar sesión.

Si el usuario no desea registrarse, puede ingresar a la página como invitado. Este ingreso es el más restrictivo, ya que solo puede buscar trabajadores y ver sus perfiles, pero no presenta la opción de contactar al trabajador. El usuario puede ver las reseñas que el trabajador tenga, pero no puede ver información sobre quién la escribió.

Si se inicia sesión como vecino o administrador, se redirige a buscar trabajadores, donde tiene la opción de ver el perfil de los diferentes trabajadores, sus reseñas, su información general, y tiene la posibilidad de contactarlos. Esto abre el chat integrado de HomeFix, y allí se puede chatear con el trabajador, para hacerle preguntas sobre el servicio que se desee contratar.

También puede ver su perfil como vecino o administrador, en donde se muestran los grupos de vecinos al que pertenece, las contrataciones que tiene, los mensajes, su información general y edición de su perfil.

Si se entra a algún grupo de vecinos, el usuario puede ver los mensajes parroquiales, ver quiénes lo integran y sus administradores. Si se es administrador, se pueden escribir nuevos mensajes y eliminar vecinos del grupo.

Con respecto a las contrataciones, estas tienen 3 estados: "En espera" significa que el proveedor realizó un presupuesto tentativo, que debe ser aprobado por el vecino. Una vez que se acepta pasa a estado "Aceptada", y luego tiene la opción de pasar a "Finalizada". Cuando está en este último estado, se le da la posibilidad al vecino de escribir la reseña.

Con respecto a los mensajes, simplemente se listan todas las conversaciones que tuvo el vecino con los trabajadores. Recordemos que la única manera de iniciar una conversación con un trabajador es a partir del botón de contactar en su perfil.

Si se es usuario trabajador, al iniciar sesión se redirige a su perfil, en donde puede ver su información general y editarla, sus mensajes con los diferentes vecinos, y las contrataciones que están "En espera", es decir, aguardando que un vecino las acepte. También tiene la opción de ver las contrataciones "Aceptadas" y las "Finalizadas".

Puede también buscar a otros trabajadores, ver su perfil, pero no puede contactarse con ellos. Como este trabajador no pertenece a ningún grupo de vecinos, su opinión sobre el trabajo realizado por otro proveedor no sería útil, debido a que sería anónima.

4.2 Requerimientos funcionales

- La solución debe permitir a los vecinos encontrar trabajadores según las necesidades que estos tengan. Debe ser simple de buscar y filtrar sin demasiada complejidad.
- Permitir la creación de usuarios administradores de grupos de vecinos.
- Permitir la creación de usuarios vecinos.
- Permitir un chat entre vecinos y trabajadores que funcione en tiempo real y evite el uso de sistemas externos de mensajería, como Whatsapp, Telegram o SMS. El mismo debe ser de carácter privado.
- Debe ofrecer a los trabajadores crear su perfil como ellos deseen para lograr cautivar la atención de los vecinos, con fotos de trabajos previos, descripciones e información que al vecino le puede interesar.
- Enviar mensajes a todos los vecinos para informar cualquier cambio que pueda ocurrir dentro de un consorcio.
- Permitir la creación de grupos de vecinos y poder administrar y unirme con facilidad a aquellos grupos que ya existan.
- Permitir la creación de usuario vecino, que pueda agregarse a diferentes grupos de vecinos.
- Se debe poder aceptar, rechazar o finalizar presupuestos laborales, con su debido presupuesto y descripción.

4.3 Requerimientos no funcionales

- Debe ofrecer una interfaz visualmente intuitiva y simple de usar para los usuarios. Será diseñada con criterios simples, pero con pruebas extensivas de usabilidad y accesibilidad.
- Debe estar disponible en un sitio web para que pueda ser accesible a través de un navegador tanto del celular como del ordenador, y de esta manera llegar a más usuarios.
- Debe ofrecer algoritmos de hashing para el guardado de contraseñas de manera segura.
- Debe ofrecer en los distintos formularios que debe completar el usuario mensajes de error claros y fáciles de comprender.

En las siguientes secciones se dará más detalle de cómo fueron contemplados estos requerimientos no funcionales.

4.4 Requerimientos de usuarios

Dentro de esta sección se trabajarán las expectativas de los usuarios y cómo estos interactúan con el producto. Se definen 5 tipos de usuarios.

4.4.1 Invitado

El usuario invitado es aquel que no está registrado y tiene la capacidad de:

- Darse de alta/registrarse.
- Buscar empleados por rubros, nombres o área de trabajo.
- Ver el perfil de los trabajadores, su nombre, información general, las reseñas del mismo de forma anónima y una galería con fotos de trabajos realizados en el pasado por el mismo.

4.4.2 Vecino

El mismo puede **crear su perfil**, y debe presentar el nombre del grupo de vecinos al cual quiere unirse y un email que no esté siendo utilizado en la plataforma. Este email es su **identificador**, por eso debe ser único entre todos los usuarios.

Una vez en su perfil el mismo puede **editarlo**, **añadirse** a otro grupo de vecinos, abandonar un grupo de vecinos, crear un grupo de vecinos, ver **mensajes** parroquiales de sus grupos, ver a los vecinos de sus diferentes grupos y su información de contacto, y lo mismo con los administradores.

Si bien la **búsqueda** de los trabajadores es similar a la de un usuario invitado, este tiene la posibilidad de realizar una búsqueda mucho más interesante que el previamente descrito. El vecino puede asociar su búsqueda en base a un **grupo de vecinos en particular**, en donde se les va a listar primero aquellos trabajadores que han sido contratados por vecinos, luego aquellos que trabajen por la zona del grupo seleccionado y finalmente el resto de los trabajadores.

Además, este puede marcar reseñas como inapropiadas o decir si una reseña fue o no útil. Debe ser capaz de **generar una conversación** a través del chat interno de HomeFix para comunicarse con el trabajador y tener la posibilidad de **administrar contrataciones** con aquel trabajador que ya se ha contactado.

Una vez que el trabajador le envía al vecino un presupuesto, si este lo aprueba, puede ver en otra solapa todas sus **contrataciones activas**. Una vez finalizada la contratación, el mismo puede escribir una **reseña** del trabajo realizado por el trabajador.

También puede ver todos sus **mensajes** con todos los trabajadores que haya contactado, y las **reseñas** que él realizó.

4.4.3 Administrador

El usuario administrador es el encargado de **manejar** a uno o varios grupos de vecinos.

El mismo al crear su perfil, puede seleccionar que tipo de usuario quiere ser (vecino, administrador o trabajador), en este caso administrador.

Para **crear el usuario** de manera exitosa, el mismo tiene que proveer un nombre de grupo no existente, la dirección del mismo (única también) y un email no utilizado en la plataforma.

Una vez creado su perfil y el grupo, este puede entrar a manejar su perfil, **editarlo**, **agregarse** a un nuevo grupo de vecinos, **crear** un nuevo grupo, enviar mensajes parroquiales a los grupos los cuales administra y **abandonar** el grupo si así lo desea (siempre y cuando no sea el único administrador). Puede ver los vecinos de ese barrio, obtener la información de contacto de los mismos y **eliminar** a algún vecino por cualquier motivo. También puede ver a los otros administradores de los grupos, pero no puede eliminarlos. Este usuario es el encargado de difundir el nombre del grupo, para que los usuarios vecinos puedan agregarse al mismo.

Con respecto a las contrataciones y mensajes, tiene las mismas funcionalidades que un usuarios vecino.

Cabe aclarar que un usuario puede ser vecino y administrador al mismo tiempo, de diferentes grupos.

4.4.4 Trabajador

Puede hacer todo lo que realiza el usuario invitado y además **crear o editar** su perfil como trabajador. Esto incluye datos personales, fotografías, horarios disponibles, etc. Lo único que no puede modificar es su email, ya que es su identificador.

El mismo puede buscar trabajadores, ver sus perfiles, pero al ser únicamente usuario trabajador, no puede contactarse con ellos.

También puede **ver sus mensajes**, con vecinos que se hayan comunicado con él para la realización de cierto trabajo, puede **enviarle presupuestos** por este trabajo al vecino, puede **cancelarlos** en caso de que haya algún error, y puede **finalizar** la contratación una vez terminado el trabajo. En este caso, ya le daría la posibilidad al usuario vecino de comenzar la reseña de este trabajador.

Si hubo algún problema con el trabajo realizado, o el trabajador nunca se presentó, el vecino al también tener la posibilidad de finalizar la contratación, ya puede comenzar a escribir la reseña.

En caso de que el usuario trabajador reciba reiteradas calificaciones negativas, un usuario SuperAdmin puede incorporarlo a la lista negra de trabajadores. En este caso al usuario trabajador al intentar iniciar sesión, se le muestra un cartel de usuario bloqueado. Para manejar este caso, se cuenta con una baja lógica, es decir en el módulo 5.8 (Desarrollo de SuperAdmin) se pueden agregar y sacar usuarios de la lista negra.

Para esta primera instancia no se incluye la posibilidad de que un usuario pueda **eliminar** su propia cuenta. Por tratarse de un MVP esta funcionalidad quedó fuera del alcance y será contemplada en futuras iteraciones del producto. Si bien es una limitación propia de la plataforma en esta primera instancia, lo que se buscó es evitar que un usuario trabajador, ante una calificación negativa vuelva generar un nuevo perfil y no quede registro de su historial negativo.

Finalmente, el usuario trabajador también puede visualizar la fecha y hora en que entró a por última vez su perfil a ver sus conversaciones. Esta información también está disponible para el usuario vecino y le será de utilidad para asegurarse de que se está contactando con un usuario activo dentro de la plataforma.

4.4.5 SuperAdmin

Puede realizar lo mismo que un usuario invitado y además revisar las **reseñas marcadas como inapropiadas** y **dar de baja trabajadores** que tengan malas reseñas o mal rating en reiteradas ocasiones. Estos trabajadores estarán en la *lista negra*, para que así no puedan iniciar sesión ni volver a registrarse con el mismo email.

4.5 Arquitectura

En la figura 1 puede observarse el diagrama de arquitectura elegido para HomeFix. Al ser un MVP (Producto mínimo viable), se busca resolver los problemas con el menor costo de infraestructura posible.

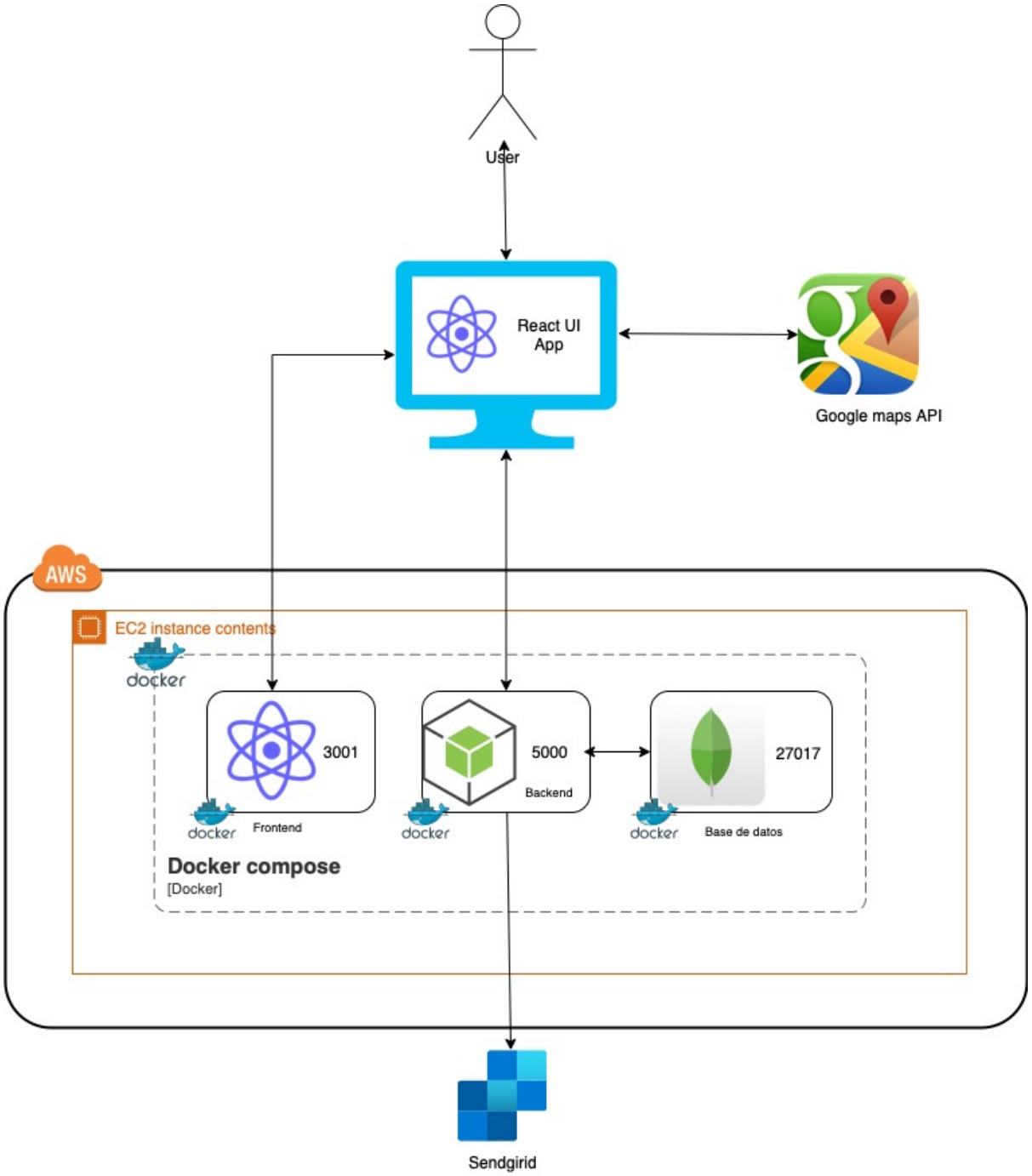


Figura 1: Diagrama de arquitectura (Fuente: elaboración propia).

En los puntos a continuación se explica como interactúan cada uno de estos componentes

fundamentales para el funcionamiento de la plataforma.

4.6 Backend

Este tiene la responsabilidad de ofrecer y disponibilizar todas las funcionalidades y requerimientos funcionales que la solución ofrece de manera segura y fácilmente accesible desde el Frontend.

El backend es una API Rest desarrollada en Node Js, que es uno de los mejores entornos de ejecución de JavaScript para desarrollo de aplicaciones y en conjunto con la librería Express nos permite realizar API 's escalables, rápidas y seguras.

Node.js es un entorno de ejecución para JavaScript de código abierto principalmente para la capa del servidor basado en el motor v8 de Chrome. Se optó por esta herramienta ya que es la herramienta más usada en el mercado para el desarrollo de aplicaciones. Su curva de aprendizaje, escalabilidad y comunidad permiten desplegar proyectos de manera segura y fácil. Una de las alternativas que se analizó fue Spring para el desarrollo del backend en Java, pero se encontró que Node.js en conjunto con Express era más rápido, más ligero, menos boilerplate y utiliza menos recursos. [21]

Este componente se debe comunicar con dos servicios diferentes, una base de datos en MongoDB, que se encuentra dentro del mismo entorno de AWS [3], y un servicio externo que se encarga del envío de emails (Sendgrid).

A su vez, el backend es consumido por el frontend en React.js [17], que luego se detalla más a fondo. Para la conexión con el frontend, el backend disponibiliza endpoints en HTTP y además, para que el chat en tiempo real sea posible, se realizó una implementación de secure web sockets. Se analizó la posibilidad de utlizar PubNub o API Gateway Sockets, pero se terminó realizando una implementación propia utilizando la librería de Socket.io [18], ya que las alternativas son caras monetariamente. Socket.io es una biblioteca basada en eventos para aplicaciones web en tiempo real. El objetivo del mismo es permitir la conversación entre un vecino y un trabajador y que las mismas estén autenticadas.

Otro factor importante para el desarrollo del backend es la securización de sus endpoints. Para lograr esto con los distintos roles que existen en la aplicación se optó por JWT (JSON Web Tokens) en donde dentro de cada objeto se encuentra información del usuario, el rol y el identificador del mismo.

Finalmente se utilizó CORS para solo permitir que el frontend consuma de los recursos que ofrece el API Rest.

4.6.1 Diseño de bajo nivel - Patrón 3 Tier

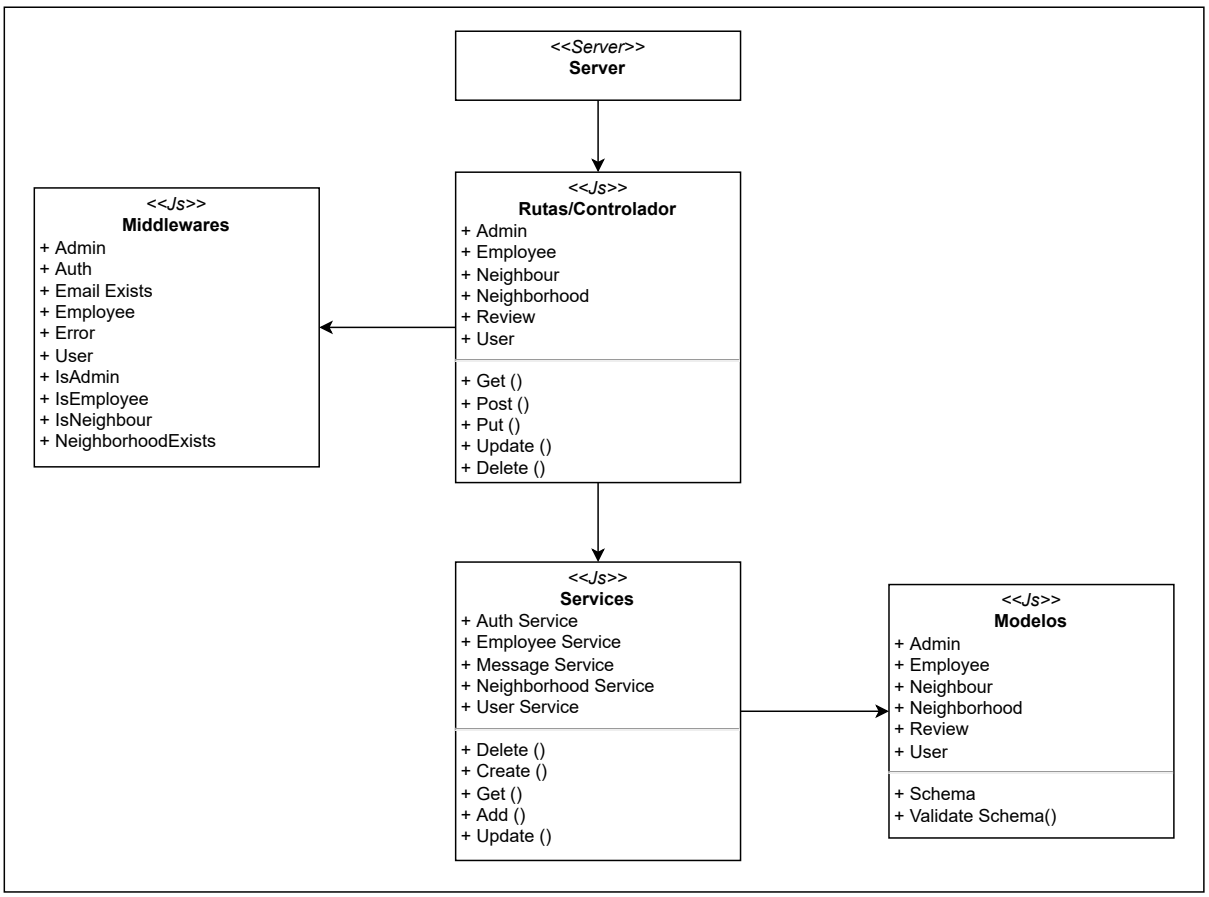


Figura 2: Patrón (Fuente: elaboración propia).

Se puede ver en la figura 2 que se siguió el patrón 3-tier. El objetivo de esto es principalmente organizar de mejor manera la aplicación, permitir que solo una parte se encargue de la lógica de negocio, poder reutilizar código y hacerlo lo más mantenible posible.

El primer *tier* (rutas/controlador) se encarga de procesar las llamadas provenientes de las peticiones hechas a la API. Express.js fue importante en este tier ya que permitió hacer uso de los endpoints que fueron configurados y asociados con middlewares para el análisis de las peticiones hechas hacia cada recurso del backend.

El segundo *tier* (services) representa la lógica de negocio asociada al proyecto en sí. Cada estructura tiene asociada un servicio y este va a tener la lógica para cada recurso de la API asociada.

El último *tier* son los modelos que luego serán guardados en la base de datos. Esta información a su vez, antes de ser ingresada a la base de datos, es validada usando la librería Joi para que no se inserten elementos inválidos en la base de datos.

Finalmente, se puede observar una capa llamada middleware que se encarga del manejo

de los requests realizados por quien consume las rutas del backend y devolverle al mismo un mensaje de error en caso de que el request haya sido invalido.

4.6.2 Estructura del backend

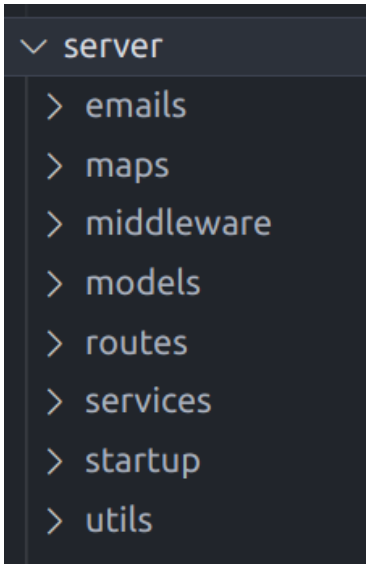


Figura 3: Estructura del código en backend

El backend posee varias carpetas para ordenar lo más posible el funcionamiento del mismo.

- Emails: se encarga de tener los templates de cada email enviado desde el backend. En la sección de sendgrid se abarca más en detalle para qué son estos emails, pero cada template tiene su sección de HTML y css.
- Startup: se encarga de inicializar las conexiones con otros servicios externos, como puede ser la base de datos y también cargar los controladores.
- Utils: tiene código que es reutilizado por todo el backend que facilita el desarrollo del mismo.
- Routes: tiene los handlers para cada endpoint provisto por la API Rest y con el uso de los middlewares ubicados en la carpeta con el mismo nombre, se hacen las validaciones pertinentes a cada ruta provista por la API. Además se encarga de enviar mensajes de error si es que llegan a existir.
- Services: como bien se explicó anteriormente, se encarga de toda la lógica relacionada a la lógica de negocios.
- Models: se encarga de los modelos que van a ser guardados en la base de datos.

4.7 Base de datos

Como base de datos principal se optó por MongoDB [14], una base de datos no relacional (NoSQL) orientada a documentos JSON. Esta se encarga de almacenar toda la información relacionada a la aplicación, desde información de los usuarios hasta el chat entre un vecino y un trabajador.

La motivación para elegir esta base de datos fue la simplicidad a la hora de realizar cambios, ya que esta es muy flexible a cambios de estructura. La posibilidad de tener un esquema dinámico que se moldee en base a las necesidades del proyecto y que pueda variar en base a los cambios que hubieron en las entregas parciales fue un punto importante a la hora de elegir esta base de datos frente a una base de datos relacional. Fue también importante a la hora de desarrollo ya que permite hacer cambios en la estructura de la base de datos sin afectar el funcionamiento de la aplicación.

Otro beneficio de MongoDB es su comunidad y el hecho de que hace muy buena sinergia con JavaScript [1], el lenguaje de programación elegido para el desarrollo de este proyecto. Justamente se eligió mongoose, una librería para el manejo de MongoDB en JavaScript que permite hacer queries de agregación y maneja relaciones entre el esquema provisto por la capa de modelos y la información guardada en la base de datos en sí. Esto además nos permite modelar los objetos de la base de datos como objetos en JavaScript y resulta en una gran optimización del tiempo de desarrollo.

4.8 Envío de emails

Para el envío de emails se optó por Sendgrid [7] que es una solución en la nube de SMTP que permite enviar emails sin necesidad de tener servidores para esto del lado de la aplicación ni tener que mantenerlos.

Para HomeFix se utilizaron 3 emails para diferentes usos.

1. Registro de cuenta: Si bien en otra sección del informe se especifica el flujo del mismo, se envía un email para validar que el usuario no quiera impresionar a otra persona utilizando un email que no es de su pertenencia.
2. Recuperación de cuenta: Para recuperar la cuenta se envía un email con un link para poder recuperar la cuenta cambiando la contraseña.
3. Vecino se unió al grupo: Un email se envía a todos los administradores de un grupo cuando un nuevo integrante se une al mismo.

Se contemplaron solo algunos de los casos posibles para el envío de email y los que quedaron fuera de alcance en esta versión se detallan en la sección 9.2 de Trabajos Futuros.

4.9 Frontend

El frontend de HomeFix debe ser fácil de usar, intuitivo y las funcionalidades básicas deben ser *mobile friendly* para verse de una manera agradable tanto en un ordenador como en un teléfono celular.

Debe ser el centro en donde todos los distintos usuarios puedan relacionarse entre sí y hacer uso de las funcionalidades que HomeFix puede brindar.

Durante la concepción del proyecto se evaluó si realizar el frontend como una aplicación móvil o una aplicación web. Se decidió ir por este último camino por varios motivos:

- Las aplicaciones web llegan a una mayor audiencia ya que no es necesario instalar una aplicación en el celular para poder hacer uso de sus funcionalidades. El objetivo era evitar esta barrera de entrada ya que no queremos que este sea un impedimento para el uso de la aplicación web. [9]
- Es más fácil de mantener una aplicación web ya que si se quiere hacer un cambio en una aplicación móvil se necesita pedir permiso a App Store o Google Play y luego esperar a que todos los usuarios actualicen a la nueva versión de la aplicación.
- El último factor determinante para esta decisión es que es menos costosa y no requiere permiso de nadie para que esta sea una aplicación en producción.

4.9.1 Aspectos técnicos

Para el desarrollo de la aplicación web se eligió React.js, un framework en JavaScript.

React es una librería de desarrollo para JavaScript para la creación de páginas web. El mismo es un proyecto realizado por Facebook de código abierto para el desarrollo de Single Page Applications. Es una de las librerías para JavaScript más utilizadas en el mercado, y dentro de las alternativas se analizó Vue js y Angular. Se terminó decidiendo por React por los siguientes motivos (los datos utilizados son de una encuesta realizada por State of JS [19] que extrae datos de stack overflow):

- Es el framework para front end más utilizado con un 80%, mientras que Angular es usado por el 54% de la comunidad y Vue con un 51%.
- Es la herramienta con mejor imagen positiva en comparación con sus competidores.
- Tiene una mayor comunidad. Es de los repositorios de Github que mayor soporte recibe. El mismo posee más de 180.000 estrellas en Github y más de 10 millones de descargas.
- Funciona muy bien con componentes reutilizables y es el framework de frontend mas utilizado por las mejores 500 compañías del mundo (Fortune 500) [16]

Por el lado de diseño, se eligió la librería Material UI (MUI v5) [15]. Al inicio del proyecto se utilizó la versión 4, pero ni bien salió la 5 durante la segunda entrega del proyecto se tomó la decisión de hacer una migración a la nueva versión para evitar cualquier tipo de error y tener los diseños más actualizados que provee esta librería. Los diseños de los componentes de dicha librería cumplen todos los lineamientos de los requerimientos no funcionales que HomeFix necesitaba.

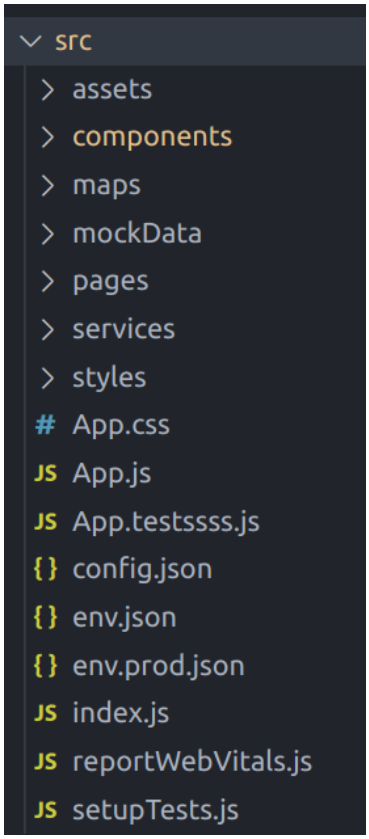


Figura 4: Estructura del frontend

Como se observa en la figura 4, se muestra cuál fue la estructura del proyecto del lado del frontend.

- assets: Posee todos los elementos estáticos, como por ejemplo imágenes y logos de la aplicación.
- components: Aquí se encuentran todos los componentes que la aplicación necesita para visualizar la información. Se encuentran tanto componentes comunes como los que resuelven algún diseño en particular.
- services: Se encuentra la conexión con servicios externos o globales. Además se encarga de reutilizar las llamadas al backend para evitar repetir código que tenga que ver con respuestas del mismo. Entre los archivos que se encuentran en este directorio se destacan:

- apiService.js: Es el encargado de hacer los requests al backend.
- store.js: Posee el estado general de la aplicación, utilizado principalmente para la búsqueda de empleados.
- authService.js: Se comunica con el backend para la autenticación y el manejo de los JWT.

4.10 Diseño de alto nivel

En la imagen 5 se puede observar cómo funciona la aplicación para el usuario invitado para distintos tipos de acciones y el flujo convencional a seguir, que consiste en ingresar a la aplicación, filtrar por categorías y luego por trabajadores. Cabe destacar que el mismo no presenta autenticación, por ende, no presenta JWT, los otros usuarios en cambio, si van a utilizar JWT.

Es importante mencionar que para cada acción de tipo POST, los formularios tienen validaciones de manera tal que no se pueda ingresar información no permitida a la base de datos.

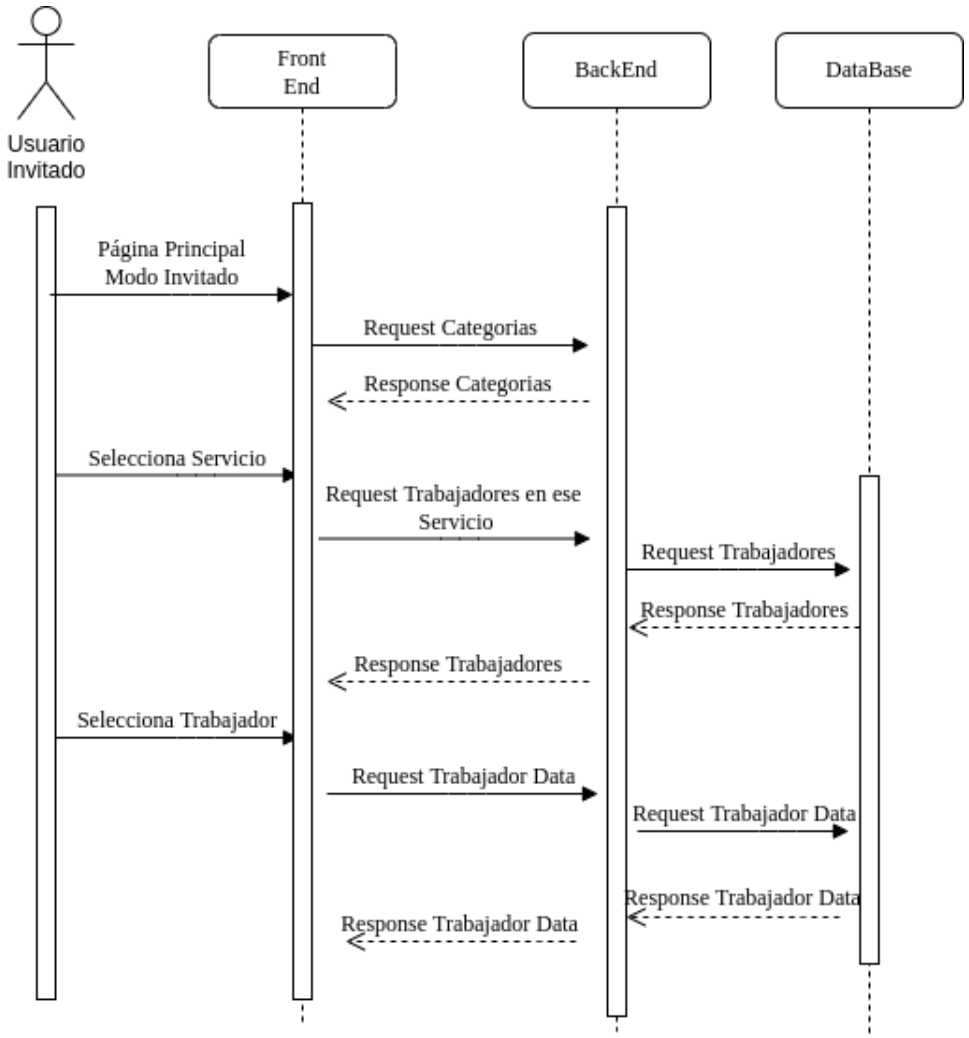


Figura 5: Diseño de alto nivel (Fuente: elaboración propia).

A continuación se brinda otro ejemplo de flujo, pero para un usuario vecino, es decir, autenticado en la plataforma.

Al realizar el LogIn, se envían las credenciales al backend, este se comunica con la base de datos la cual devuelve al backend el registro del usuario y a su vez, el backend devuelve un token (JWT) al cliente que representa la información básica de ese usuario y que sirve como autenticación en la plataforma. Previo a esto, el backend se va a encargar de validar la contraseña a través de un Hash para verificar que es el usuario correspondiente.

Este, al realizar cualquier acción que requiera autenticación, enviará su JWT al back, para que el mismo valide si es correcto y si se tiene permisos para realizar lo pedido.

Para mostrar un ejemplo más concreto, el usuario vecino ahora quiere buscar trabajadores en los diferentes servicios. El mismo se dirige a la página de búsquedas, la cual solicita al backend las categorías disponibles. Cuando el vecino selecciona la categoría deseada,

el back end le solicita a la base todos los trabajadores prestadores de ese servicio. Al seleccionar un trabajador, se le pide la información a la base de datos. Notemos que hasta este momento, no se requiere autenticación alguna.

Cuando el vecino desea contactar al trabajador (acción autenticada), se envía el pedido de lo que desea realizar con el JWT, el cual el backend nuevamente valida, y si es correcto sigue solicitando las cosas necesarias a la base de datos, en este caso el chat de ese vecino con el trabajador.

En este ejemplo también se ilustra que cuando el vecino contacta al trabajador, se crea un websocket, y cuando el backend responde que todo está correcto, se envía el response websocket, para que puedan chatear sin problemas.

Este ejemplo es para algo particular, pero puede llevarse a todo tipo de ejemplos, ya que su funcionamiento en todos los casos es parecido.

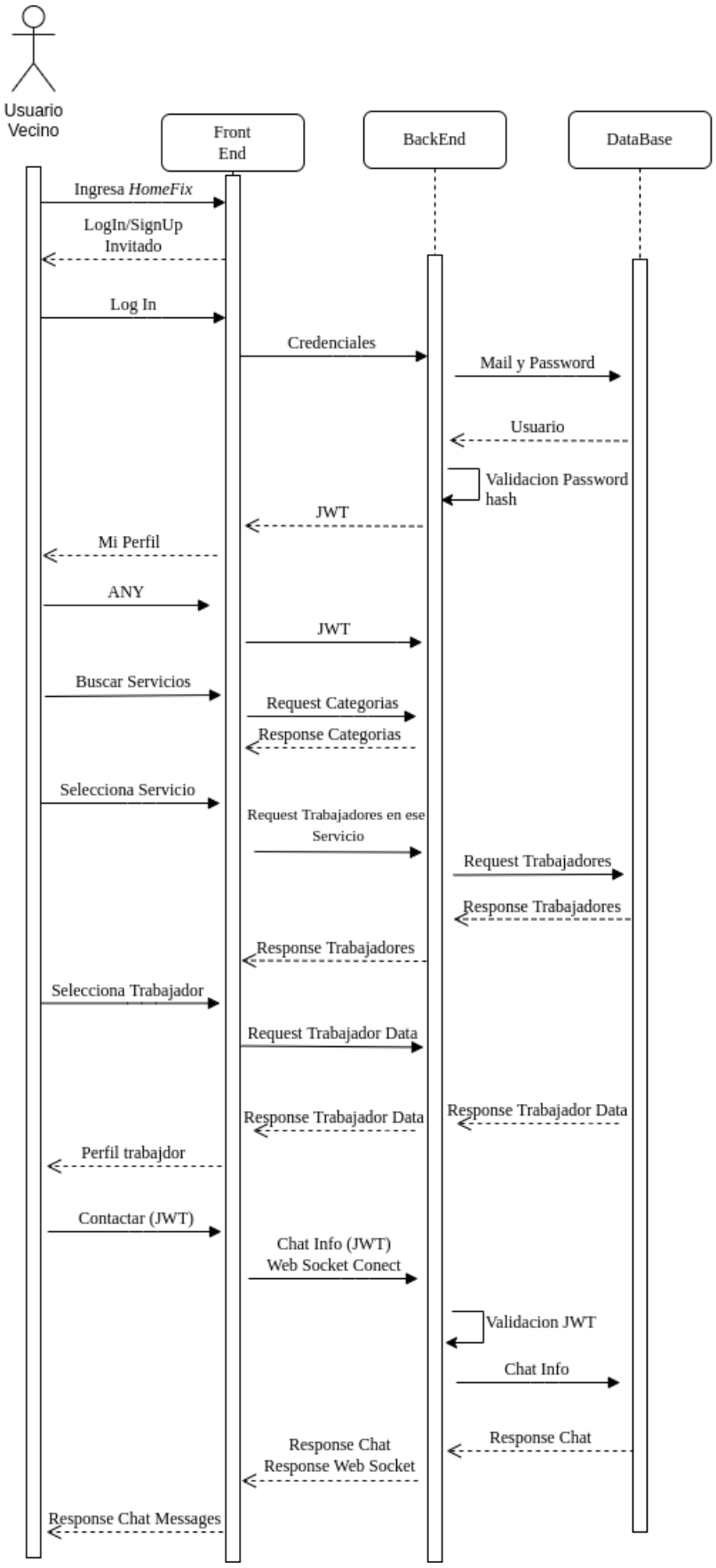


Figura 6: Diseño de alto nivel (Fuente: elaboración propia

4.11 Infraestructura

En este apartado se explicaran la infraestructura actual de HomeFix, las limitaciones que la misma presenta, y cual seria una infraestructura ideal, para el funcionamiento más óptimo posible del sistema.

4.11.1 Infraestructura actual

Por el lado de infraestructura, se utilizó la cuenta gratuita de AWS que la cátedra nos otorgó. Si bien presenta ciertas limitaciones que serán explicadas en el próximo apartado, nos permitió disponibilizar la aplicación utilizando servicios o workarounds para realizar todas las tareas que se necesitaban.

Tanto el back-end, front-end y base de datos se instalaron en una instancia EC2 de Amazon Web Services [4] con una imagen de Linux, que mediante el uso de Docker [6] y Docker Compose, inicializamos la aplicación y la orquestramos disponibilizando la misma. El precio de esta instancia es gratuito por un año, lo que nos permite realizar cuantas pruebas queramos.

En cuanto a las conexiones externas como la API de google maps [10] y Sendgrid para envío de email, fue suficiente con las configuraciones que la instancia de EC2 nos proporcionó.

Respecto a las imágenes, se decidió utilizar MongoDB [14] para persistir las mismas y otros puntos que se explicarán en el apartado de limitaciones.

4.11.2 Limitaciones

Como se mencionó anteriormente, la cuenta de AWS provista por la cátedra tiene varias limitaciones en el apartado de servicios disponibles. Esto afectó al desarrollo del proyecto debido a que ciertas acciones dentro de la aplicación pueden demorarse, como por ejemplo, la búsqueda de trabajadores, la creación de usuarios, subir imágenes a las galerías de los trabajadores, entre otros. Aun así, la aplicación es totalmente usable, pero se pueden realizar modificaciones que significan costos que permitirían una mejor experiencia al usuario y un uso de mejores prácticas.

Otra limitación de la aplicación es que debido al uso de la cuenta gratuita de AWS, esta no nos permite hacer uso del servicio Route53, que permite utilizar dominios más amigables y legibles y de esta manera evitar el uso de la IP para el acceso del mismo.

También, otra limitación es el disco utilizado para la instancia de EC2 de 8gb, este almacenamiento es tanto para la aplicación, como para la base de datos (incluyendo imágenes). El tamaño del mismo podría llenarse en el caso de que la aplicación sea muy utilizada.

4.11.3 Infraestructura óptima

Para tener un funcionamiento óptimo de la aplicación, se tendrián que realizar varios cambios a la infraestructura, especialmente a los servicios utilizados de AWS (o cualquier proveedor de los mismos). Como los integrantes del equipo tienen conocimiento de AWS, se usarán los servicios existentes proporcionados por este proveedor, pero cualquier otro posee alternativas similares.

En primer lugar, lo ideal sería guardar las imágenes en un object storage. Para esto se elegiría S3 ya que es el object storage de AWS. En el mismo, se guardarán imágenes relacionadas a la foto de perfil de los usuarios y la galería de imágenes de los trabajadores.

En segundo lugar, sería ideal que el frontend esté ubicado dentro un bucket S3 así el mismo se encarga de guardar el contenido estático que presenta la aplicación. También se utilizarían los servicios de Route53, para poder utilizar una URI propia y cloudfront como CDN de la aplicación para agilizar los tiempos de respuesta.

En tercer y último lugar, lo ideal sería tener una instancia dedicada y réplicas para la base de datos. Esto agilizaría los tiempos de respuesta pero incrementando el costo de la infraestructura.

5 Desarrollo

En esta sección se detallan las interfaces que se utilizaron para resolver las distintas funcionalidades de HomeFix. Como se mencionó anteriormente, fueron hechas con componentes de Material UI. Todos los formularios de HomeFix presentan validación tanto en el back-end como en el front-end.

5.1 Tipos de usuarios

En el lado del back end y front end se hace la distinción de 5 tipos de usuarios diferentes: invitado, vecino, administrador, trabajador y superAdmin. Estos usuarios se pueden dividir dentro de dos grupos diferentes, los usuarios comunes de la aplicación y el Super Administrador.

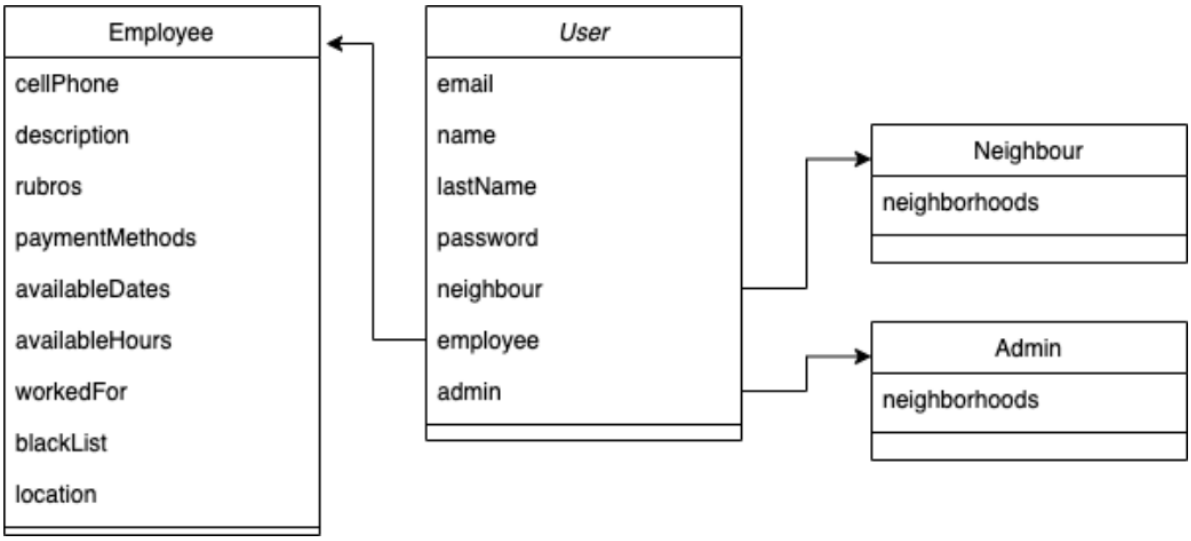


Figura 7: Diagrama de clase de usuarios (Fuente: elaboración propia).

Como se puede observar, el modelo de un usuario es quien maneja todo lo relacionado con la autenticación de cada usuario.

Un usuario puede ser un vecino y administrador al mismo tiempo, pero no puede ser ninguno de estos usuarios si es un trabajador. Esta lógica se maneja del lado de la creación de usuarios en el lado del back end.

La contraseña de los usuarios se guarda hasheada por temas de seguridad y el email de cada usuario es único, por lo cual no pueden existir dos usuarios con el mismo email.

5.2 Registro de usuarios

Al entrar al sitio web de HomeFix se observa la pantalla principal.

Figura 8: Página Principal

Si se selecciona el botón de la derecha *Iniciar Sesión / Registrarse* puede observarse el formulario de inicio de sesión.

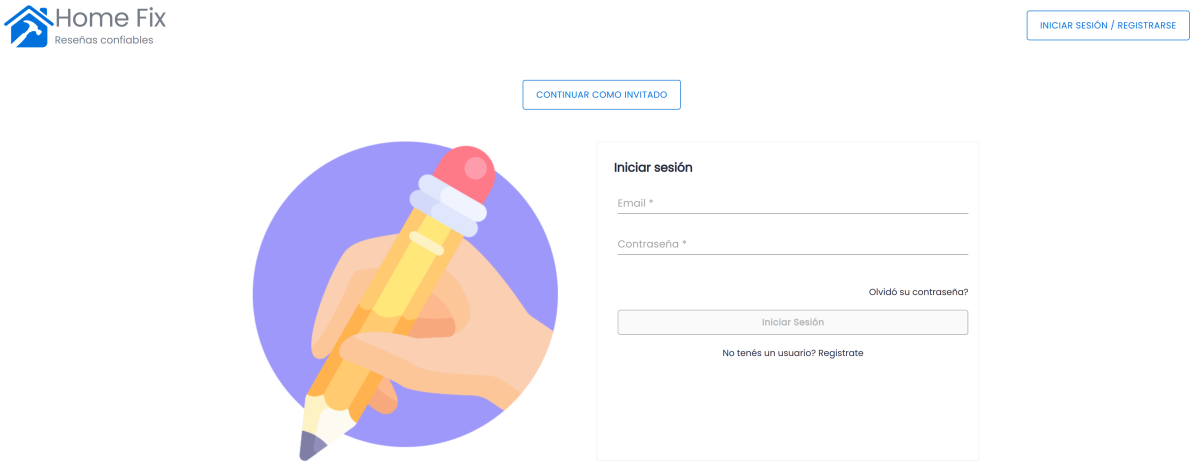


Figura 9: Formulario de inicio de sesión

Si se selecciona *No tenés un usuario? Registrate*, se observa la pantalla de la figura 10.

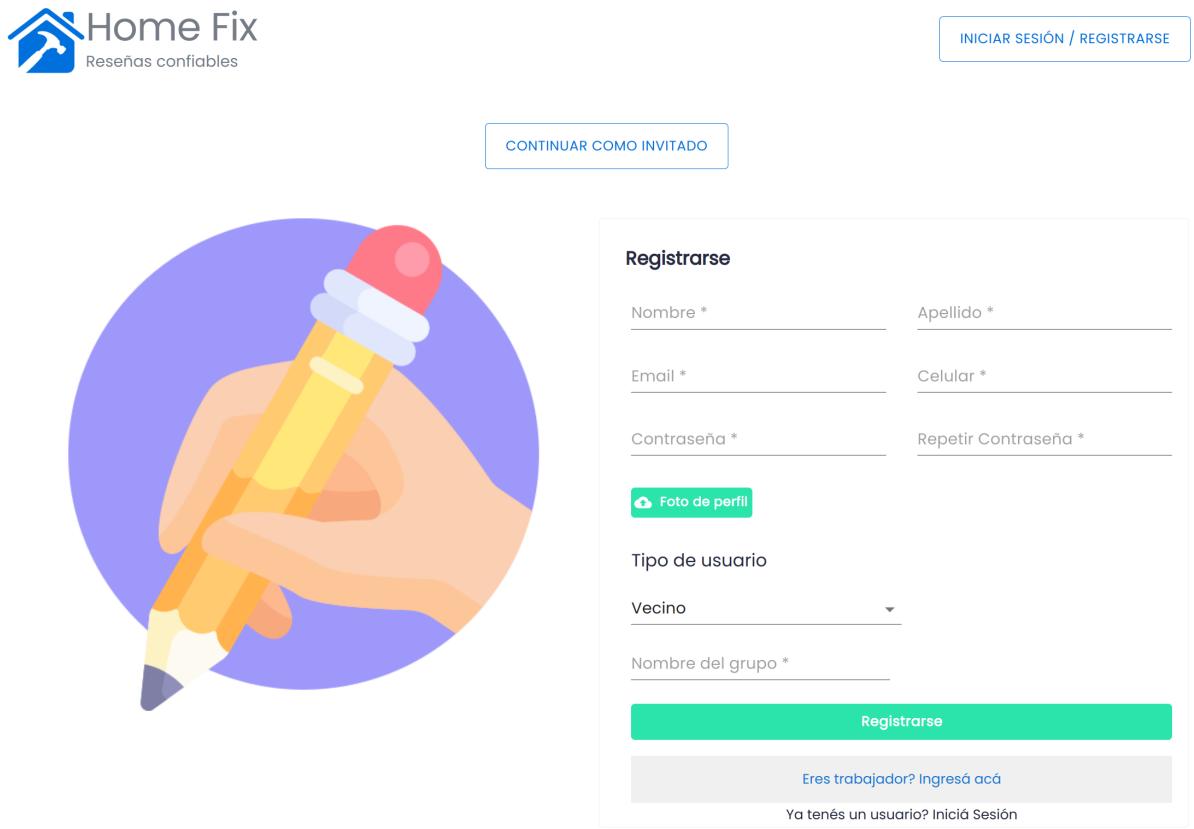


Figura 10: Formulario de registro

5.2.1 Registro de Vecino

Si se desea registrar a un usuario del tipo vecino, el usuario debe completar los campos mostrados, los mismos son todos obligatorios, a excepción de la foto de perfil. Se cuenta con validaciones para todos los campos usando la librería Joi [12]. A continuación se muestra un ejemplo de errores en el formulario de registro.

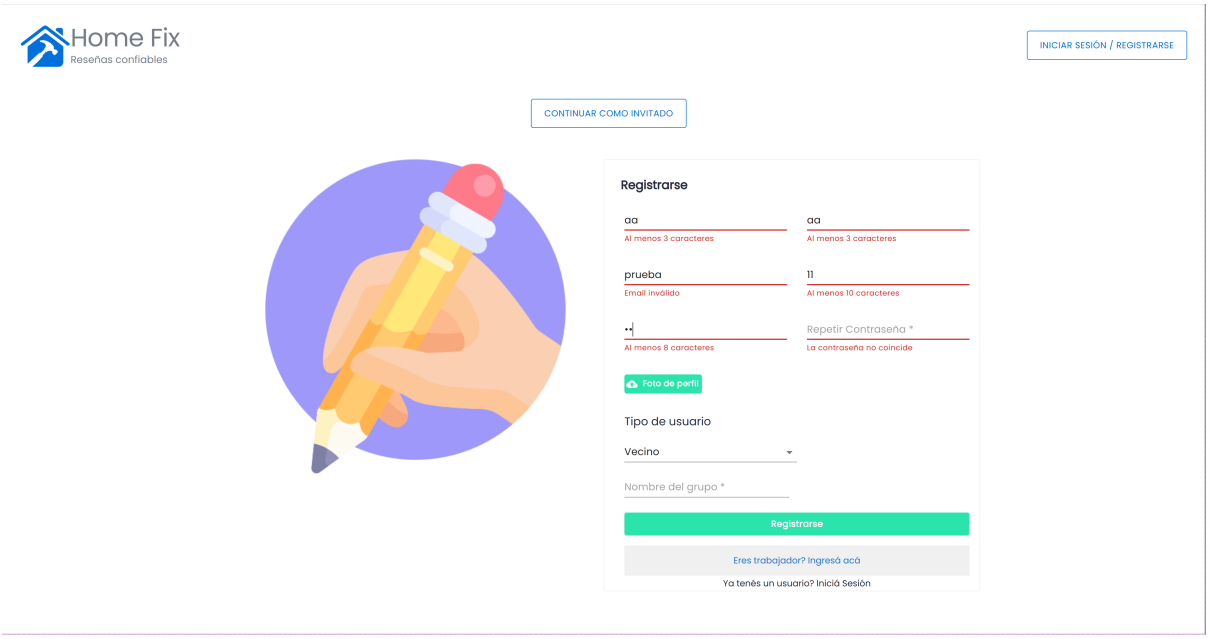


Figura 11: Formulario de registro con errores

Una vez que los campos están completos y sin errores, el usuario debe seleccionar el tipo de usuario que desee crear. En este caso *vecino*. El mismo debe completar el Nombre del grupo de vecinos al que desea unirse. Si el grupo no existe, se le mostrará un cartel de error al usuario.

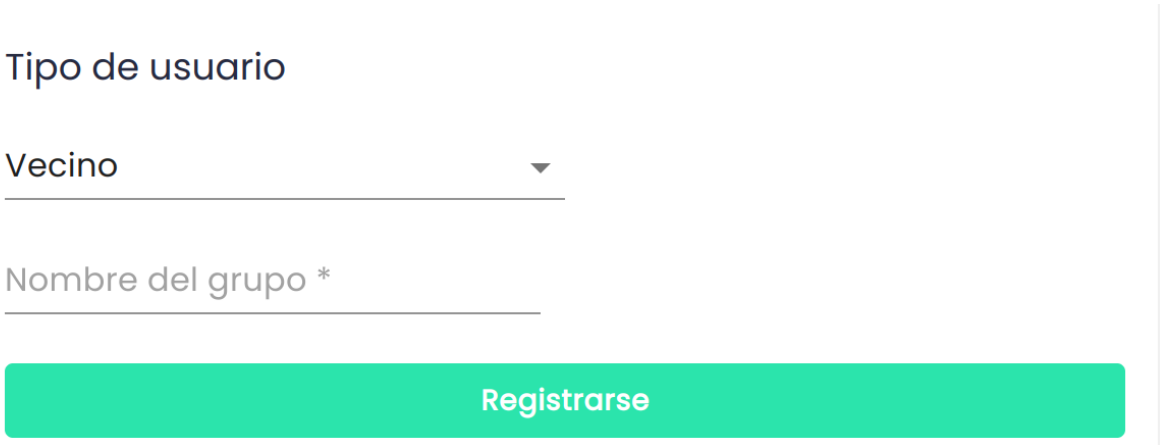


Figura 12: Formulario de registro tipo de usuario vecino

Si no hay errores, el formulario completo se verá como la figura 13.

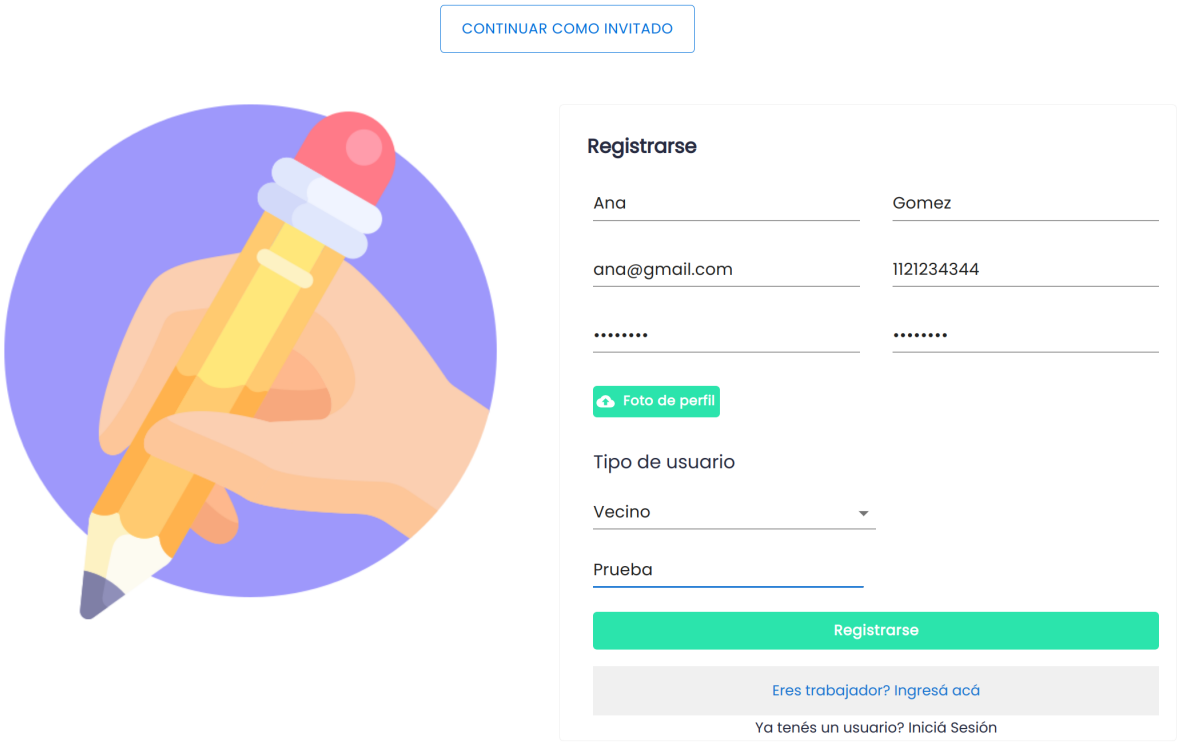


Figura 13: Formulario de registro completo

Al presionar el botón de REGISTRARSE, se mostrará el cartel de la imagen 14.

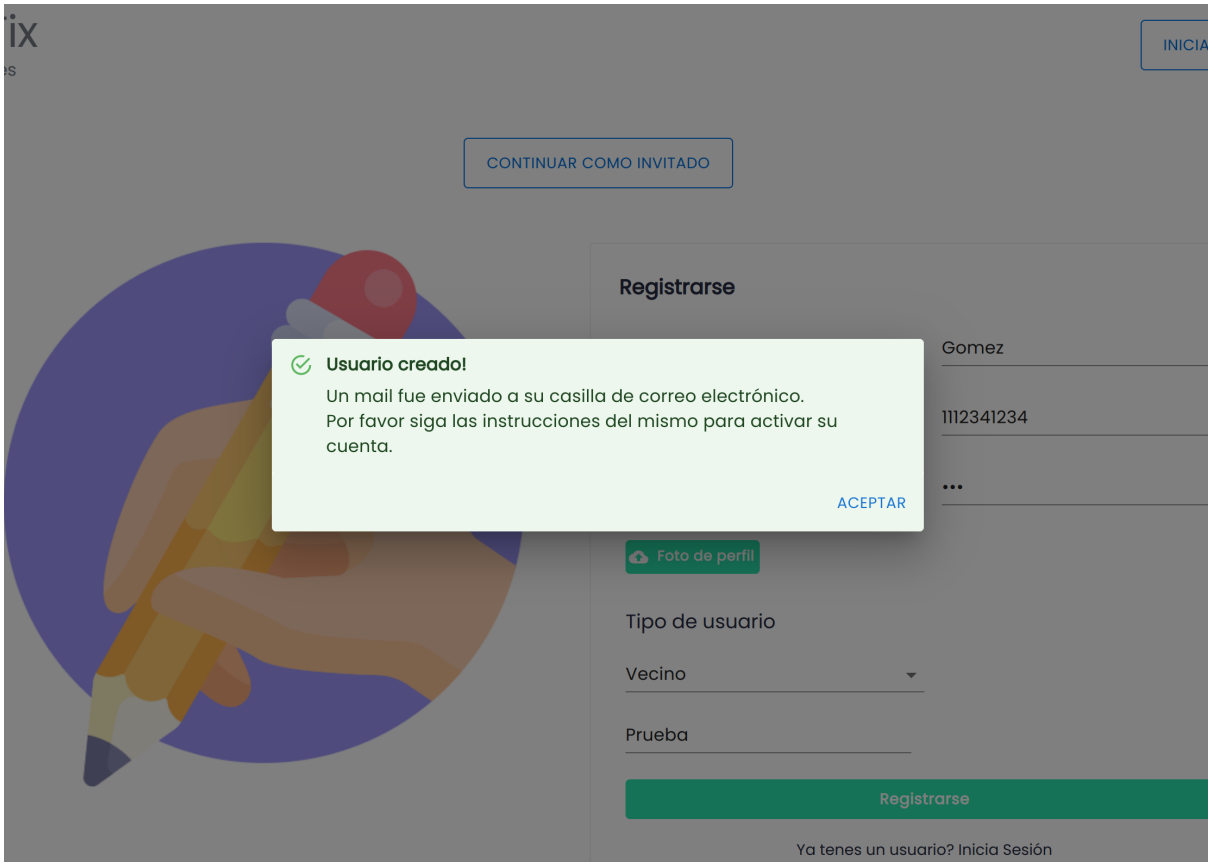


Figura 14: Mensaje Éxito creación de usuario vecino

Cabe recordar que el email es el identificador único de los usuarios, es decir, que si se ingresa un email ya registrado, se le mostrará un mensaje de error al usuario.

Si se revisa la casilla de correo electrónico, se podrá ver el email de la figura 15, en el cual si se presiona el botón de ACTIVAR CUENTA, redirigirá al usuario a la página de login de HomeFix para que el mismo inicie sesión con el email y contraseña provistos al registrarse.

Figura 15: Email activación cuenta

5.2.2 Registro de administrador

Si se desea crear un tipo de usuario administrador, el mismo debe completar los campos de la figura 10 y seleccionar como tipo de usuario *administrador*. Esto le muestra el formulario de la figura 16, en donde debe completar el código del grupo de vecinos que desea crear y la dirección del mismo. Para completar la dirección, se utiliza la herramienta auto complete de la API de Google Maps [10], la cual le presenta al usuario posibles direcciones válidas, y a su vez un mapa para ubicar físicamente la dirección que ingresó.

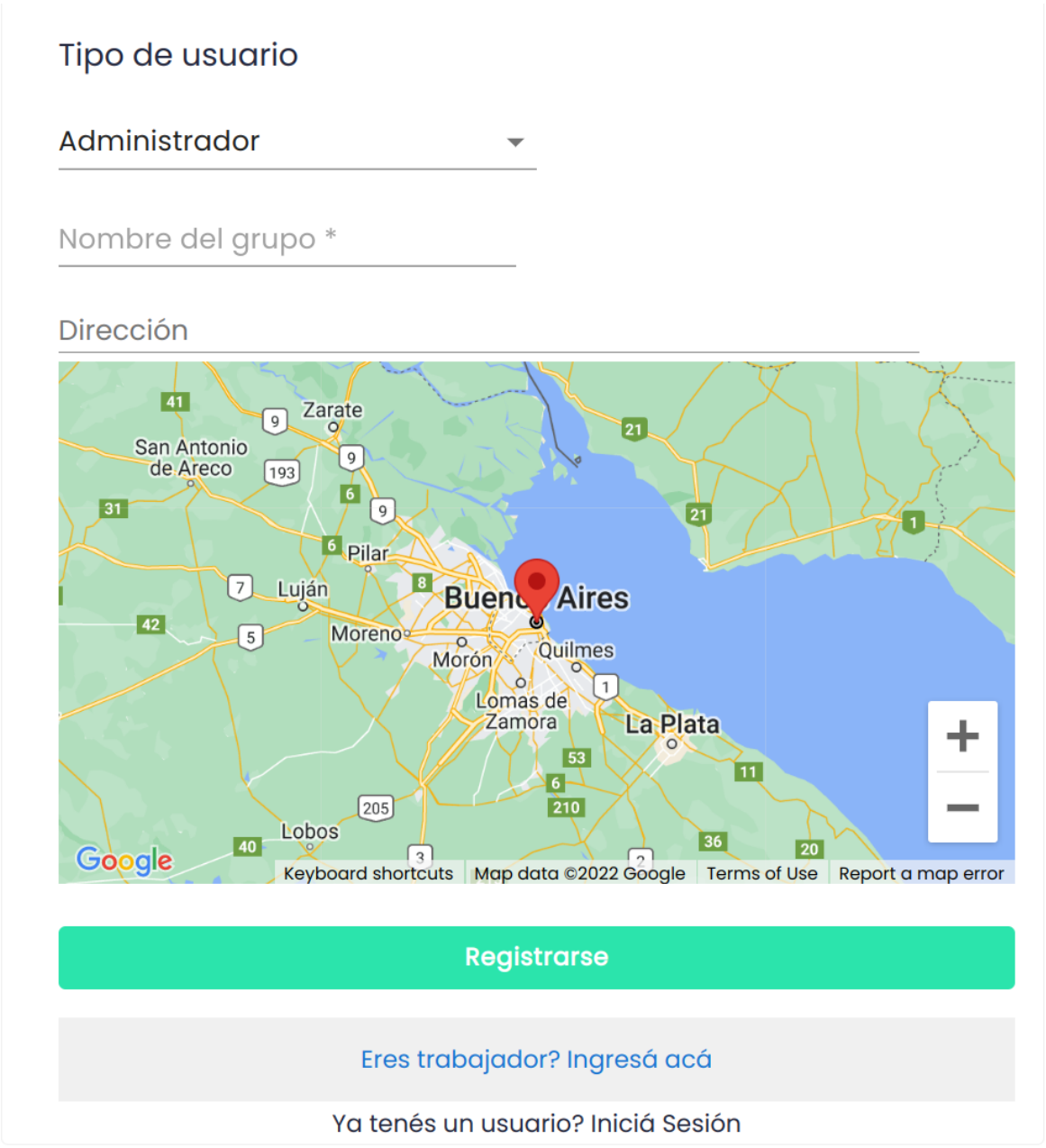


Figura 16: Formulario de registro tipo de usuario administrador

Ambos campos deben ser únicos, y si no lo son, se muestran carteles de error como puede observarse en la imagen 17, en el margen superior derecho. Estos errores se realizaron utilizando componentes de MUI v5 [15].

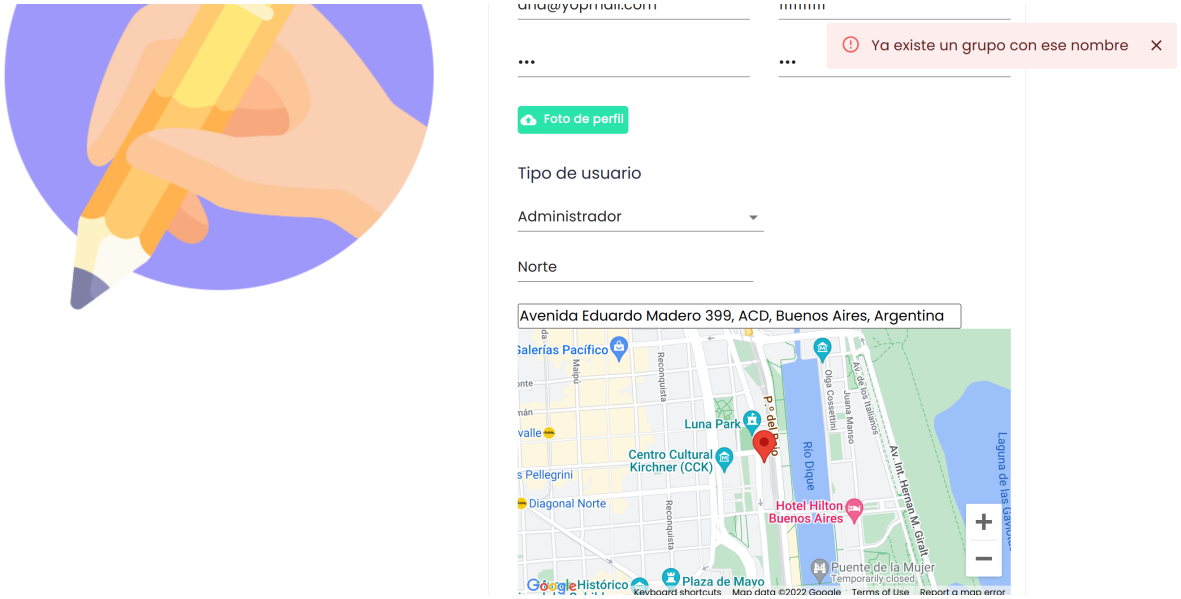


Figura 17: Error formulario de registro tipo de usuario administrador

Si no hay errores en el formulario de registro, el mismo se verá como la imagen 18.

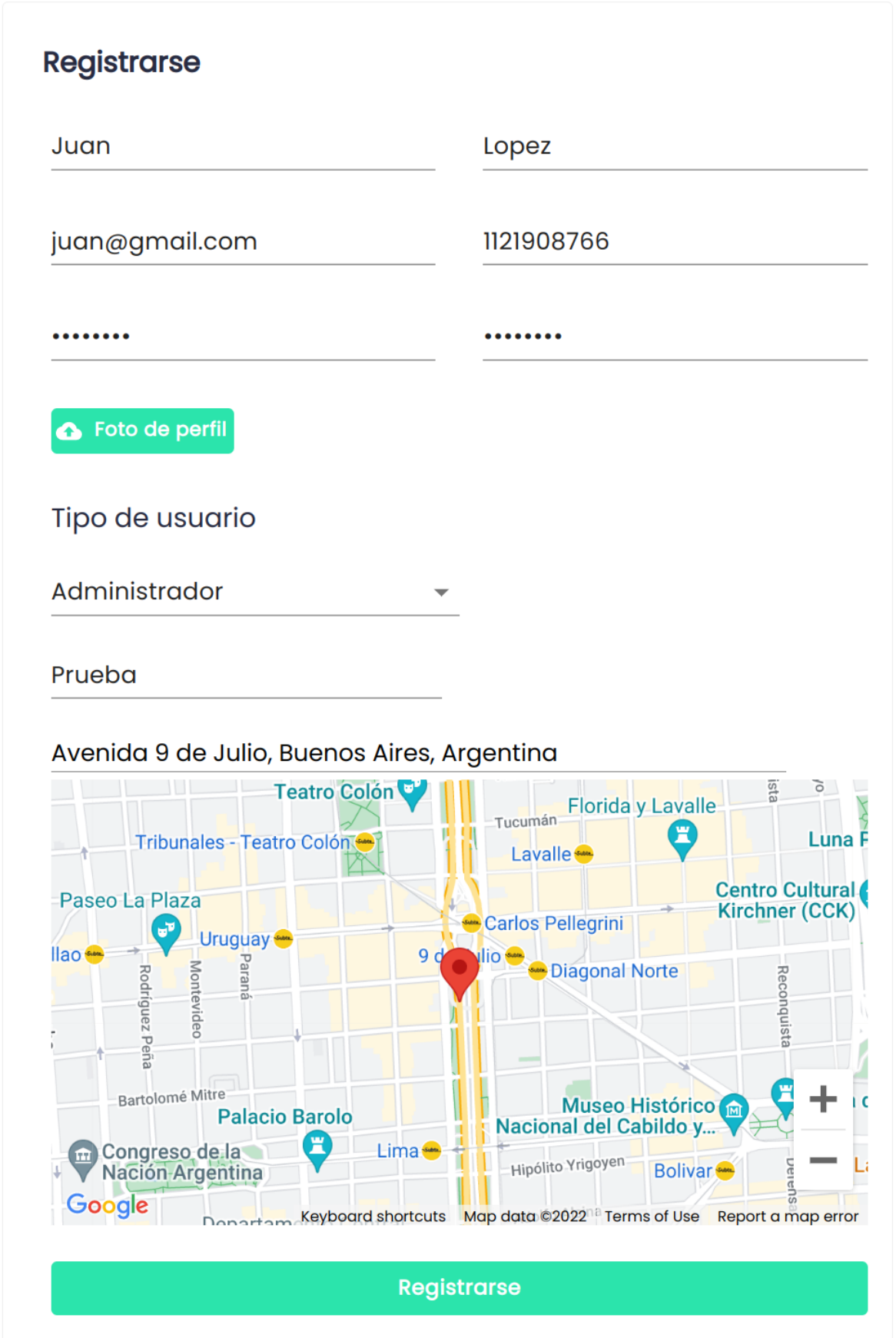


Figura 18: Formulario de registro tipo de usuario administrador

Al presionar el botón de REGISTRARSE, y si todo fue exitoso, se le mostrará el mismo mensaje de éxito que al usuario vecino, y recibirá el mismo email para activar la cuenta.

5.2.3 Registro de Trabajador

Si desea registrarse como un usuario trabajador, se debe clickear *Eres trabajador? Ingresá acá* de la figura 10, lo cual redirige al formulario de registro de un trabajador, como puede observarse en la figura 19.

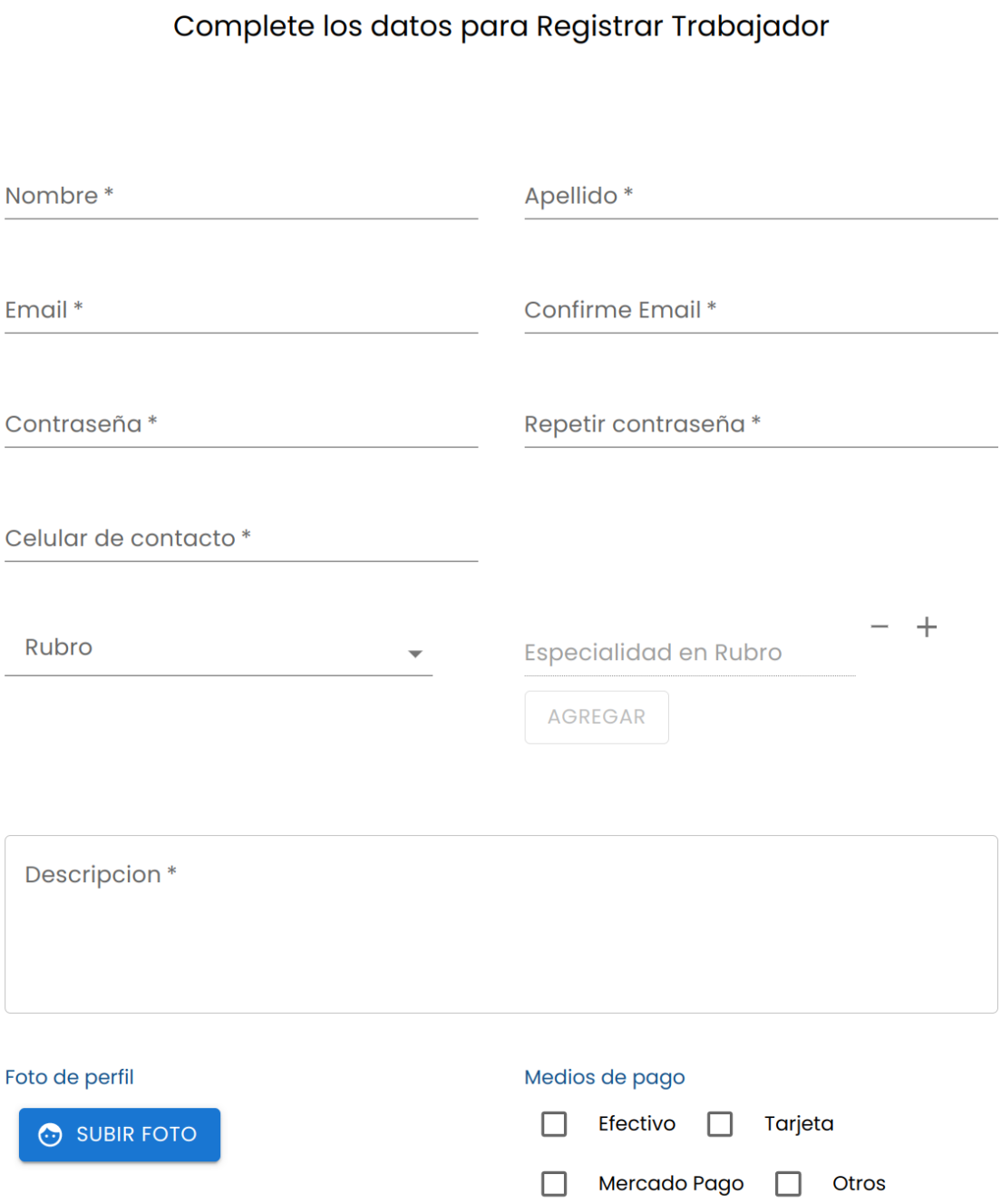


Figura 19: Formulario de registro tipo de usuario trabajador (1)

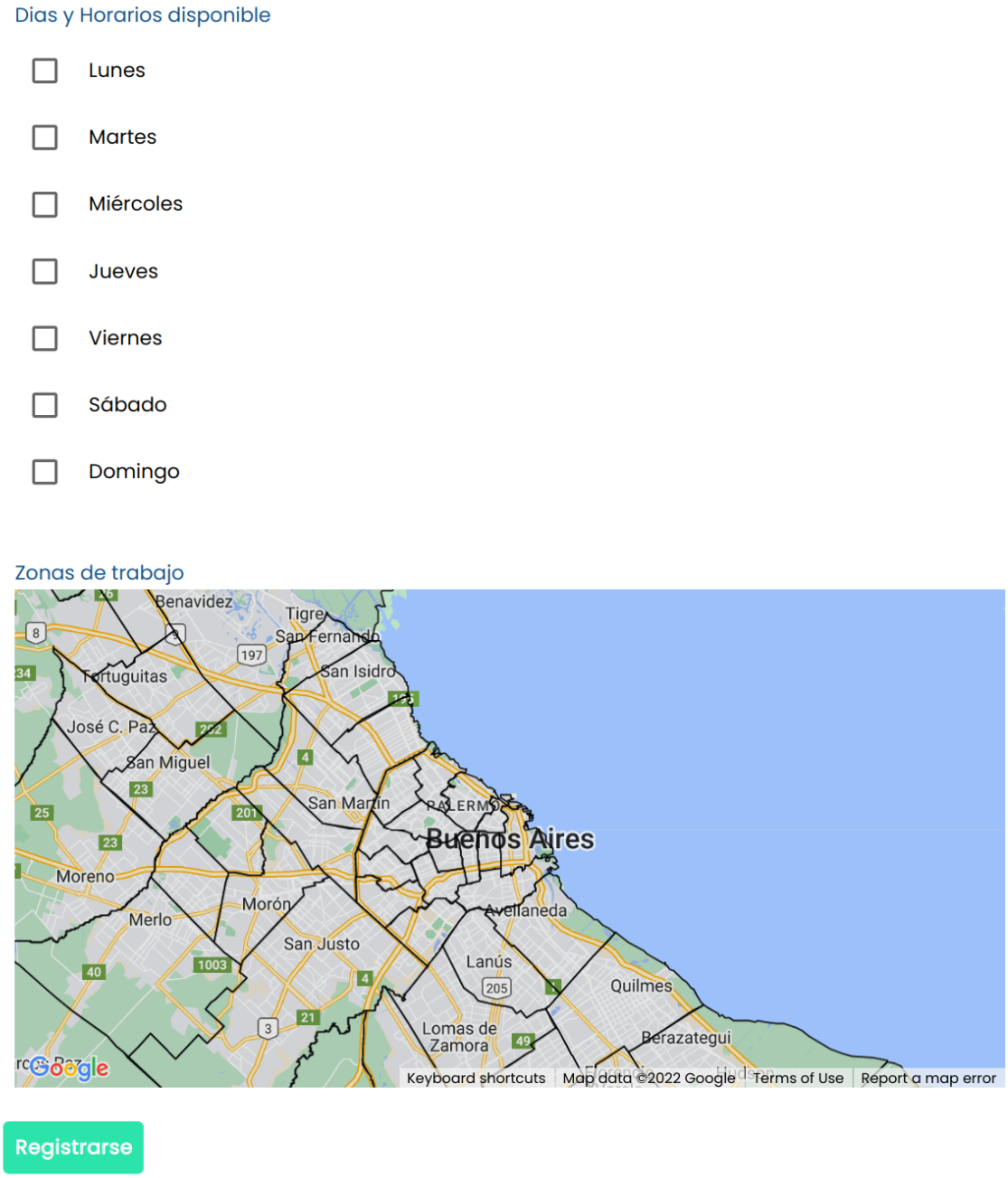


Figura 20: Formulario de registro tipo de usuario trabajador (2)

En este formulario el usuario trabajador debe completar sus datos personales, una descripción del mismo, en la cual puede especificar sus métodos de trabajo, puede describirse, etc. También debe seleccionar los medios de pago que acepta, y tiene la posibilidad de subir una foto de perfil. También se le pide seleccionar el rubro en el cual trabaja, y la especialidad en el mismo. Pueden seleccionarse más de un rubro, y más de una especialidad en el mismo, como podemos ver en el ejemplo de la figura 21.

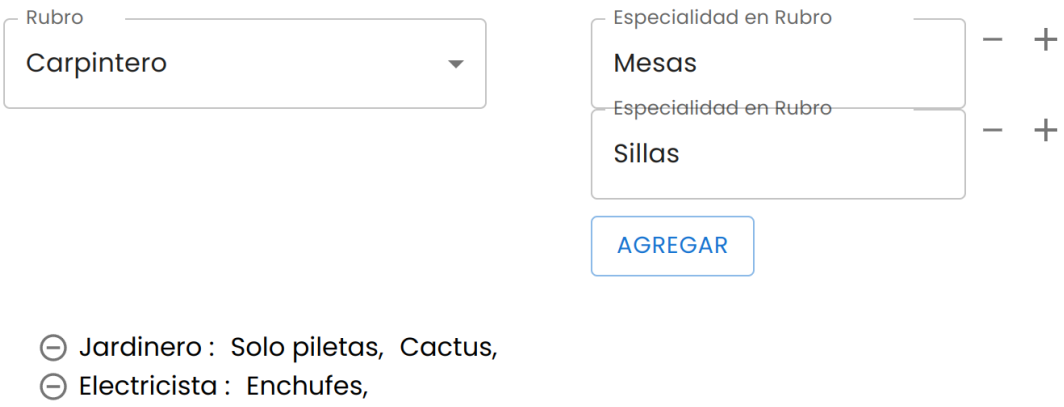


Figura 21: Descripción de rubros

Al seleccionar días disponibles, se despliega un menú en el cual el usuario puede seleccionar en qué momento del día está disponible. Por ejemplo, un cerrajero podría especificar que trabaja todos los días, las 24hs, para urgencias.

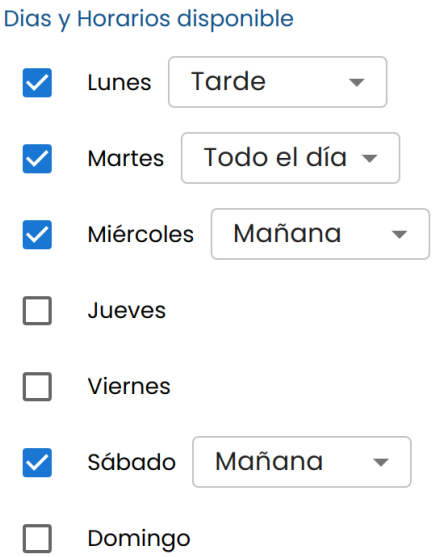


Figura 22: Descripción de días y horarios disponibles

Para que el trabajador especifique en qué lugares trabaja, se le presenta el mapa de la Provincia de Buenos Aires (figura 23). En el mismo, el usuario puede clickear en qué lugares trabaja, esto le brinda a los usuarios una visualización más clara, a comparación de lo que podría ser una lista de lugares.

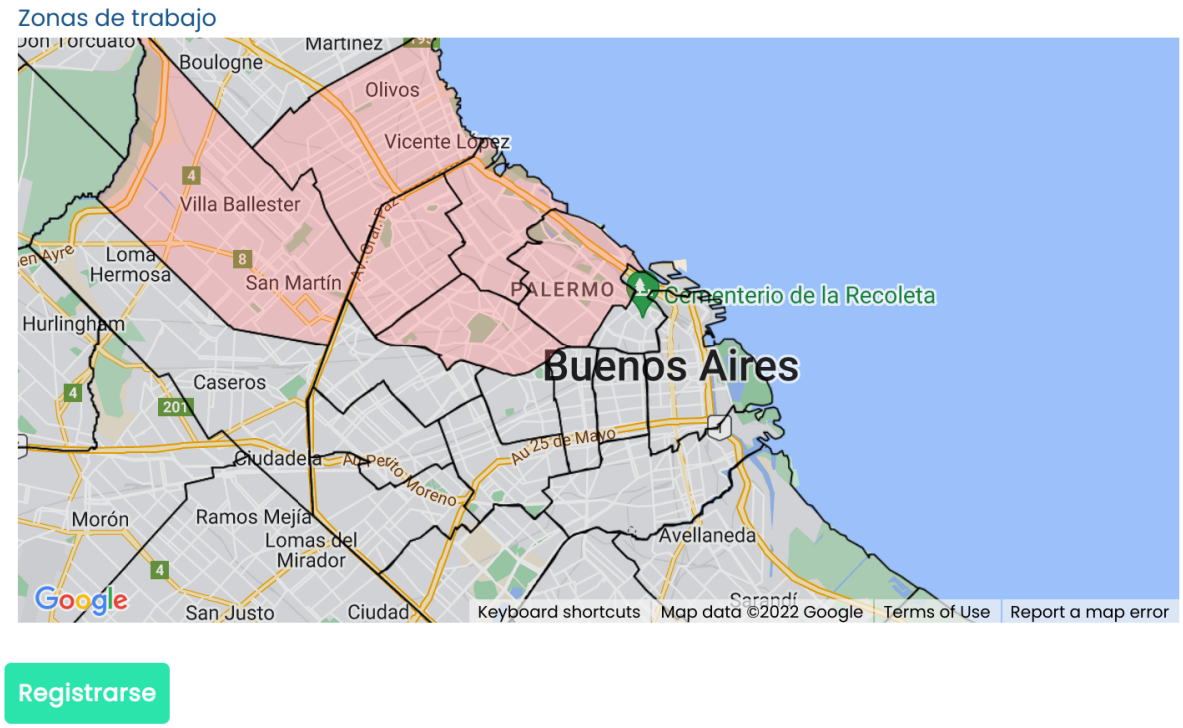


Figura 23: Selección de lugares de trabajo

Al presionar el botón de REGISTRARSE, y si no hay problemas con el formulario, se le muestra al usuario el mismo cartel de éxito que a un usuario vecino o administrador y se le envía el email para que el mismo active su cuenta.

Con respecto a la lógica del back-end, se validan todos los formularios de todos los tipos de usuarios (tanto en el front-end como en el back-end) corroborando que los campos sean válidos. Una vez que todo es correcto, se hashea la contraseña utilizando un salt y se genera el modelo de *usergalleries*, encargado de contener las imágenes del usuario y ahí se le envía un email con un token JWT (distinto al de inicio de sesión) que contiene información no sensible para la generación del usuario. Cabe destacar que las imágenes se guardan en la colección de *usergalleries*. En el caso de ser trabajador, los rubros (parte de un modelo) están incluidos en el JWT que es enviado por email.
En esta primera versión, los usuarios se persisten en la base de datos una vez que la cuenta se active, es decir, una vez que los usuarios seleccionan el ACTIVAR CUENTA desde el email que se les envía. Lo único que se persiste antes de la activación es la imagen de perfil (si es que se seleccionó alguna).

5.3 Login

A continuación se explica como es el funcionamiento tanto del lado de front-end como del back-end para los distintos tipos de usuarios.

En el lado del back-end todos los tipos de usuarios (Administrador de un grupo, un vecino o un trabajador) consumen de un mismo endpoint. Para realizar el acceso a la cuenta, la contraseña se guarda hasheada en la base de datos y a la hora del login se compara utilizando la librería de bcrypt, una función de hash muy utilizada para contraseñas.

El mismo genera un JSON Web Token que tiene información relevante de que tipo de usuario es (en el caso de ser un administrador y vecino simultáneamente se generan con ambos parámetros) y se firma con una variable de entorno solamente accesible desde el lado del servidor. Luego, la respuesta a la llamada del front end contiene este JWT con el tipo de usuario y el email.

Por el lado del diseño del front end para el inicio de sesión se realizó un formulario usando componentes de MUI v5 [15] y se contó con validaciones para ambos campos usando la librería Joi [12]. Cabe destacar que para los 3 tipos de usuario, el front end para el login es el mismo.

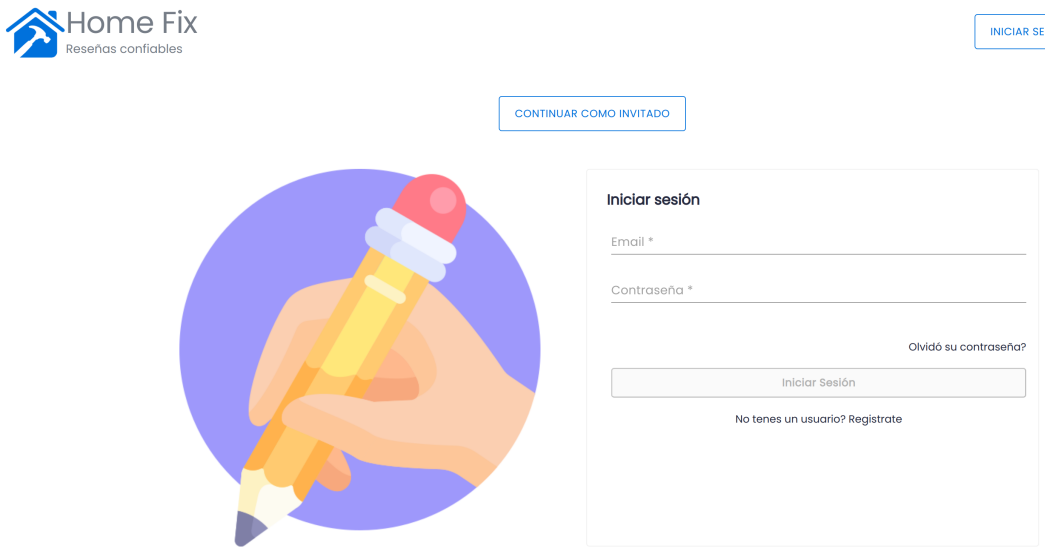


Figura 24: Formulario de inicio de sesión

De haber un error con alguno de los campos o del lado de la respuesta de la API, un label rojo se mostrará por debajo del input indicando el error asociado. En este caso, el email es incorrecto ya que no se registró un usuario con ese email.

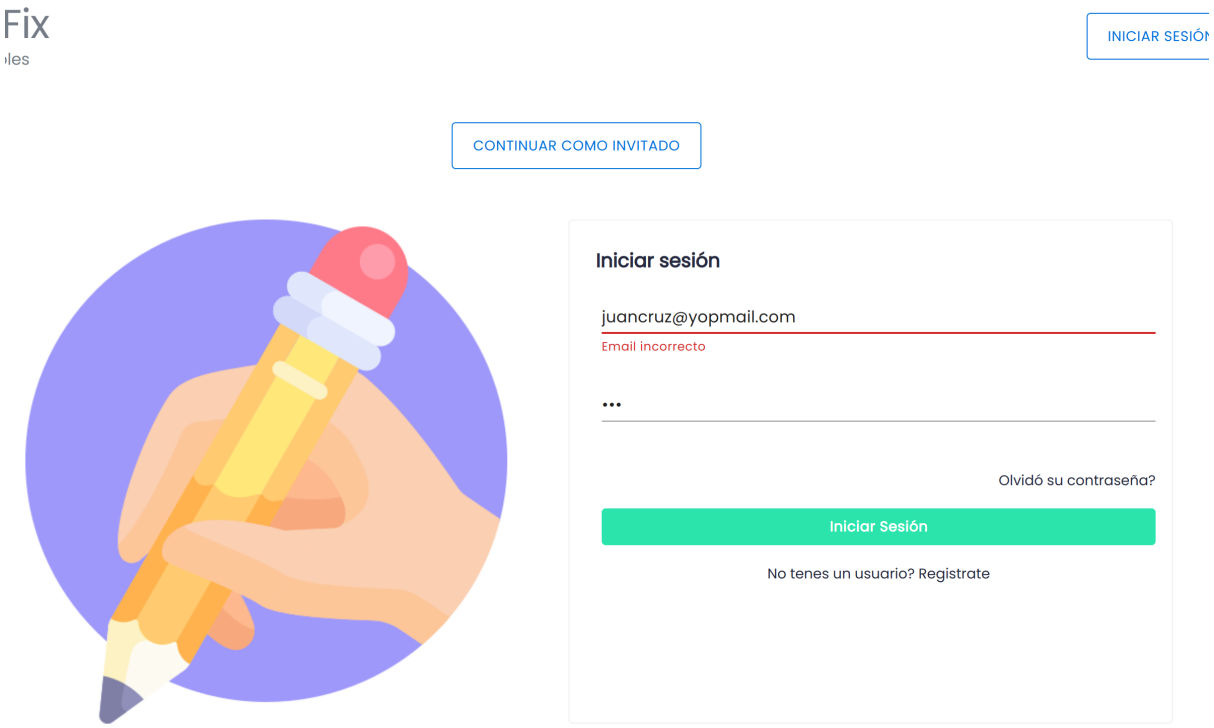


Figura 25: Formulario de inicio de sesión con error

En caso de que el inicio de sesión sea exitoso, el front end se va a encargar de guardar el JWT dentro del local storage que React provee, que tiene muchos mayores beneficios para el manejo de sesiones que las antiguas cookies de los navegadores. De esta manera, cada vez que se quiera saber quien es el usuario autenticado, se verifica el JWT, para poder tener la información básica del usuario. Finalmente, se redirigirá al usuario a la página de su perfil.

5.3.1 Recuperación de contraseña

Si el usuario olvida la contraseña para ingresar a HomeFix el mismo puede recuperarla.

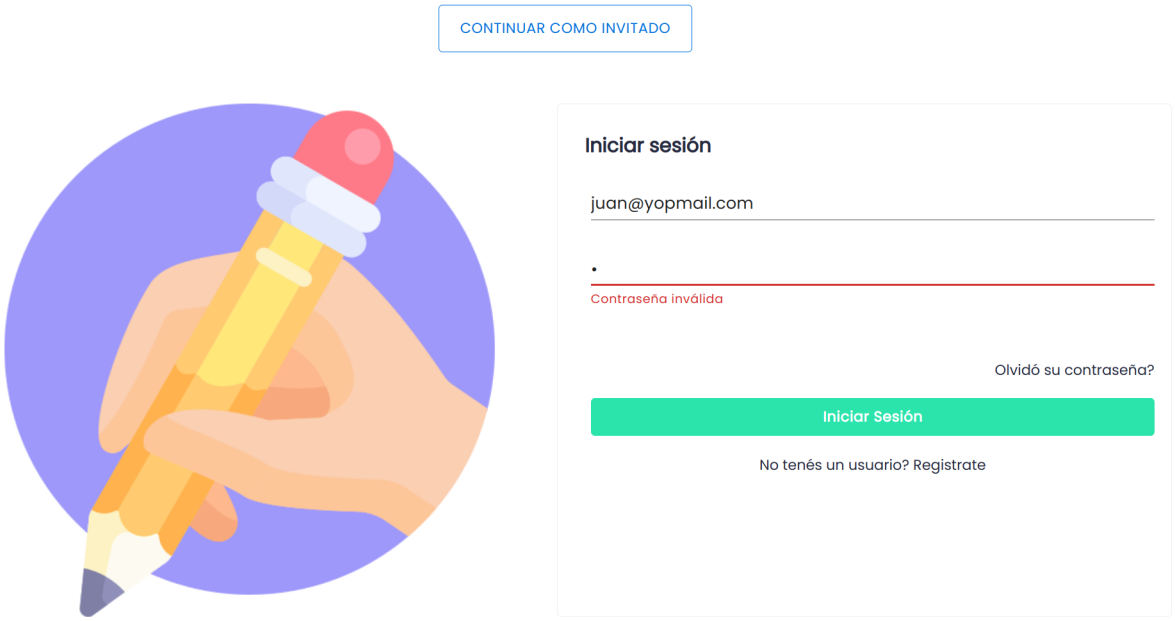


Figura 26: Contraseña inválida para el inicio de sesión

Si se clickea sobre “Olvidó su contraseña”, se le redirige a la página de la imagen 27, en la cual el usuario debe ingresar su email.

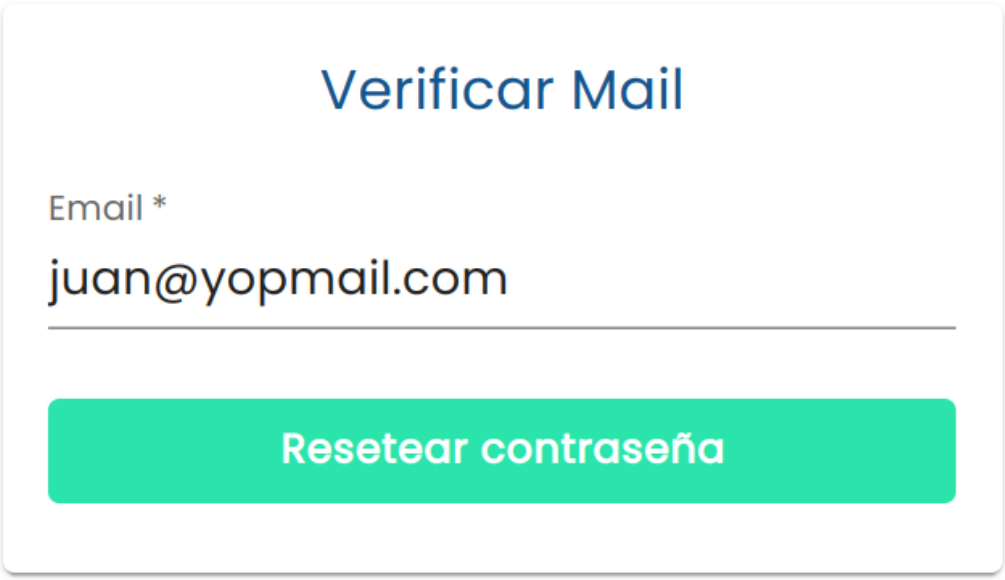


Figura 27: Ingreso de email para reseteo de contraseña

Una vez presionado el botón, el usuario recibirá el email de la figura 28.

Figura 28: Email de reseteo de contraseña

Al clickear sobre RESTAURAR CONTRASEÑA, se redirige al usuario a la página 29, en la cual ingresa la nueva contraseña dos veces, para evitar conflictos con los posibles errores de tipeo.

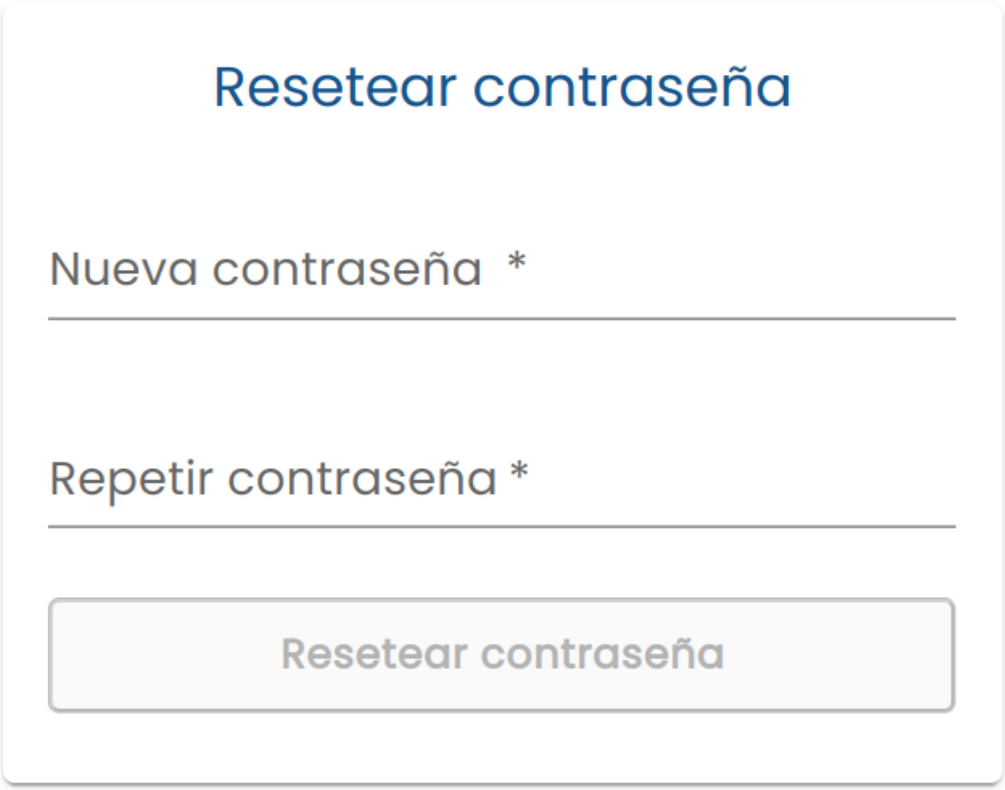


Figura 29: Formulario de reseteo de contraseña

Una vez clickeado el RESETEAR CONTRASEÑA, aparece el mensaje de éxito de la figura 30, el cual luego redirige al usuario a la página de logIn nuevamente.

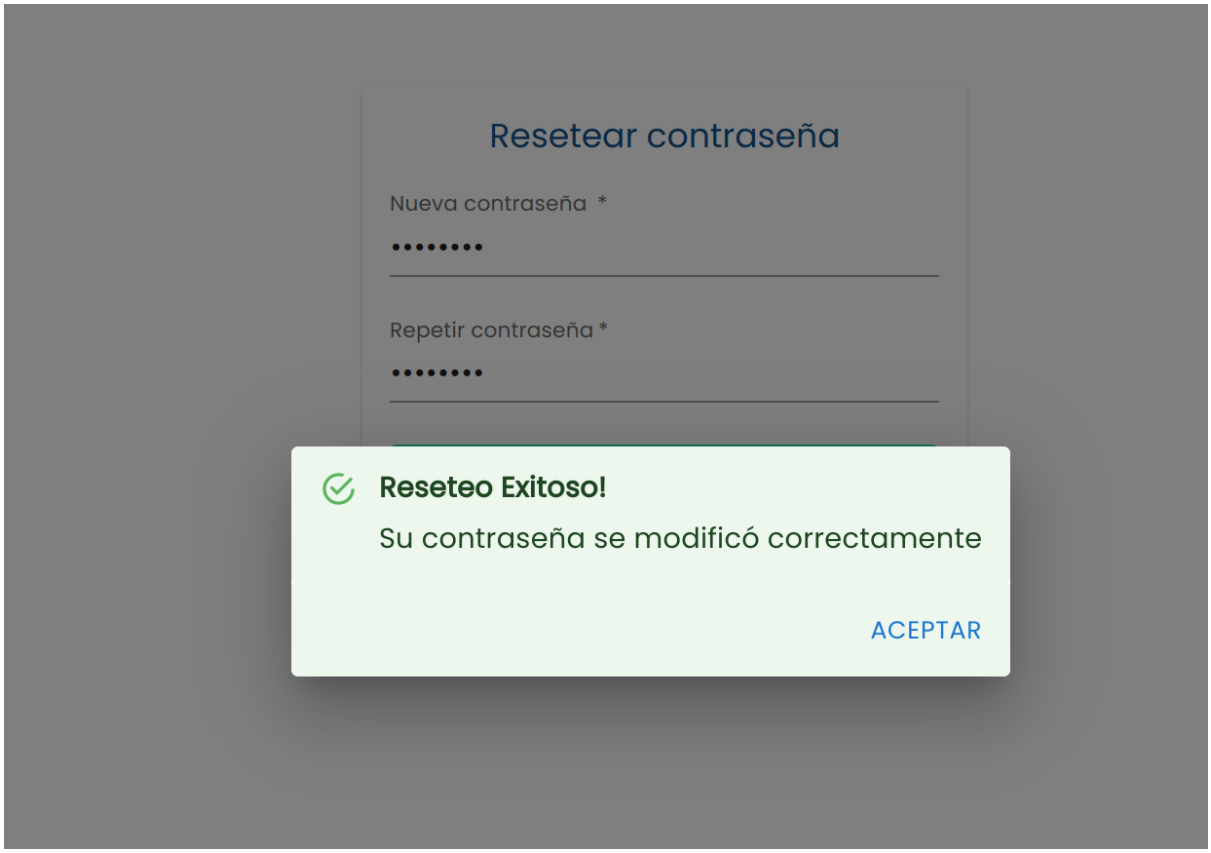


Figura 30: Cartel de éxito al resetear contraseña

Si por algún motivo se quiere usar nuevamente el mismo email enviado para resetear la contraseña, el mismo tira un error, ya que fue utilizado en el pasado.

Figura 31: Error al resetear contraseña

Por parte del back-end, en el modelo del usuario hay un campo llamado ResetLink, el cual inicialmente se encuentra vacío. Este campo está reservado para guardar el token que será generado al momento que un usuario quiera cambiar su contraseña. Este token es único, ya que es generado a partir del *id* del usuario y permitirá que el back-end valide la autenticidad del usuario.

Cuando el usuario clickea sobre *"Olvidó su contraseña?"* de la figura 26, se lo redirige al formulario de la figura 27 para que complete el email de la cuenta que desea restablecer.

Una vez presionado el botón de *"Resetear contraseña"* de la figura 27, se genera el token y se guarda en el campo ResetLink del modelo del usuario.

A continuación, el usuario recibe un email que le permite navegar al formulario de la figura 29. Cuando el usuario presiona el botón de *"Resetear contraseña"* de la figura 29, se le envía al back-end la nueva contraseña del usuario y el token mencionado anteriormente. Este se compara con el valor del campo ResetLink guardado en el modelo del usuario y si coinciden se guardará la nueva contraseña provista por el usuario.

Finalmente, se resetea el campo ResetLink del modelo del usuario al valor original *(vació)*, para evitar un ataque de replay si se quiere volver a usar el mismo email enviado.

5.4 Búsqueda de trabajadores

La aplicación presenta un buscador de trabajadores, por nombre y/o apellido del mismo y descripción.
Si se busca un trabajador sin iniciar sesión, puede verse únicamente la barra de Buscar en el header. En este caso se listan los trabajadores, de todas las categorías, que contengan la letra P.

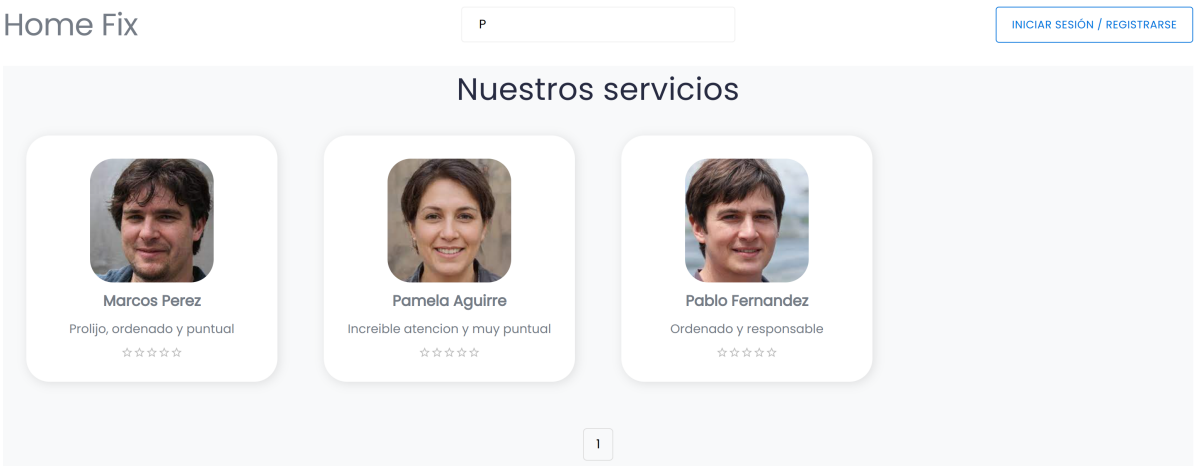


Figura 32: Búsqueda

Si se desea buscar trabajadores de una categoría en particular, se puede ingresar a esa categoría y buscar por el nombre que se desee. En el siguiente ejemplo (figura 33), se listan carpinteros con la letra P.

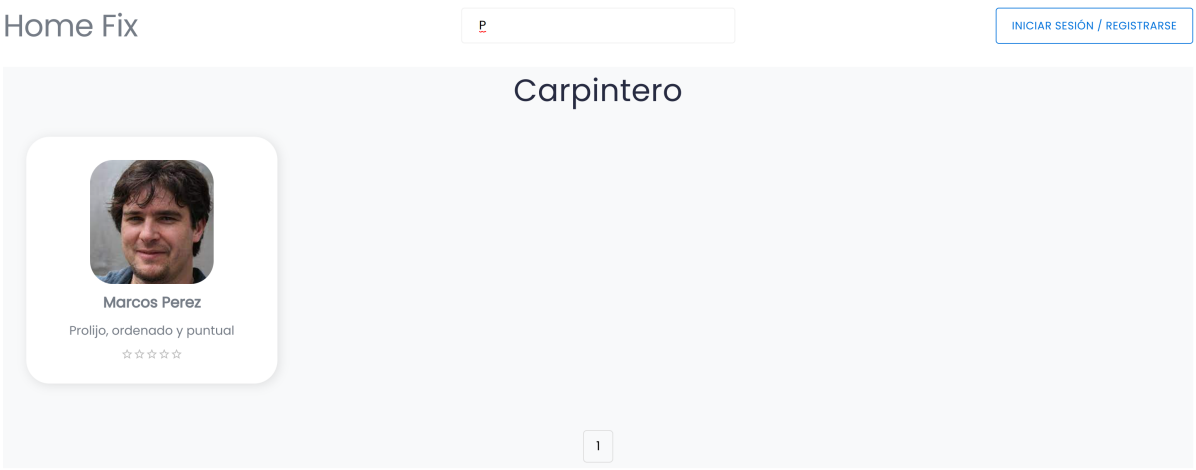


Figura 33: Búsqueda de Carpinteros

5.4.1 Selección de categoría

Si uno inicia sesión con un perfil de vecino o administrador de vecinos, puede observar un cambio en el Header (a comparación de la figura 32): se agrega un selector, para que el usuario pueda seleccionar y buscar por grupo de vecino. Esto significa que los trabajadores van a organizarse de manera diferente, primero se listan los empleados que hayan sido contratados por alguien del grupo seleccionado, luego los empleados cercanos al grupo seleccionado y luego el resto. En la figura 34 puede observarse que, si ahora se inicia sesión y se selecciona el grupo “Sur” y se busca nuevamente carpinteros con la letra M, se observa una lista organizada de manera diferente.

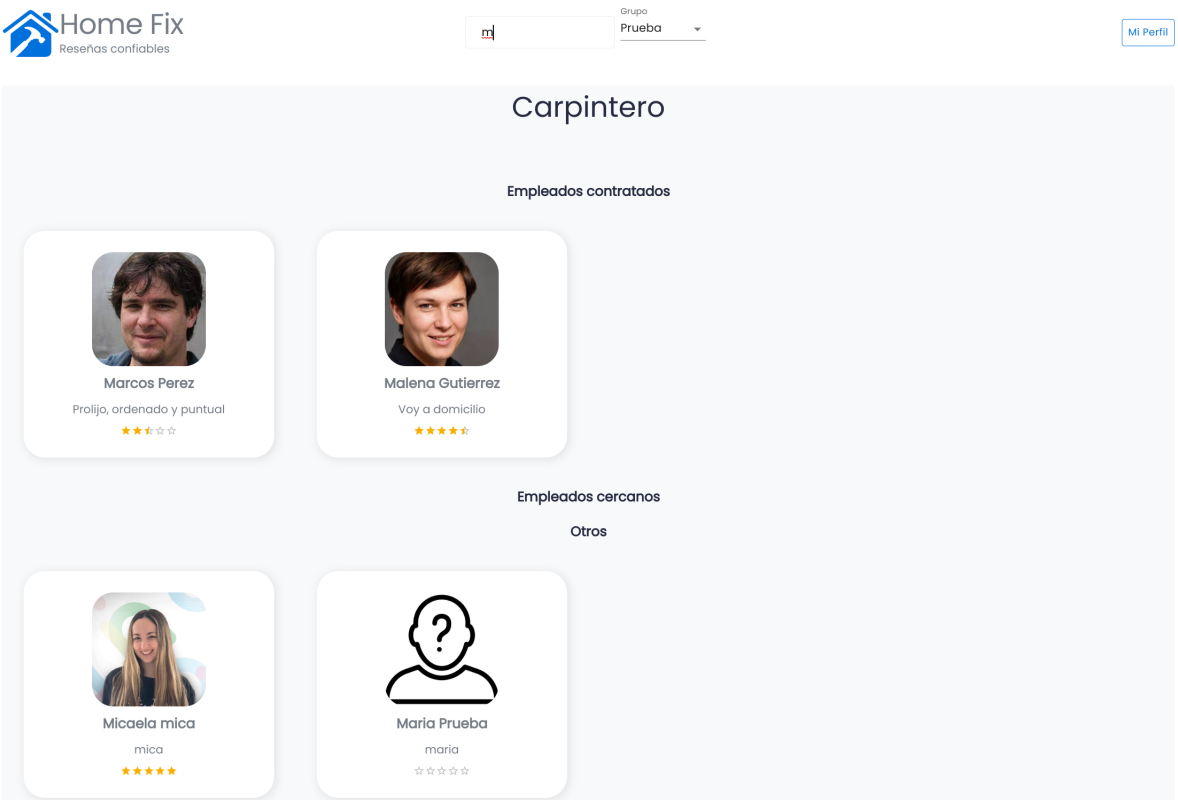


Figura 34: Búsqueda de Carpinteros por barrio

Con respecto al diseño del back-end, todos los filtros de búsqueda que ocurren son a través de los query parameters. Estos son número de página (para paginación), rubro al cual refiere, campo del buscador y el vecindario (presente únicamente si se inicia sesión con un perfil vecino o administrador).
El buscador se encarga de matchear tanto por nombre, apellido o descripción, si existe rubro, trae a todos los trabajadores de ese rubro. Claramente, se muestran únicamente trabajadores que no estén en la blacklist. Todas las búsquedas presentan paginación.
Si el usuario es vecino o administrador, se encuentra autenticado y selecciona el grupo por el cual quiere buscar, podemos dividir la lógica en 3 subqueries. La respuesta de esta contiene 3 listas: trabajadores que trabajaron para vecinos del grupo seleccionado, trabajadores que trabajan en la zona del grupo seleccionado, y por último el resto.
Si el usuario no está autenticado o es un trabajador, se utilizan los filtros de la búsqueda, sin nada en particular.

5.5 Perfil Trabajador

En esta sección se mostrarán las diferentes vistas de un perfil trabajador.

5.5.1 Vista sin login

En la figura 35 podemos ver la vista de un perfil de trabajador al realizar la búsqueda. Podemos observar sobre la izquierda, la foto de perfil del trabajador, nombre y apellido del mismo, su descripción, la última vez activo del trabajador, sus estrellas, las cuales indican nivel de satisfacción de otros usuarios que la contrataron, y reseñas de estos usuarios que lo hayan contratado. La última vez activo indica la última vez que el trabajador entró a sus conversaciones. Esto fue realizado para ayudar a los usuarios a ver si el perfil es activo, o si hace mucho que no entra a la plataforma, un dato relevante para que los usuarios evalúen si contactarlo o no.

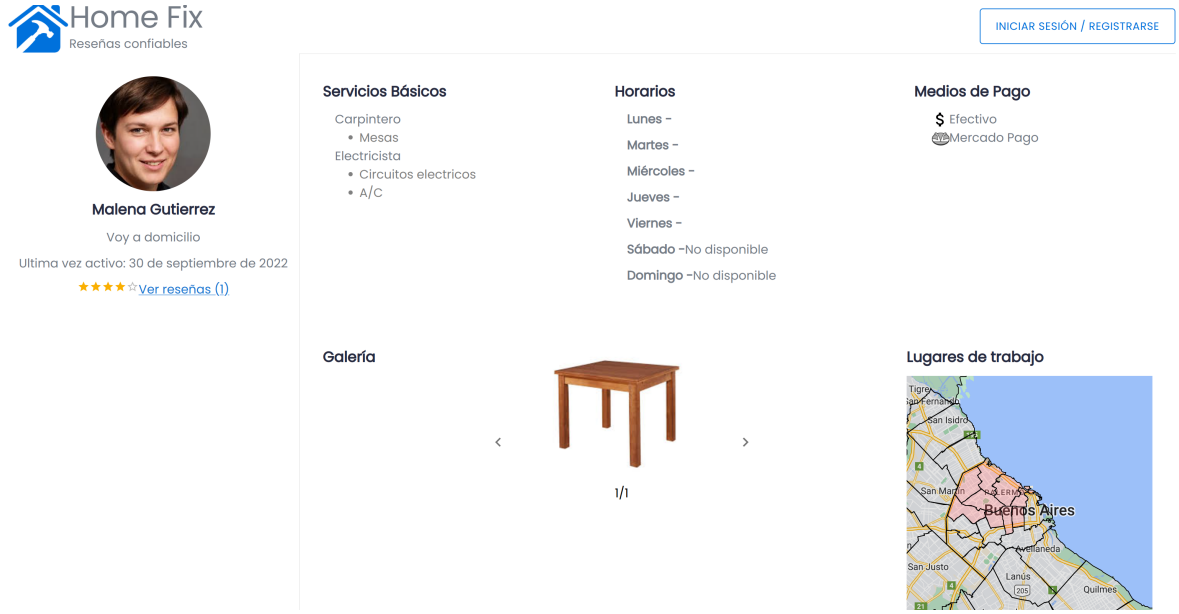


Figura 35: Perfil Trabajador

Si se clickea sobre Ver reseñas, se observan las mismas en la figura 36, en donde se ven cuantos likes o dislikes tiene la reseña.

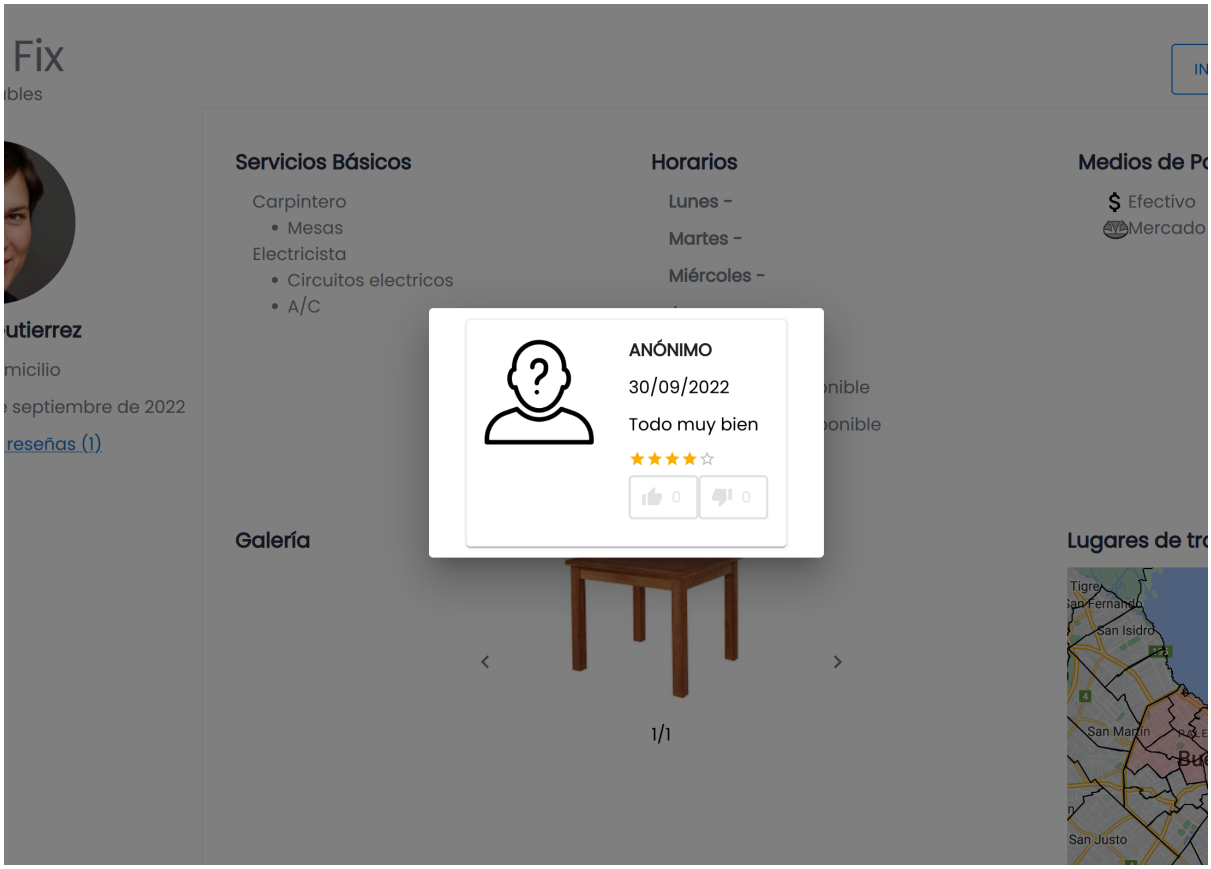


Figura 36: Reseñas Perfil Trabajador

Puede observarse que las reseñas son anónimas, ya que al no haber iniciado sesión, no se conocen esos vecinos. Esta decisión fue tomada para respetar la privacidad de los mismos.
El puntaje presentado en el rating, en forma de estrellas, es un promedio de todas las reseñas que presente el trabajador. Un usuario no puede darle *me gusta* o *no me gusta* su propia reseña, y los mismos son únicos por usuario por reseña.

Si se retoma la vista general del trabajador de la figura 35 , se observan los servicios básicos que el trabajador presenta, los horarios en el cual trabaja, los medios de pago que acepta, galería de imágenes con trabajos anteriores y un mapa con los lugares de trabajo.

5.5.2 Vista con log in

Si se inicia sesión con el usuario trabajador, se observa en la figura 37 una vista muy similar a la sin login (figura 35), pero se verá el botón de editar perfil, el cual permite editar toda la información del usuario a excepción del email, y se verán dos pestañas nuevas, una con contrataciones y otra con conversaciones.
También se puede observar un nuevo botón bajo la Galería del trabajador. Este permite

seleccionar y subir imágenes de trabajos anteriores. Las mismas se van a subir al modelo de *usergalleries* del trabajador, que funciona como un storage de las imágenes para los usuarios.

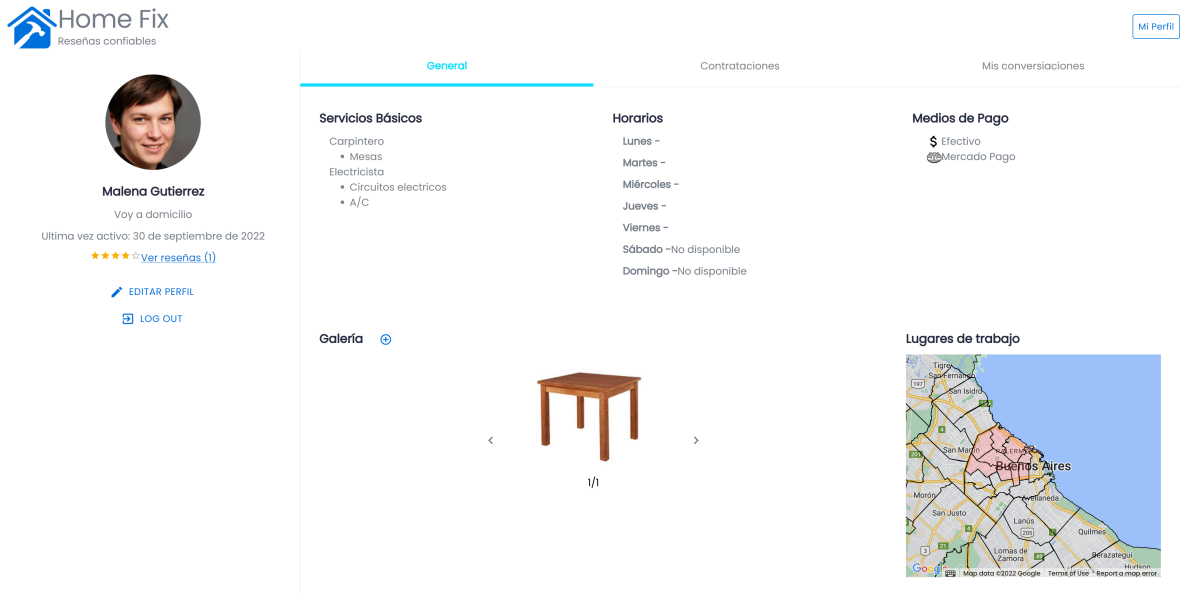


Figura 37: Perfil Trabajador Log In

Si se clickea sobre las reseñas, ahora estando autenticado con un perfil vecino, podemos observar en la figura 38 como una reseña es anónima, ya que esa persona no es vecino del usuario, y otra reseña no es anónima, ya que ese usuario si es vecino del usuario autenticado. Estas reseñas se listan primero por vecinos, luego por anónimos.
También, ahora autenticados, se muestra la opción de reportar la reseña. Si la misma tuviera 20 reportes, se levanta un flag y deja de aparecer en el perfil de los trabajadores, y le aparece al superadmin, para que él decida si la reseña es inapropiada y deje de mostrarse, o no la es, y la misma volverá a ser mostrada en el perfil del trabajador.

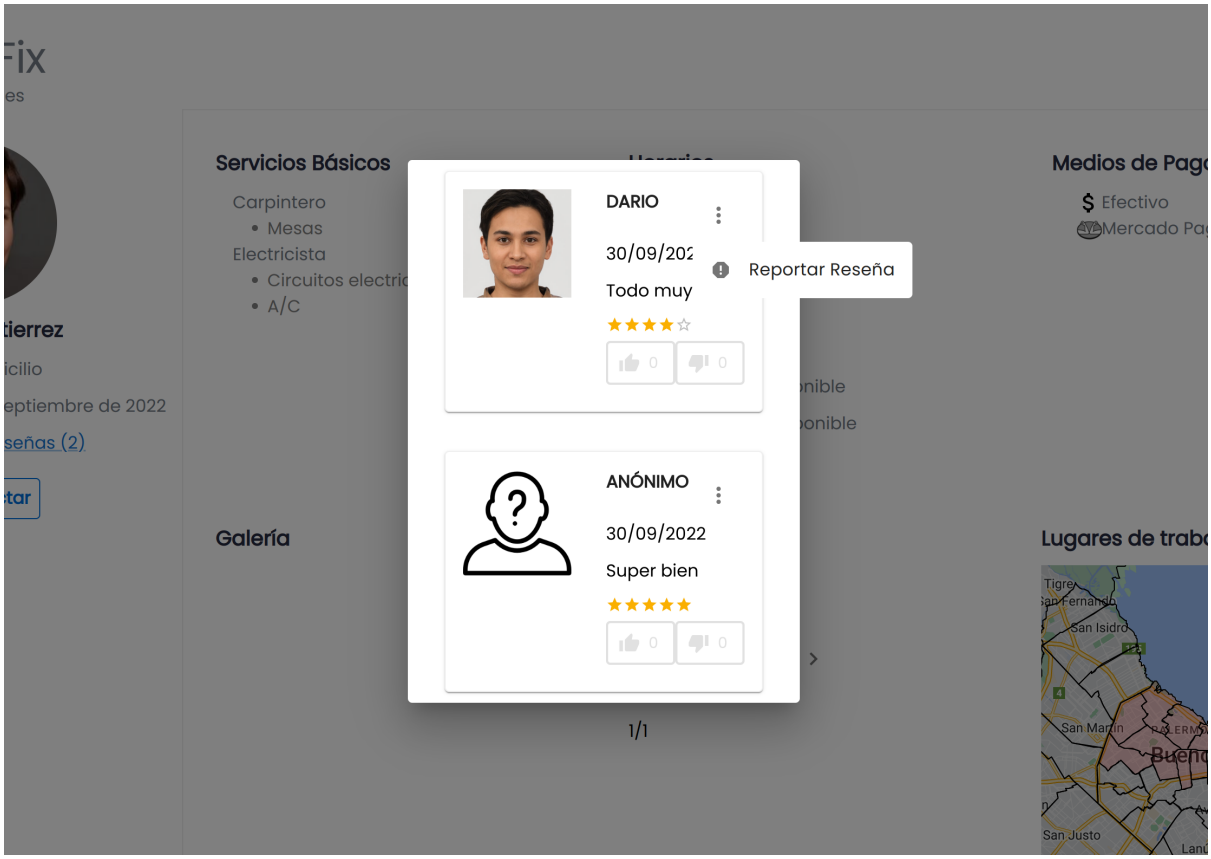


Figura 38: Reportar Reseña

Si por algún motivo el super administrador desea reportar la cuenta de un trabajador, y este intenta iniciar sesión, se le mostrará el cartel de la figura 39, para que el mismo se contacte con HomeFix y pueda solucionar su situación.

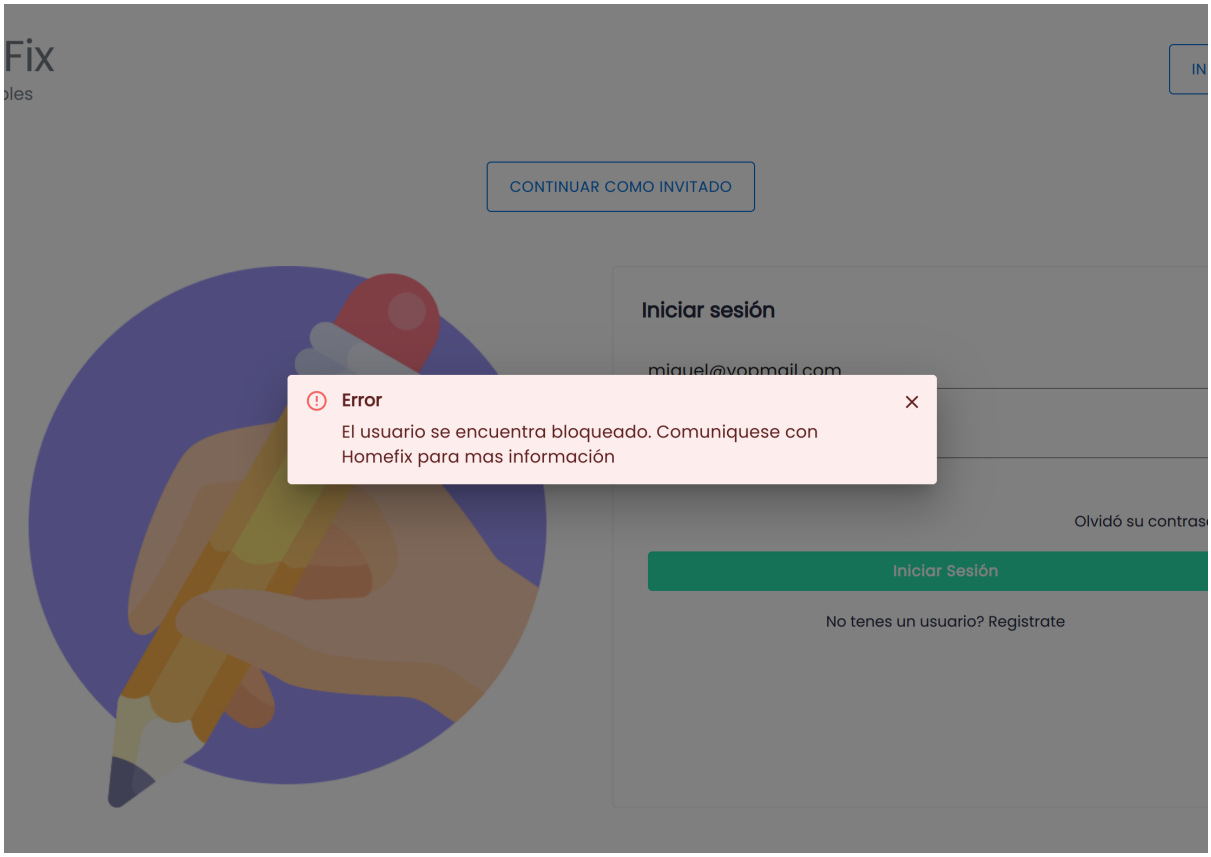


Figura 39: Log In bloqueado

5.5.3 Contrataciones

Al entrar en la pestaña de contrataciones, se verá una vista como la figura 40, la cual tiene un selector para los distintos estados de las contrataciones (En espera, aceptadas y finalizadas) y se listan las mismas con su información.

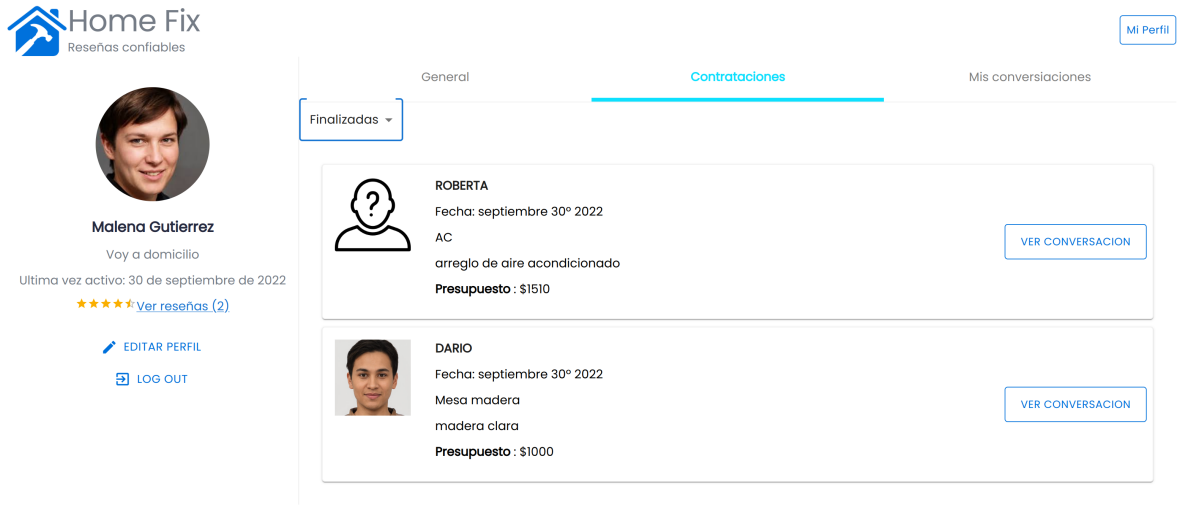


Figura 40: Contrataciones Finalizadas

5.5.4 Conversaciones

Con respecto al back-end, las conversaciones abren los sockets explicados en la sección de back-end4.6 y diseño de alto nivel.

Si se selecciona la pestaña de conversaciones, se listan las que tenga el usuario trabajador con los vecinos y se le da la opción de que el trabajador le envíe un presupuesto al vecino.

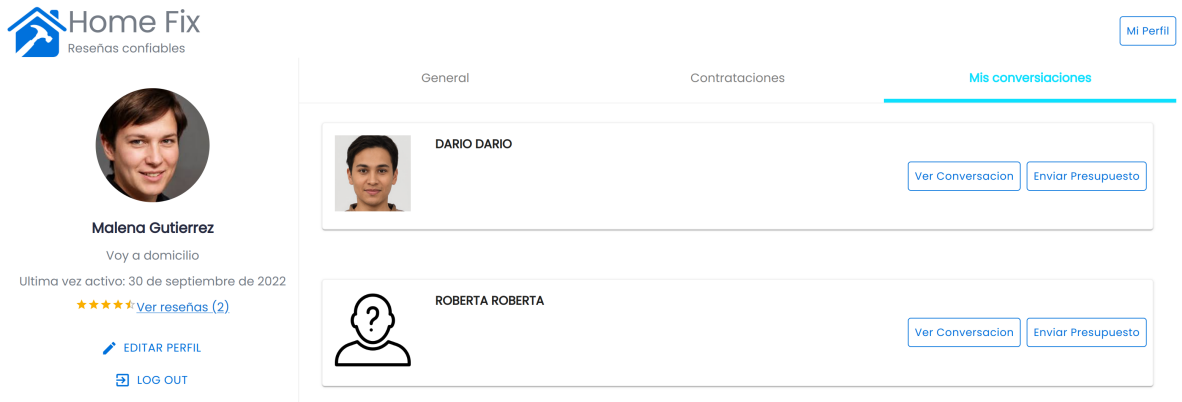


Figura 41: Lista de Conversaciones

A diferencia de un chat desde la vista de vecino, a la vista del trabajador se le añade el

botón de enviar presupuesto, para que el mismo le envíe al vecino.

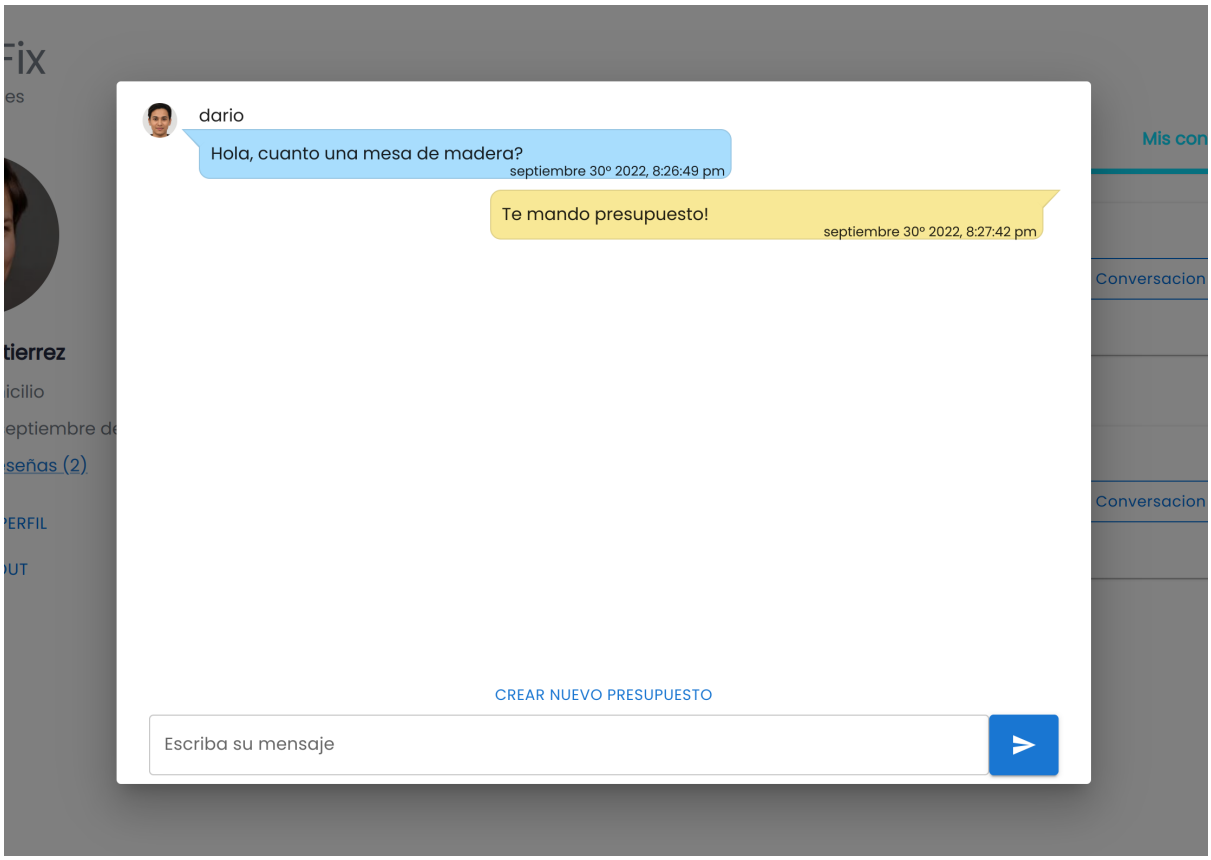


Figura 42: Chat

5.5.5 Presupuestos

Si el trabajador desea enviarle un presupuesto a un vecino, puede seleccionar el botón en la vista de mis conversaciones, o directamente desde el chat. En la imagen 43 se muestra el formulario para que el trabajador rellene la información necesaria para comenzar un presupuesto. Se tiene que completar un título y una descripción del trabajo el cual se va a realizar, el precio del trabajo, y el rubro del mismo. Si todo sale bien, se le muestra un mensaje de éxito al usuario.

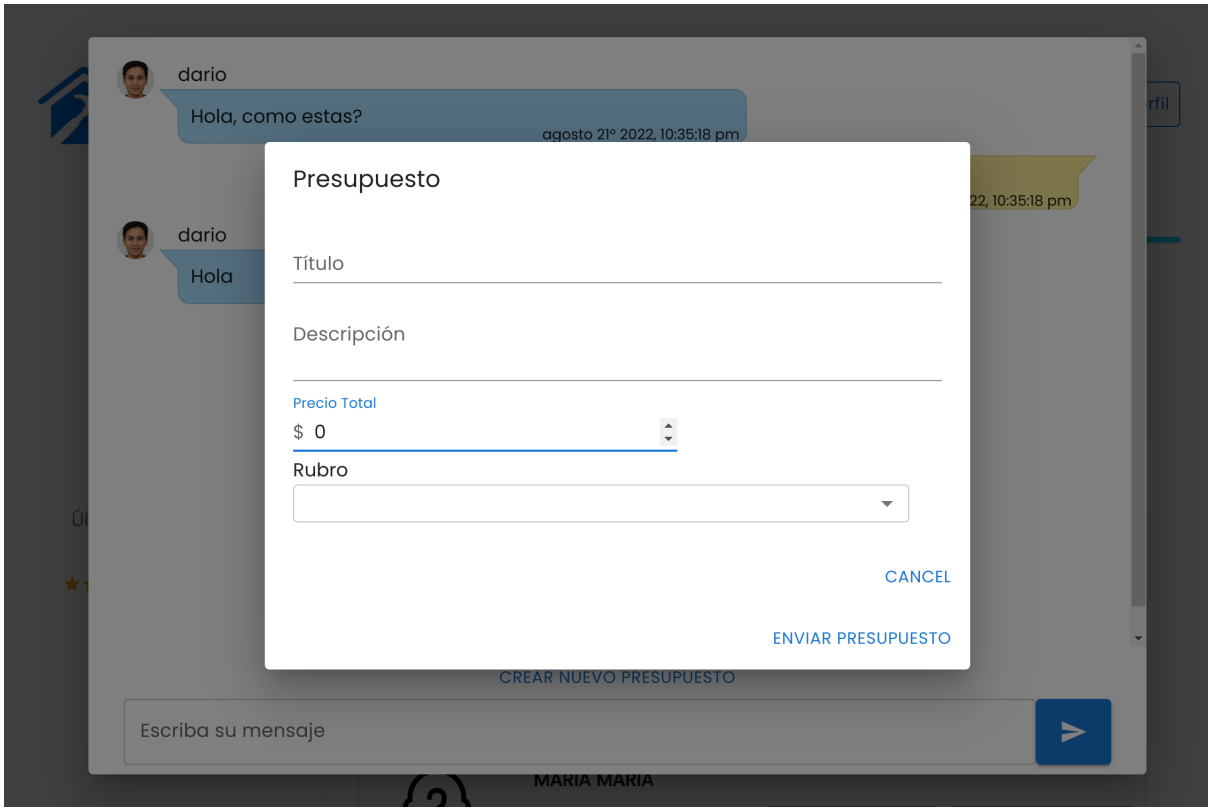


Figura 43: Presupuesto

5.6 Perfil Vecino

En la siguiente sección se mostrarán las diferentes vistas relacionadas al perfil de un usuario vecino.

5.6.1 Mi información

Al iniciar sesión con un usuario que es vecino, se lo redirige a la página de su perfil. También el mismo puede dirigirse a su perfil clickeando sobre el botón ubicado en la parte superior derecha de Mi Perfil.

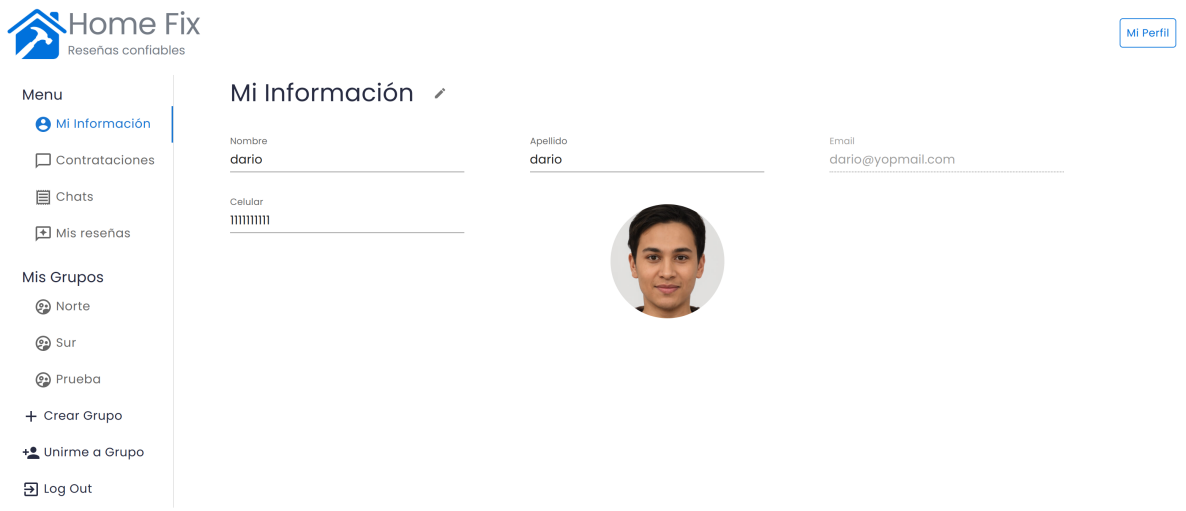


Figura 44: Perfil Vecino

Lo primero que este ve es su Información personal: Nombre, Apellido, Email, Celular y su foto de perfil, si es que selecciono una al crearlo. Puede observarse un lápiz a la derecha de Mi Información, el mismo es clickeable y hace referencia al Editar Información. Si se lo clickea puede verse el formulario de la figura 45, para poder modificar la información que se desee.

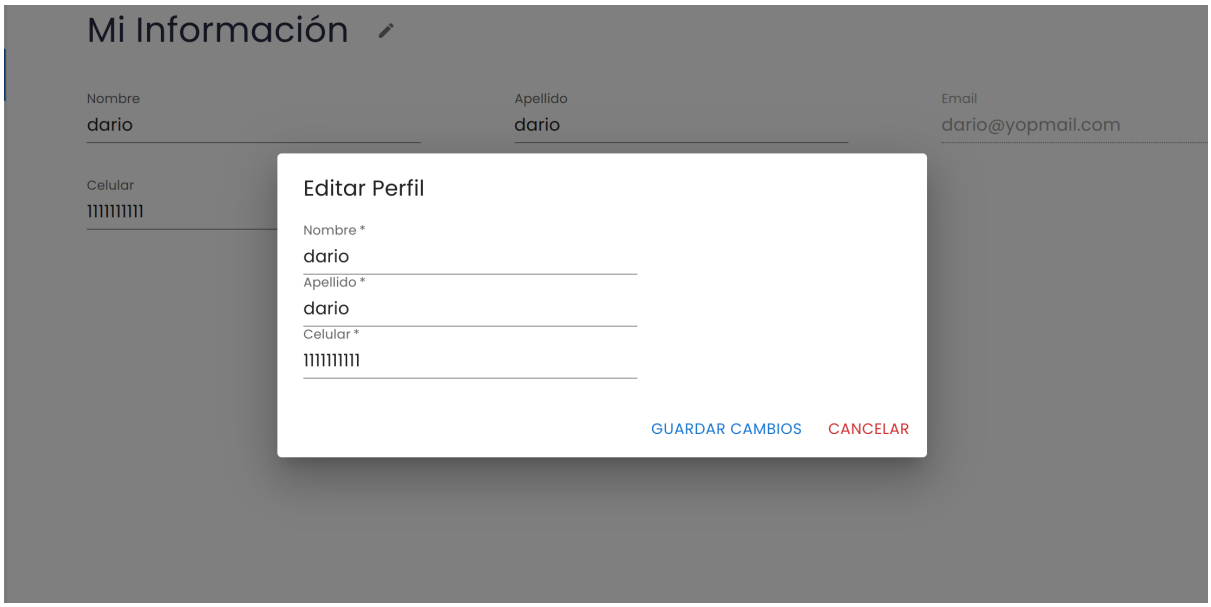


Figura 45: Perfil Vecino Información

Como se mencionó anteriormente, el email es el identificador único, por este motivo el mismo no puede modificarse. Si se hacen modificaciones y todo salió de manera exitosa,

se le muestra un cartel de éxito al usuario.
Para modificar la foto de perfil, el usuario puede pararse con el cursor sobre la misma y verá un texto de Cambiar Imagen. Si se lo presiona, se abrirá una pantalla con los archivos .jpeg y .png existentes en el dispositivo, para que el usuario seleccione el que desee, como nueva foto de perfil.

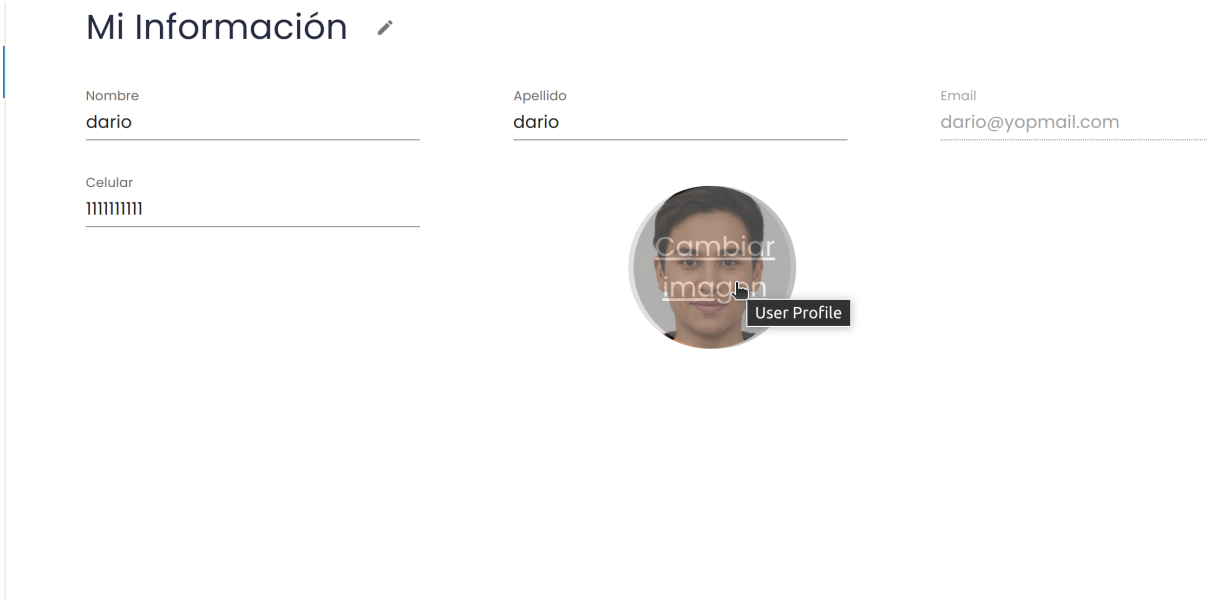


Figura 46: Perfil Vecino Foto Perfil

5.6.2 Contrataciones

Si miramos nuevamente la Figura 44, vemos en el menú de la izquierda otra pestaña que hace referencia a las contrataciones que tiene el vecino. Pueden verse 3 otras solapas para los distintos estados de las contrataciones.

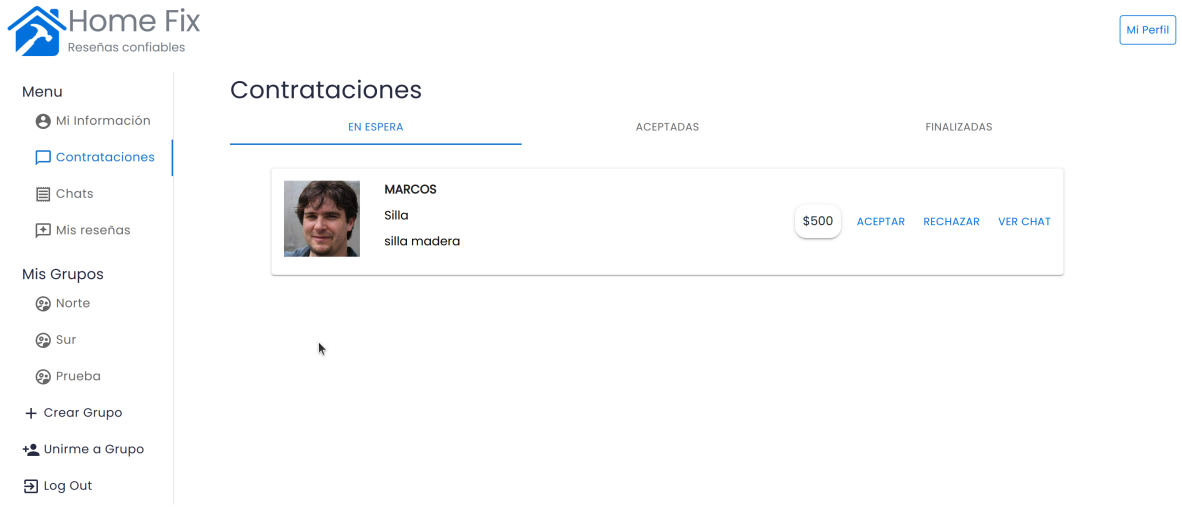


Figura 47: Contrataciones de Vecino en Espera

En la figura 47 puede observarse una contratación en espera, en la cual se muestra el nombre del trabajador, un título y una descripción de la contratación, el precio de la misma, y la foto de perfil del trabajador. Se pueden observar 3 botones, uno para aceptar la contratación si es que no hay errores, otro para rechazarla si se presentan errores, y otro para ver el chat con ese trabajador.
Si se acepta la contratación se observa un mensaje de éxito y podemos observar la misma en la pestaña de ACEPTADAS (figura 47). Cabe resaltar que es posible tener varias contrataciones en curso, con distintos trabajadores.

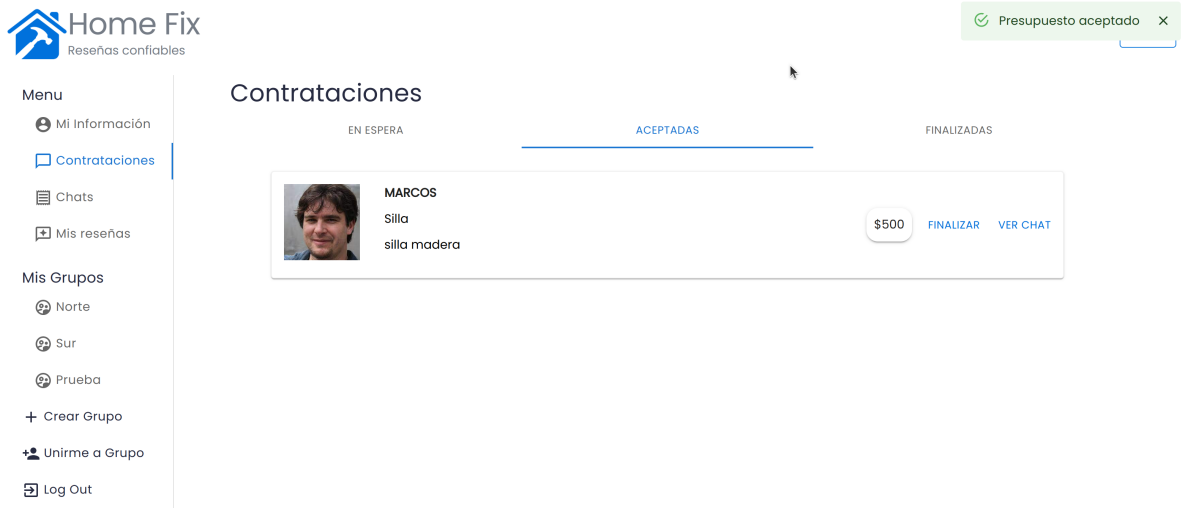


Figura 48: Contrataciones de Vecino Aceptadas

En la pantalla de la imagen 48 el usuario tiene la posibilidad de finalizar la contratación, si es que el trabajo fue terminado o de ver el chat con el trabajador. Si por algún motivo el trabajador no finaliza el trabajo, o el vecino no está conforme con el mismo, el mismo puede finalizar la contratación de todos modos, esta mueve la contratación al estado finalizado, dándole la posibilidad al vecino de escribir una reseña sobre el trabajador. Siempre que la finalización sea exitosa, y no haya algún error (por parte del servidor) se le mostrará al usuario un mensaje de éxito.

Si observamos la pestaña de contrataciones finalizadas (figura 49) vemos dos botones: ver el chat con el trabajador y el botón de escribir review.

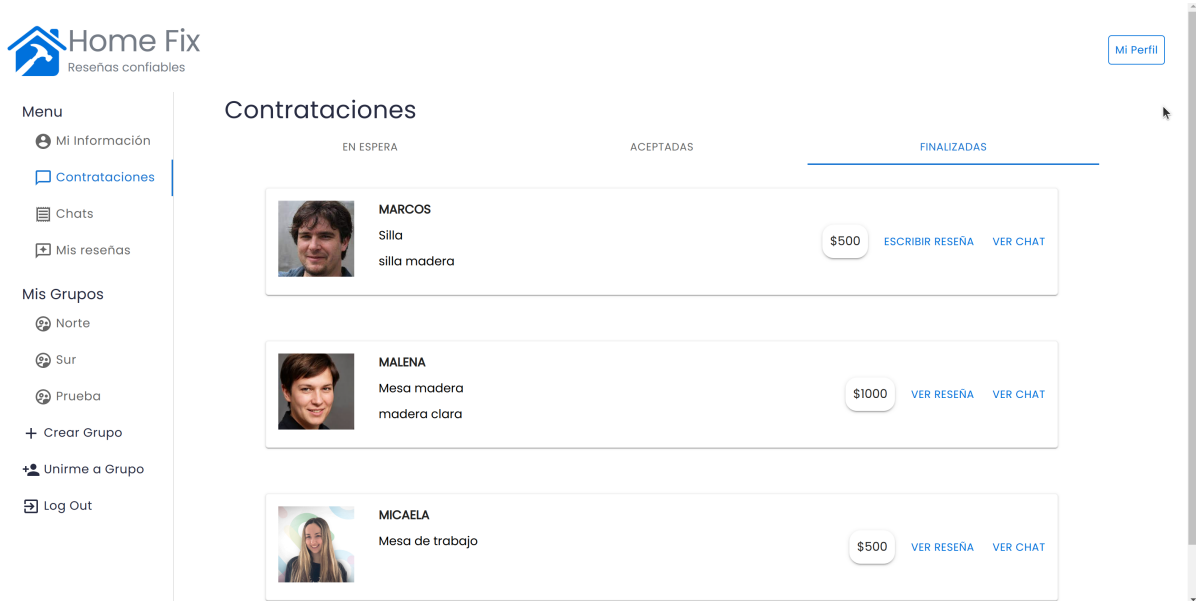


Figura 49: Contrataciones de Vecino Finalizadas

Si se desea escribir una reseña, se le mostrará al usuario el diálogo de la figura 50, en donde el mismo tiene la posibilidad de dar su opinión y puntuar con estrellas al trabajador. La opinión no es obligatoria, pero la puntuación con estrellas si lo es.

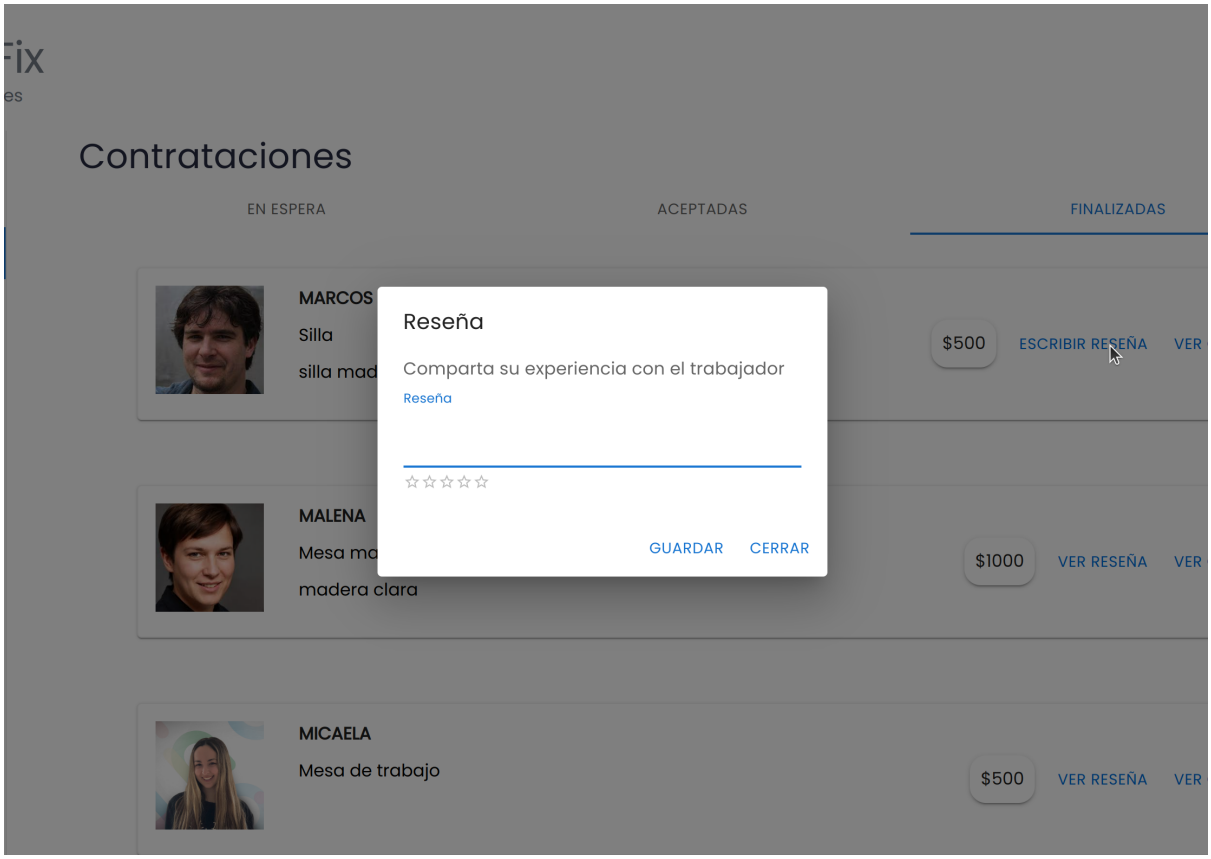


Figura 50: Escribir Reseña

Una vez completo, y si todo salió bien, se le muestra un mensaje de éxito al usuario y se le da la posibilidad de ver la reseña que escribió.

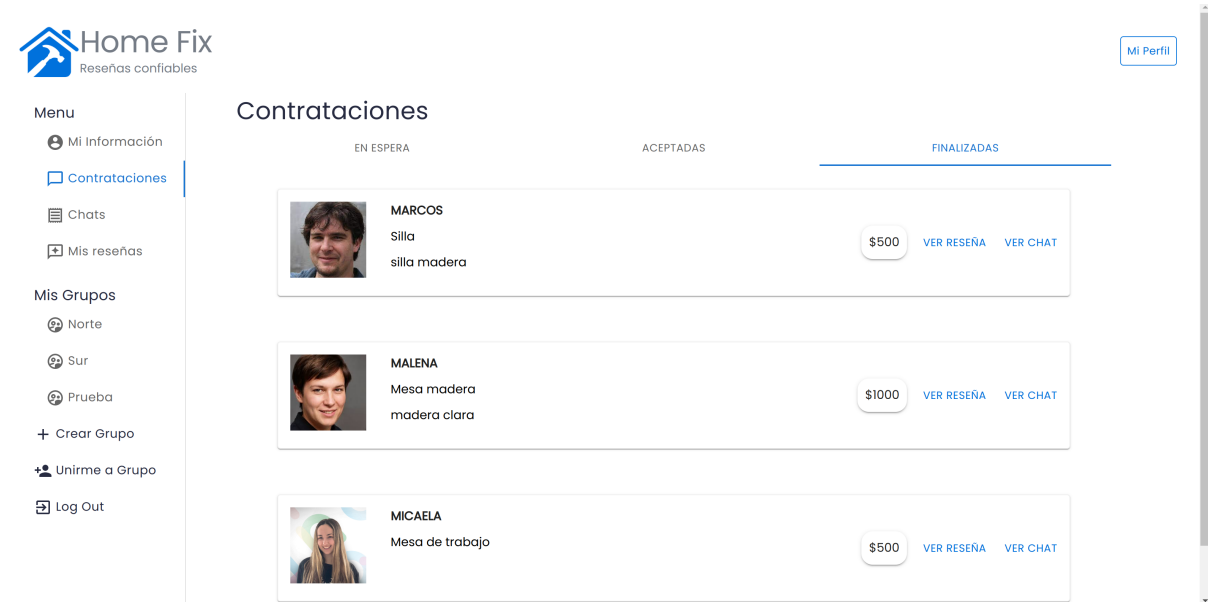


Figura 51: Contrataciones de Vecino Finalizadas

5.6.3 Chats

Otra opción es ingresar a la pestaña de Chats, en donde el usuario puede ver sus conversaciones con los trabajadores a los cuales haya contratado.

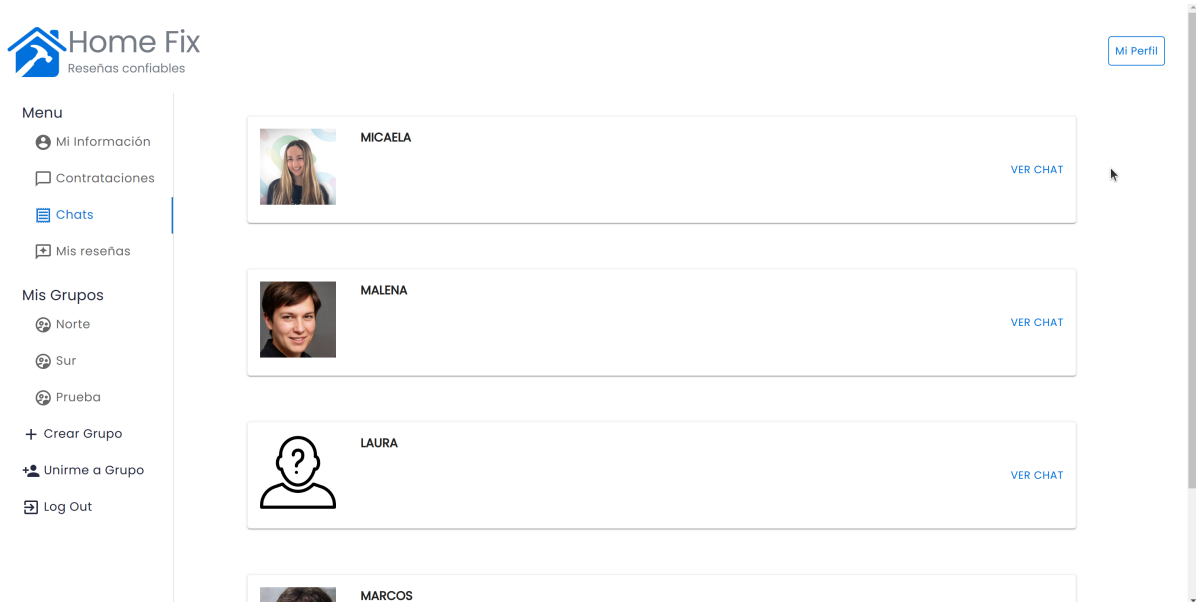


Figura 52: Chats vecino

Al clickear sobre VER CHAT, se muestra el chat con ese trabajador, con la opción por supuesto de enviar mensajes.

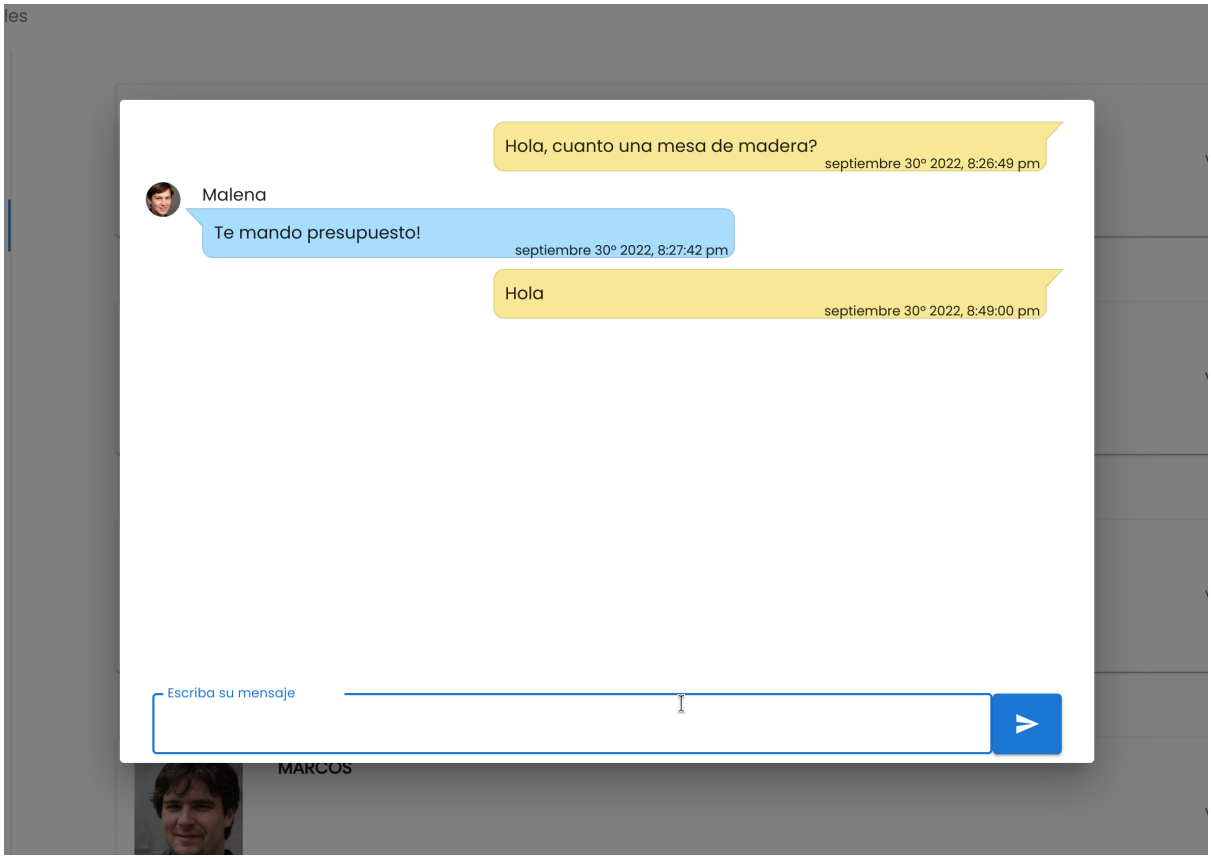


Figura 53: Conversación Vecino - Empleado

5.6.4 Mis reseñas

Otra de las opciones del menú de la izquierda nos permite ver las reseñas escritas por el mismo usuario, con la cantidad de likes o dislikes que la misma presenta.

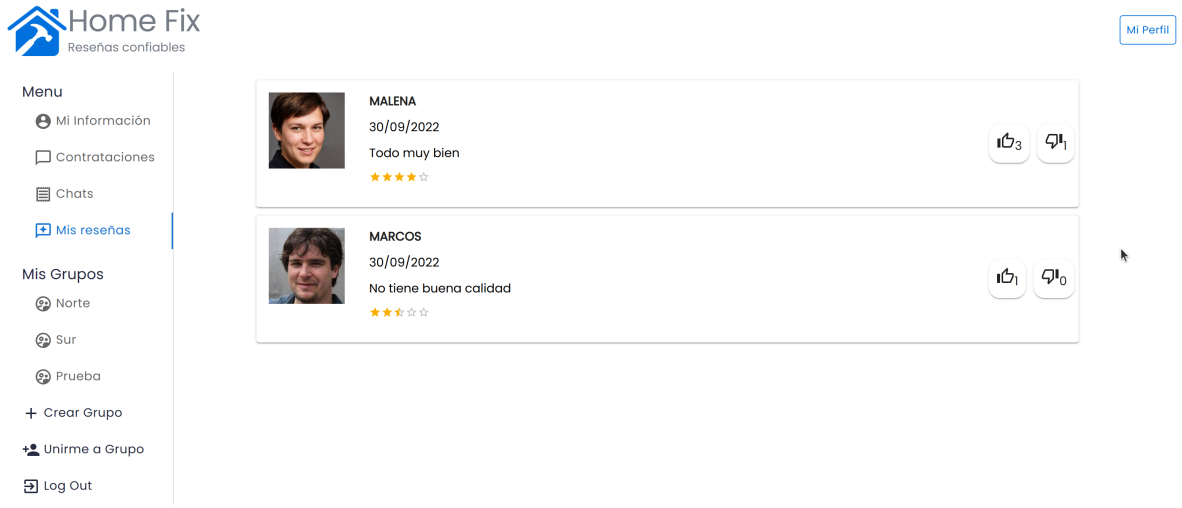


Figura 54: Reseñas vecino

5.6.5 Mis Grupos

Luego vemos la lista de los grupos de vecinos a los cuales pertenece el usuario, en este caso Norte, Sur y Prueba. Si se clickea sobre el barrio Norte, se observan los vecinos que pertenecen a ese grupo, los administradores del mismo en otra pestaña, y los mensajes que el administrador haya mandado al grupo. Estos mensajes cumplen la funcionalidad de ser parroquiales, es decir, avisos que se le quieran dar a todos los vecinos del grupo.

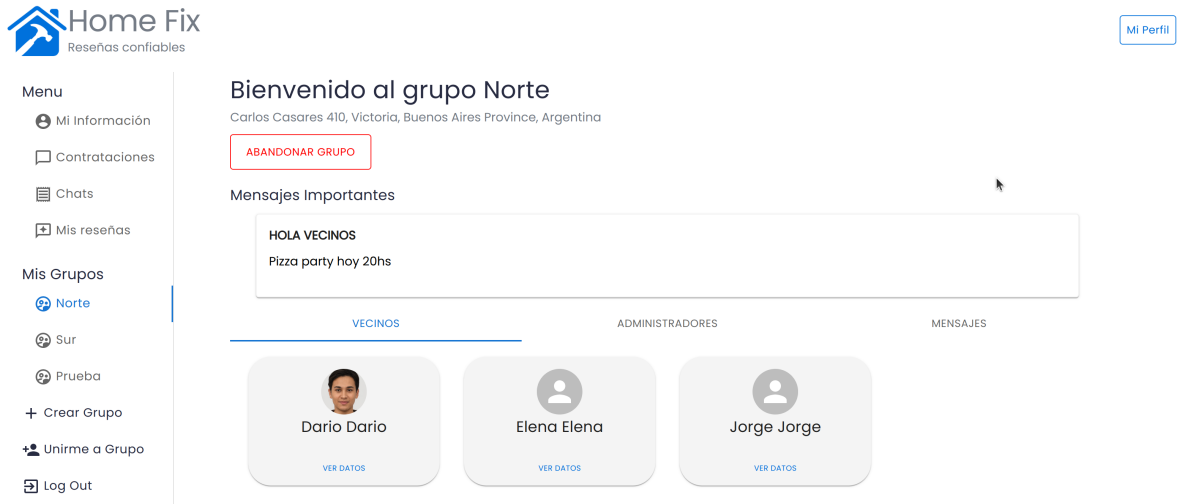


Figura 55: Vista principal grupo

Si se selecciona sobre VER DATOS, el vecino puede observar el celular y el email de

ese vecino, para contactarlo por cualquier problema, lo mismo con los administradores en su correspondiente pestaña. También podemos observar el botón de ABANDONAR GRUPO, si ese vecino así lo desea.

5.6.6 Crear grupo

Un vecino también puede crear un grupo, clickeando sobre la opción de crear grupo, y se le abrirá la página de la figura 56, en la cual el mismo solo debe completar el nombre del grupo y la dirección, ambas deben ser únicas. Ahora el vecino pasa a tener un perfil de administrador para ese grupo.

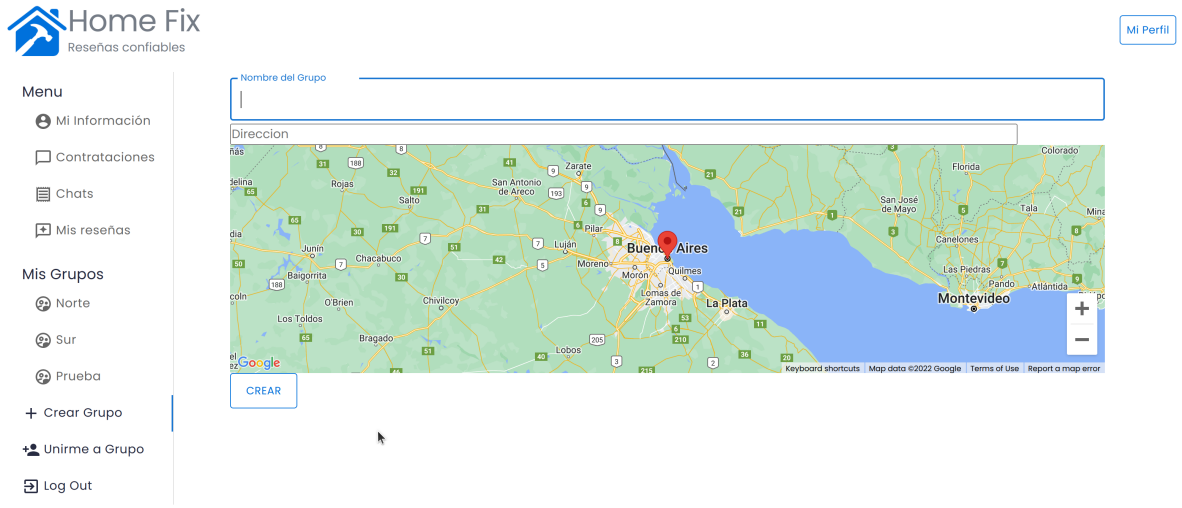


Figura 56: Creación de un grupo

Una vez completos los campos de nombre del grupo y dirección, y si no hubo errores (como podrían ser la repetición de un nombre del grupo o una dirección), se le mostrará un cartel de éxito al usuario, y podrá ver el nuevo grupo listado en la sección de Mis Grupos.

5.6.7 Unirse a un grupo

También el usuario puede unirse a un grupo, si tiene el nombre del mismo. Si desea hacerlo, selecciona la opción de Unirme a grupo en el menú de la izquierda, y verá la pantalla de la imagen 57, en donde al ingresar el nombre del grupo correctamente, se unirá al mismo.

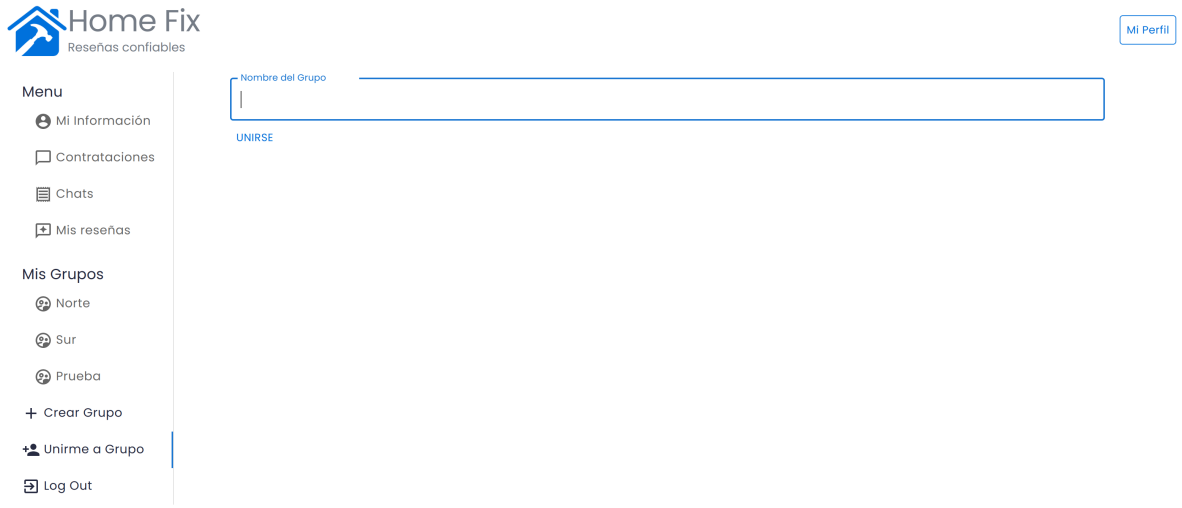


Figura 57: Unirse a un grupo

Por último, el usuario tiene un botón de Log Out, para cerrar la sesión.

5.7 Perfil Administrador

El perfil de administrador puede realizar las mismas acciones que un usuario vecino: puede ver su información, editarla, puede ver sus contrataciones, sus reseñas, puede tener conversaciones con trabajadores, puede crear y unirse a grupos.

5.7.1 Mis grupos

En esa sección el usuario puede ver sus grupos, y si es administrador de alguno tendrá la posibilidad de ver el botón de eliminar en los vecinos del grupo (por si hay algún vecino que entró al grupo por equivocación, o por alguna razón se desea eliminar a alguien del mismo) y también puede enviar los mensajes parroquiales. En la imagen 58 se muestra, a diferencia de la figura 55, el perfil de un administrador, con un botón "+" para agregar los mensajes parroquiales, y el botón rojo de ELIMINAR a cierto vecino.

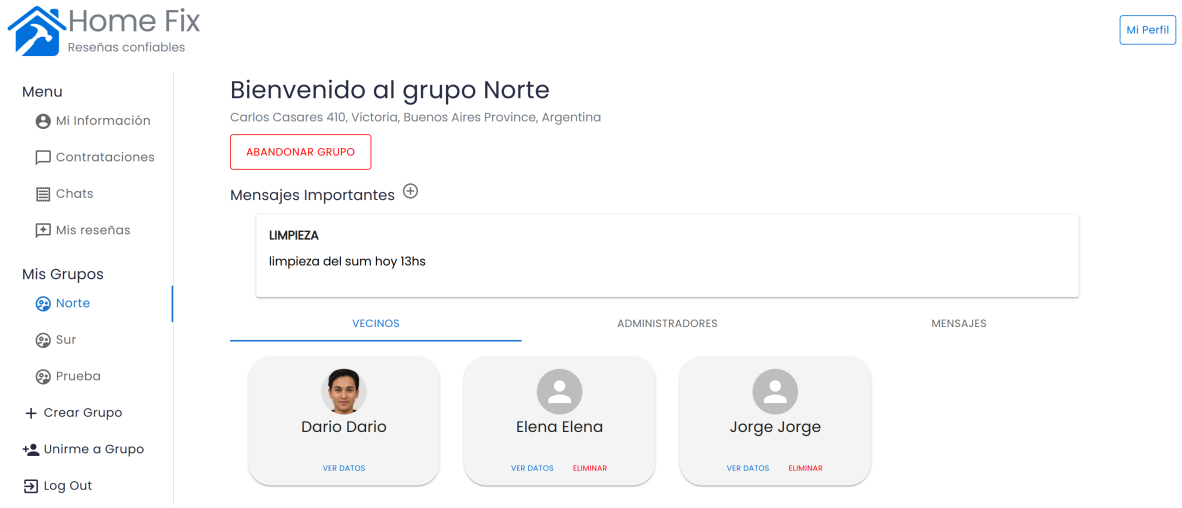


Figura 58: Vista principal grupo como administrador

5.8 SuperAdmin

Para ingresar a las funcionalidades de super administrador, se sigue el link **superAdmin/login**, en el cual se ve la pantalla de la figura 59.

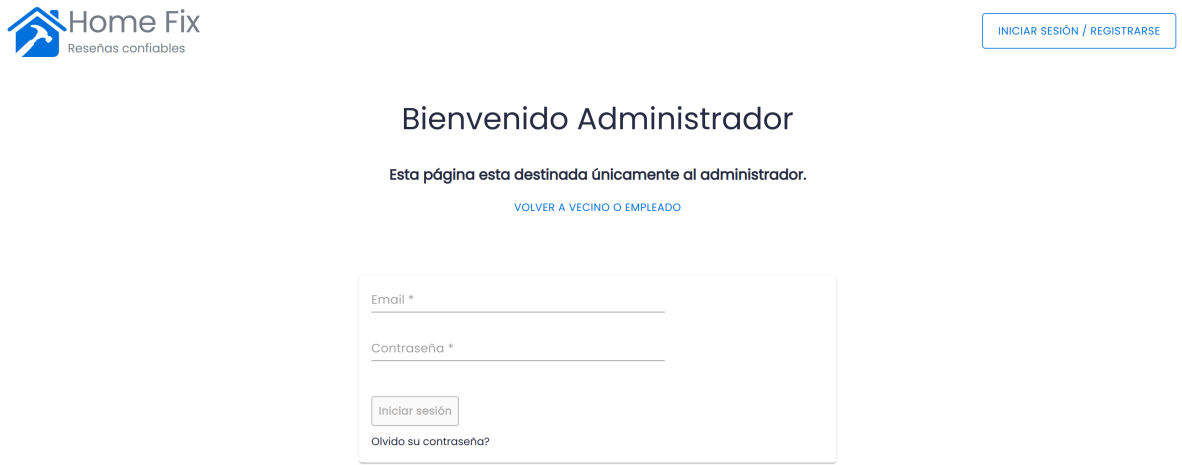


Figura 59: Iniciar sesión como administrador

Una vez ingresadas las credenciales correctamente, puede verse una tabla con 3 pestañas: trabajadores, reseñas reportadas y trabajadores reportados.
Si se clickea sobre trabajadores, se listan todos los de la aplicación, y puede verse información

sobre los mismos. La tabla permite ordenar alfabéticamente por nombre, apellido, email de los trabajadores, o numéricamente por rating. Este último orden es importante, ya que el super administrador tiene la posibilidad de reportar trabajadores, sobre todo los que tengan bajo rating, y es útil que los mismos sean ordenados con facilidad.

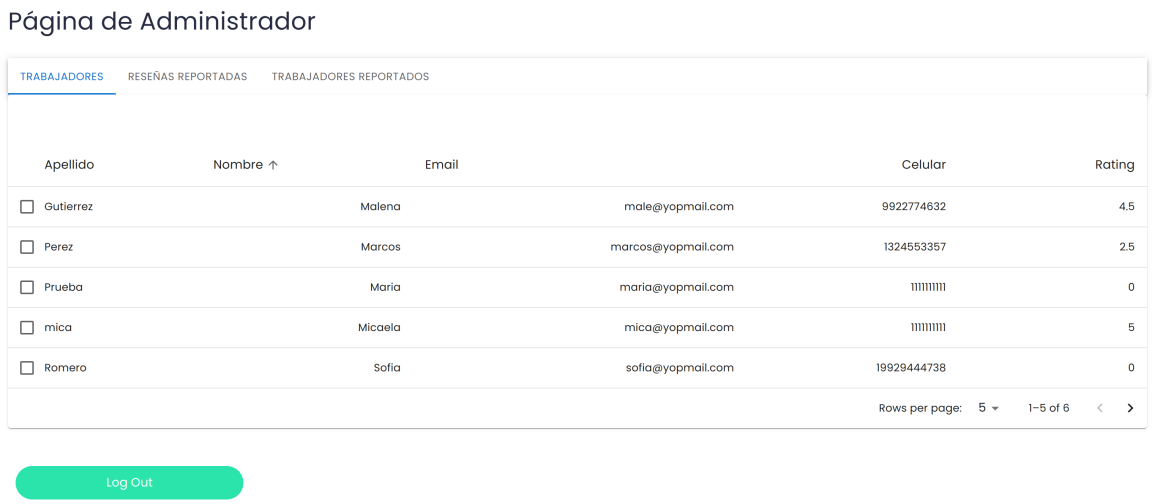


Figura 60: Tabla trabajadores

Si por alguna razón se desea reportar al trabajador, se debe clickear el cuadrado a la izquierda del apellido, y luego el botón de reportar, como se muestra en la figura 61.

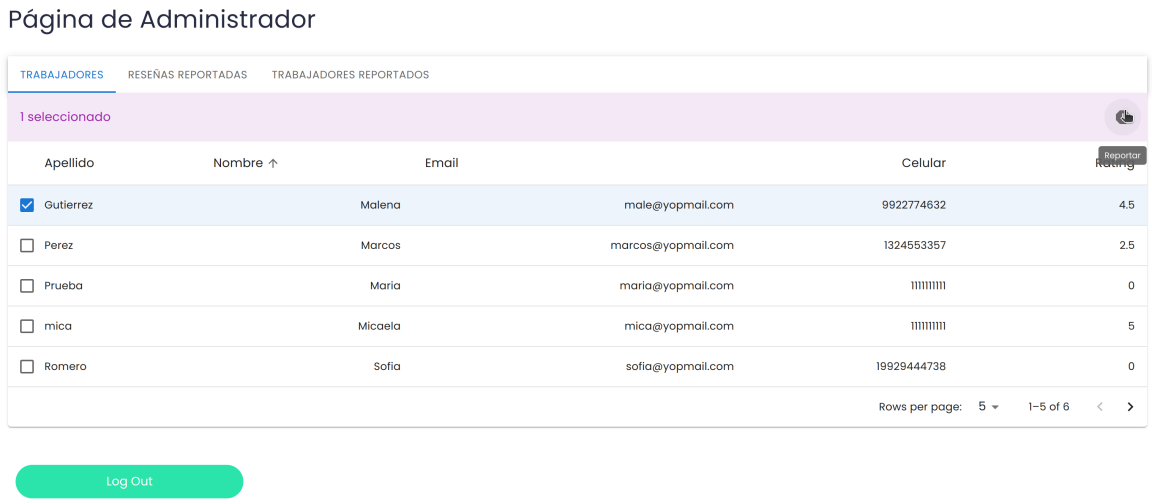


Figura 61: Reportar trabajador

Una vez reportado, el trabajador desaparece de la lista de trabajadores, y pasa a la

pestaña de trabajadores reportados. Si el mismo desea iniciar sesión, se le mostrará el cartel de la figura 39, y no aparecerá en las búsquedas.

Si se desea revertir el reporte de un trabajador, se dirige a la pestaña de trabajadores reportados, se clickea sobre el cuadrado a la izquierda del apellido y se selecciona la opción de *"Desreportar"*. Esto permite que el trabajador vuelva a la pestaña de trabajadores, el usuario va a poder iniciar sesión y va a aparecer nuevamente en la búsqueda de los vecinos.

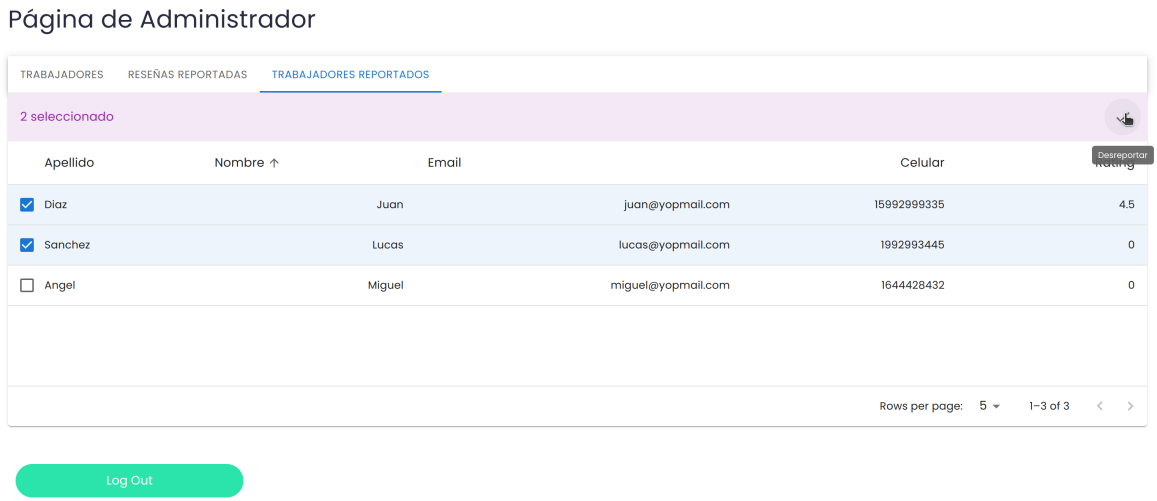


Figura 62: Desreportar trabajador

Si nos dirigimos a la pestaña de reseñas reportadas, podemos ver la lista de las mismas. Si no se desea desreportar la reseña, la misma queda ahí. Caso contrario en el que se quiera desreportarla, ya que no es ofensiva, se lo hace seleccionando la casilla de la izquierda, y luego presionando sobre el botón de *"Desreportar"*.

Si la reseña se encuentra reportada, no se mostrará en el perfil del trabajador, ni influye en el rating del mismo.

6 Testing

El objetivo de esta sección es expresar todas las medidas tomadas para las pruebas que se llevaron a cabo. El objetivo de las mismas es validar que la lógica de la aplicación funcione correctamente, que no hayan errores que perjudiquen otros aspectos de la misma y que ante cualquier tipo de cambio, no ocurran fallos en cadena.

Se hizo uso de dos tipos distintos de testing: Tests de integración y Usability tests.

6.1 Test de Integración

Este tipo de tests se basan en escribir pruebas en donde se trata de decidir y probar si las distintas funcionalidades de la aplicación funcionan correctamente. El gran foco de las mismas era testear completamente la integración del back end y analizar que todos los requerimientos de las mismas se cumplan.

Para esto, se utilizó la librería de Jest para crear el ambiente de pruebas y en conjunto con supertest se testearon los endpoint de manera E2E (End to end), que significa de principio a fin. Tests de integración más relevantes que se realizaron:

- Admin
  - Traer perfil de administrador
  - Crear perfil con rol de administrador
- Chat
  - Crear chat entre vecino y trabajador
  - Traer todos los chats entre dos usuarios
- Mensajes parroquiales
  - Crear un mensaje válido
  - Eliminar un mensaje
- Vecino
  - Traer todos los vecinos de un grupo de vecinos
  - Actualizar algún campo del vecino (teléfono celular)
- Contrataciones
  - Traer contrataciones
  - Crear contrataciones

  - Aceptar una contratación
  - Finalizar una contratación
  - Cancelar una contratación
- Usuarios
  - Validar cantidad de usuarios
  - Traer un empleado
  - Traer un vecino
  - Traer un administrador

Como precondición a estas pruebas se creó una base de datos de prueba para no alterar la base principal. Además, antes de cada test, se tira abajo la base de datos y se vuelven a cargar los datos para aislar las pruebas lo máximo posible.

6.2 Pruebas de usabilidad

Se realizaron pruebas con usuarios estudiantes de Ingeniería informática y además expertos en el área de diseño UX de manera recurrente ante cualquier gran cambio en el área de frontend, para buscar feedback de calidad de los cambios realizados, de alguien con conocimiento en materia de diseño. Siguiendo las buenas prácticas entendemos que podríamos haber extendido la prueba a vecinos de una comunidad y administradores de edificios, sin embargo en esta instancia no profundizamos en las mismas. Las pruebas realizadas nos brindaron feedback útil de personas con conocimientos, lo cual ayudó a iterar el producto. También se realizaron pruebas informales con amigos y miembros de nuestra familias, que cuentan poco conocimiento tecnológico. En estas pruebas ellos usaron la aplicación en diferentes iteraciones del producto y dieron su feedback para la mejora continua del mismo.

El objetivo de estas pruebas consisten en simular lo que haría un usuario convencional, fuera vecino, trabajador o administrador, y se les dio tareas acordes a las posibilidades dentro de la aplicación. Las mismas se listan a continuación.

Las pruebas se realizaron de manera *presencial*, es decir, mirando la pantalla y las acciones de los usuarios en tiempo real. A partir de las mismas se tomaban notas de cuales eran las tareas mas confusas, o que no lograron el objetivo provisto, y se les pedía feedback. Esta información era pasada a una tarjeta en la herramienta Trello [2], para que sea implementada por el equipo de desarrollo en su debido momento.

De haber algún inconveniente común entre los usuarios, se realizaban cambios del lado de UX y se les pedía a estos que vuelvan a realizar esa misma tarea. Muchos cambios

fueron realizados gracias a sugerencias de los entrevistados.

Algunas de las tareas realizadas fueron:

- Crear un usuario de tipo trabajador.
- Editar perfil para mostrar que solamente trabaja los fines de semana, por la mañana.
- Crear un usuario administrador, con nombre del grupo “Oeste” y cualquier dirección.
- Iniciar sesión como administrador y unirse al grupo Oeste.
- Crear un usuario vecino y unirse al grupo “Oeste”.
- Contactar a un trabajador y pedirle un presupuesto.
- Aceptar un presupuesto.
- Finalizar un presupuesto y escribir reseña.
- Como trabajador, enviar presupuesto a un vecino.

7 Metodología de Trabajo

Para el presente trabajo se siguió una metodología de trabajo ágil, en la cual se tuvieron sprints de dos semanas entre los integrantes del grupo (a excepción del mes de vacaciones). En estas reuniones le mostrábamos los avances del sprint anterior a la tutora, se revisaba el mismo y recibíamos feedback. Si habían funcionalidades que iterar, se agregaban en tarjetas al backlog de tareas en Trello, para luego ser implementadas. En estas reuniones también se planificaba lo que se iba a desarrollar en las siguientes dos semanas y que funcionalidades sería bueno priorizar para agregar a la plataforma.

Se definían historias de usuario al principio de cada sprint (o con 2 sprints de anticipación). Una vez completada cada historia de usuario, se iteraba sobre el producto para definir la siguiente y realizar los cambios necesarios para la integración de esta nueva funcionalidad.

Al comienzo del desarrollo de HomeFix se tenía un plan de proyecto definido, pero no se tenían definidas todas las funcionalidades que iba a tener, las mismas se iban definiendo a lo largo del tiempo. Cada implementación iba desarrollando el producto de manera incremental, así permitiendo tener un producto usable en cada entrega, en vez de al final de todo el proyecto. Por ejemplo, en una instancia inicial no se pensó en el chat interno, pero mientras surgían los casos de uso, se definió que este era necesario.

Hubo tres épicas antes de conformar el desarrollo de esta plataforma, estas épicas fueron presentadas en las pre-entregas de la materia. Las mismas consistían en:

1. La creación de los usuarios de tipo trabajador, la posibilidad de poder buscarlos y ver sus perfiles.

2. La segunda épica planteó la definición de grupos de vecinos. Para esto era necesario generar dos nuevos tipos de usuarios, vecino y administrador, en donde se agregó la posibilidad de dar reseñas a trabajadores que hayan brindado servicios dentro de un grupo de vecinos. También se agregaron las contrataciones y un chat interno para poder organizar todos los trabajos que se iban a realizar y asociarlos a una reseña. Además se incluyó el super administrador para poder moderar la plataforma.

3. La última épica consistía en implementar un motor de búsqueda de trabajadores en conjunto con la posibilidad de reaccionar a reseñas. Además, se sumó a la plataforma la posibilidad de que tanto los trabajadores como los grupos brinden su ubicación, para así poder asociarlos en base al barrio en el que se encuentran normalmente.

Se adjunta al informe un diagrama de Gantt, con el cronograma seguido por el equipo, el cual presenta funcionalidades y sus tiempos de diseño, desarrollo e investigación, teniendo en cuenta la disponibilidad de los integrantes. El objetivo de el mismo es

a nivel orientativo de las tareas realizadas, este diagrama fue parte de un desarrollo incremental y como resultado final pudimos tener un panorama más amplio de las tareas realizadas a lo largo del proyecto.

Para organizar las diferentes tareas a realizar, los problemas que se iban encontrando o las ideas que iban surgiendo, se utilizó la herramienta de trabajo Trello. En el mismo se tuvieron en cuenta los siguientes aspectos:

- Backlog: En esta columna iban aquellas tareas que se debían realizar.
- To Do: En esta columna iban aquellas tareas que debían desarrollarse en el Sprint actual.
- Doing: En esta columna iban aquellas tareas que estaban siendo desarrolladas.
- Blocked: Esta columna corresponde a aquellas tareas que están bloqueadas debido a otra tarea que no haya sido realizada.
- Done: En esta columna se iban ingresando aquellas tareas que finalizaban.
- Nice To Have: Acá se insertaban ideas a medida que el proyecto avanzaba que no entraban en el scope de este proyecto, pero que eran buenas ideas a implementar en un futuro.

8 Guía de instalación

En esta sección se explica cómo levantar HomeFix de manera local y en un entorno de producción. Dentro del repositorio se encuentra el *ReadMe*, con detalles paso por paso al respecto.
El repositorio puede encontrarse en https://bitbucket.org/itba/pf-2021-app_servicios/src/master/

Cabe resaltar que el proyecto presenta un archivo *.env*, en donde se personalizan las variables de entorno de trabajo individuales. Las apis keys son secretas, por eso para conseguirlas es necesario entrar tanto a Sendgrid como a la consola de Google para obtenerlas. Las variables son:

```
API_KEY_SENDGRID=
JWT_PRIVATE_KEY=
RESET_PASSWORD_KEY=
JWT_ACCOUNT_ACTIVATE=
REACT_APP_GOOGLE_MAPS_API_KEY=
BASE_URL=
SERVER_URL=
CLIENT_URL=
```

Figura 63: Variables de entorno

- API_KEY_SENDGRID: Clave privada provista por SendGrid para el envío de emails.
- JWT_PRIVATE_KEY: Salt utilizado para la encriptación de contraseñas.
- RESET_PASSWORD_KEY: Salt utilizado para la encriptación de contraseñas provistas por el cambio de contraseña.
- REACT_APP_GOOGLE_MAPS_API_KEY: Clave privada provista por Google Maps para el uso del visualizador de mapas.
- JWT_ACCOUNT_ACTIVATE: Salt utilizado para la activación de una cuenta a través del email.
- BASE_URL
- SERVER_URL
- CLIENT_URL

8.1 Local

Es necesario clonar el proyecto y tener instalado MongoDB puerto 27017 , NPM v 6.14.4, Node.js. v12.16.2 y React v18. Luego se instala el mismo con la ayuda de npm, luego se

lo corre y la aplicación estará disponible. Los pasos a seguir son

1. En la terminal correr el comando
   `git clone https://user@bitbucket.org/itba/pf-2021-app_servicios.git`
2. Esto deberia haber creado una carpeta llamada *pf-2021-app_servicios* en el directorio correspondiente
3. *cd pf-2021-app_servicios/homefix/*
4. *npm install*
5. crear el archivo *.env* dentro de la carpeta homefix con las variables necesarias descriptas anteriormente
6. *npm run dev* levanta servidor y cliente

El repositorio se encuentra disponible en https://bitbucket.org/itba/pf-2021-app_servicios/src/master/.

8.2 Producción

Para disponibilizar la aplicación en AWS son necesarios los siguientes aspectos:

- Una instancia EC2 con una imagen de Linux y al menos 8 GB de almacenamiento.
- En el repositorio se pueden observar tres archivos importantes para el despliegue de la aplicación: dos archivos de docker (uno para el frontend y otro el backend) y un tercero que es un *docker-compose* que se encarga de levantar las dos imágenes previamente descritas, una imagen para MongoDB y un tercer proceso que se encarga de subir información mockeada a la base de datos.
- Para subir el proyecto a la instancia de EC2 se hizo un zip del mismo y se transfirió por SSH a la instancia EC2, para esto fue necesario la creación de una clave ssh para incrementar la seguridad.
- Una vez dentro de la instancia EC2 es necesario descomprimir el proyecto, correr las imágenes de docker y ejecutar el docker-compose para orquestar todas las imágenes para que la aplicación funcione correctamente.

Recordar cambiar paths a los propios del usuario

1. ```
   zip -r homefix.zip homefix -x "**/node_modules/*" &&
   scp -r -i homefix/chron-test_1.pem
   /path/pf-2021-app_servicios/homefix.zip  ec2-user@44.206.239.186:~/
    && ssh -i homefix/chron-test_1.pem ec2-user@44.206.239.186
   ```
   ```

2. `rm -rf homefix && unzip homefix.zip && cd homefix/ &&`
   `sudo docker build --file=Dockerfile.front -t frontend . &&`
   `sudo docker build --file=Dockerfile.back -t backend . &&`
   `sudo docker-compose up`

8.3 Tests

Para correr los tests es necesario tener instalado Jest. Luego correr el comando `npm run test:watch`

9 Conclusiones y trabajo futuro

9.1 Aportes

En conclusión, HomeFix busca solucionar el problema del desconocimiento y la desconfianza a la hora de contratar trabajadores para el hogar, por medio de reseñas confiables de gente conocida. Se planteó como un proyecto interesante desde el comienzo, ya que simuló el desarrollo de una aplicación en la vida real. Desde las necesidades de los usuarios, las funcionalidades, el diseño, tanto del frontend como del backend, fueron realizados por los autores desde cero, pensando al proyecto como el trabajo de ingeniero en una empresa real. Desde el punto de vista del ámbito profesional, uno de los mayores desafíos fue el diseño de la interfaz del frontend, el cual se rediseñó 3 veces a lo largo del desarrollo, cada iteración mejorando la experiencia del usuario tanto visual como funcional. Otro desafío fue definir un buen algoritmo de búsqueda de trabajadores, ya que son necesarias diferentes queries, juntando información de diferentes tablas.

9.2 Trabajo futuro

El presente trabajo muestra el potencial que tendría HomeFix, por este motivo hay muchas features que se podrían añadir al desarrollo.

- Implementar la funcionalidad de eliminar usuario de la plataforma.
- Trabajo colaborativo con un equipo de marketing, para implementar el envío de distintos tipos de emails, como ser "Resumen de tu semana en HomeFix", o "Éstas son tus novedades de la semana".
- Persistir datos de usuarios antes de que la cuenta sea activada.
- Hacer que la instancia de AWS apunte al dominio homefix.com.ar (el cual ya se encuentra registrado).
- Agregar un mapa de la República Argentina, y no solo de la Provincia de Buenos Aires, al formulario de registro de los trabajadores.
- Poder enviar contenido multimedia en el chat integrado (esta feature ya es posible en el backend, pero no desde el frontend).
- Crear un buscador por nombre de vecinos en la vista de grupos de vecinos, en el caso que se desee ver la información de cierto vecino y haya muchos para filtrar.
- Poder eliminar reseñas hechas por uno mismo.
- Poder editar reseñas hechas por uno mismo.
- Poder añadir imágenes a las reseñas.

- Poder pinnear cierto mensaje parroquial en la vista de grupo de vecinos.
- Poder eliminar fotos de la galería de trabajos, desde el perfil de un trabajador.
- Hacer la vista del SuperAdministrador responsive.
- Guardar las imágenes (tanto de perfil, como de trabajos realizados por un trabajador) en un bucket S3.
- Que las reseñas puedan ser likeadas o deslikeadas por usuarios que hayan contratado al trabajador
- Poder recibir eventos de Sendgrid, ya sea errores al enviar un email, o recepción del mismo.
- Dar la opción, en los grupos de vecinos, de tener gente de confianza fija.
- Eliminar trabajadores que no se conectan en la aplicación hace tiempo (enviarles un email con previo aviso).
- Que las reseñas por parte de los vecinos sean obligatorias.

10 Anexos

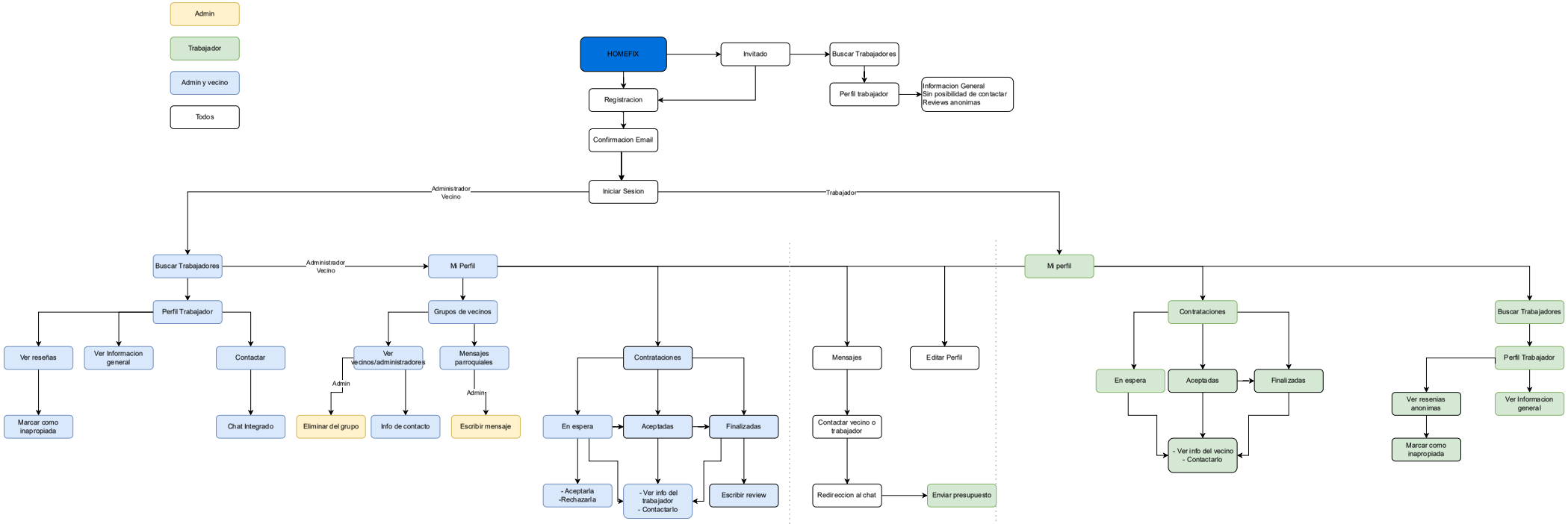
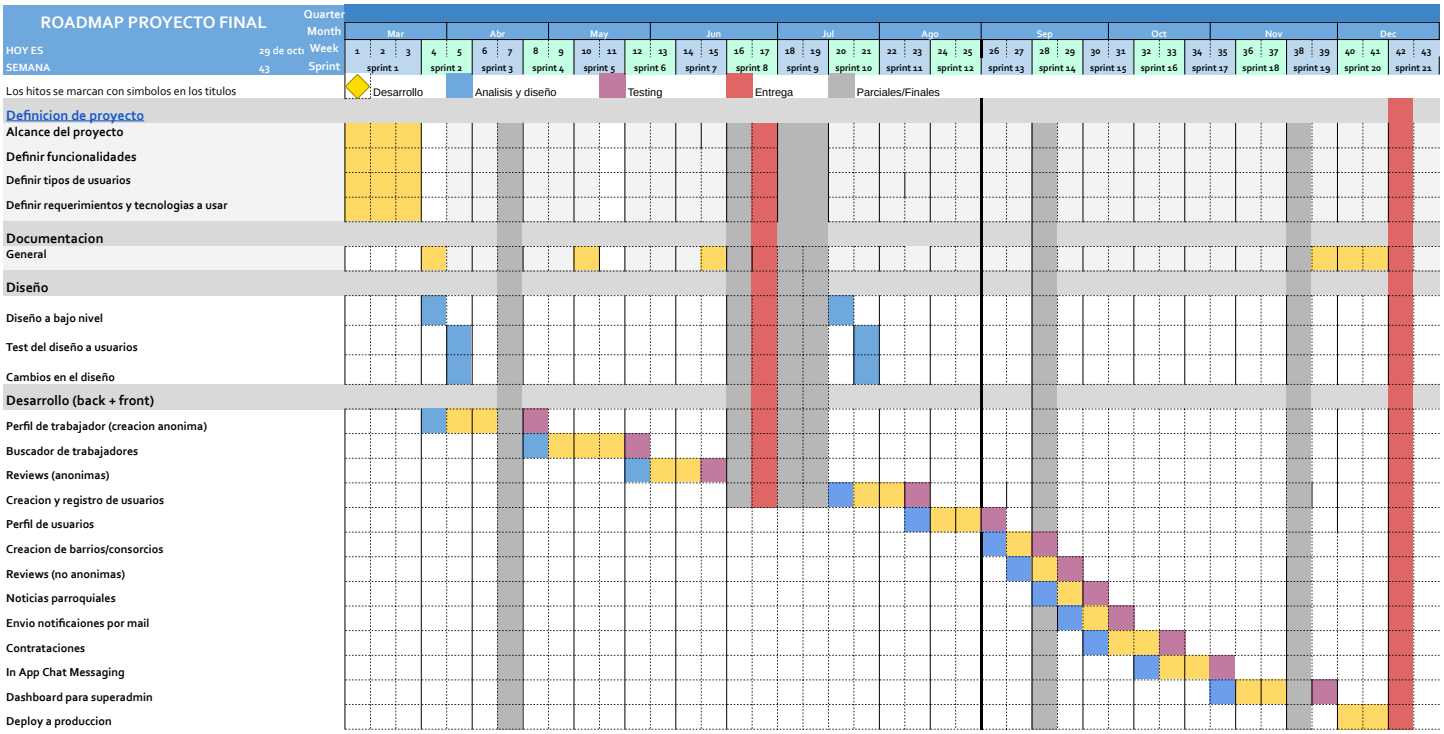


Figura 64: Diseño de alto nivel (Fuente: elaboración propia).



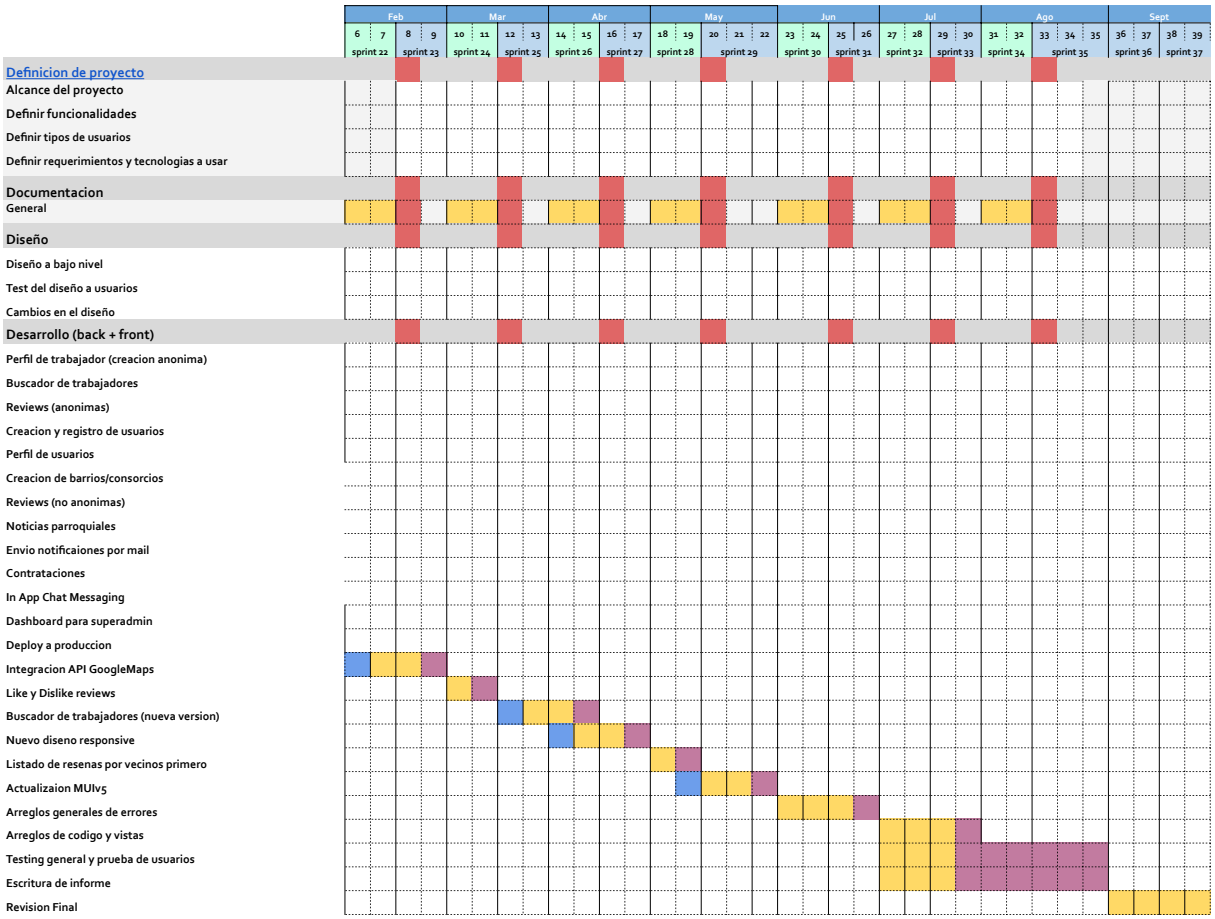


Figura Anexo: Diagrama de Gantt (Fuente: elaboración propia)

References

[1] *(JavaScript.com, s. f.)* URL: https://www.javascript.com/. (Recuperado: 30.10.2022).

[2] *(Manage Your Teams Projects From Anywhere | Trello, s. f.)* URL: https://trello.com/. (Recuperado: 30.10.2022).

[3] *AWS | Cloud Computing - Servicios de informática en la nube.* URL: https://aws.amazon.com/es/. (Recuperado: 30.10.2022).

[4] *AWS | Elastic compute cloud (EC2) de capacidad modificable en la nube.* URL: https://aws.amazon.com/es/ec2/. (Recuperado: 30.10.2022).

[5] *Davin, B. C. (2020, 26 febrero). Password Requirements - GDPR, ISO 27001/27002.* URL: https://davintechgroup.com/toolkit/password-requirements-gdpr-iso-27001-27002-pci-dss-nist-800-53/. (Recuperado: 18.11.2022).

[6] *Docker*. URL: https://www.docker.com/. (Recuperado: 30.10.2022).

[7] *Email Delivery, API, Marketing Service. SendGrid.* URL: https://sendgrid.com/. (Recuperado: 30.10.2022).

[8] *Empleos y Carreras profesionales en Argentina. Opcionempleo.com.ar*. URL: https://www.opcionempleo.com.ar/. (Recuperado: 30.10.2022).

[9] *experiencestack – Web App vs. Mobile App. (s. f.).* URL: https://experiencestack.co/web-app-vs-mobile-app-the-battle-of-the-apps-b0762a826b78. (Recuperado: 30.10.2022).

[10] *Google Maps Platform.* URL: https://developers.google.com/maps?hl=es-419. (Recuperado: 30.10.2022).

[11] *Home Solution*. URL: https://homesolution.net/ar/. (Recuperado: 30.10.2022).

[12] *JOI | The most powerful schema description language and data validator for JavaScript, s.f.* URL: https://joi.dev/. (Recuperado: 30.10.2022).

[13] *Limpia y Repara tu Hogar con Personas de Confianza. Zolvers*. URL: https://zolvers.com/. (Recuperado: 30.10.2022).

[14] *MongoDB: The Developer Data Platform.* URL: https://www.mongodb.com/. (Recuperado: 30.10.2022).

[15] *MUI: The React component library you always wanted.* URL: https://mui.com/. (Recuperado: 30.10.2022).

[16] *Peerbits. The benefits of ReactJS and reasons to choose it for your project.* URL: https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html. (Recuperado: 30.10.2022).

[17] *React – Una biblioteca de JavaScript para construir interfaces de usuario. (s. f.).* URL: https://es.reactjs.org/. (Recuperado: 30.10.2022).

[18] *Socket.IO – Bidirectional and low-latency communication for every platform. (s. f.).* URL: https://socket.io/. (Recuperado: 18.11.2022).

[19] *The State of JS 2021: Front-end Frameworks*. URL: https://2021.stateofjs.com/en-US/libraries/front-end-frameworks. (Recuperado: 30.10.2022).

[20] *Timbrit*. URL: https://www.timbrit.com.ar/. (Recuperado: 30.10.2022).

[21] *Toptal – Why would i use Nodejs. (s. f.)*. URL: https://www.toptal.com/javascript/why-the-hell-would-i-use-node-js#:~:text=with%20high%20throughput.-,Node.,understanding%20this%20is%20absolutely%20essential. (Recuperado: 30.10.2022).