CHOREMASTER, UNA APLICACIÓN WEB PARA LA GESTIÓN DE TAREAS DEL HOGAR

CHOREMASTER, A WEB APPLICATION FOR HOUSEHOLD TASK MANAGEMENT

TRABAJO FIN DE GRADO
CURSO 2022-2023

AUTOR
JUAN IGNACIO FERNÁNDEZ JOVER

DIRECTOR
RAMON GONZALEZ DEL CAMPO RODRIGUEZ BARBERO

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

CHOREMASTER, UNA APLICACIÓN WEB PARA LA GESTIÓN DE TAREAS DEL HOGAR

CHOREMASTER, A WEB APPLICATION FOR HOUSEHOLD TASK MANAGEMENT

TRABAJO DE FIN DE GRADO EN INGENIERÍA INFORMÁTICA

AUTOR
JUAN IGNACIO FERNÁNDEZ JOVER

DIRECTOR
RAMON GONZALEZ DEL CAMPO RODRIGUEZ BARBERO

CONVOCATORIA: SEPTIEMBRE 2024

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

DÍA DE MES DE AÑO

DEDICATORIA

Principalmente a mis padres que siempre me han apoyado y me han aguantado.

Y también a todas las personas que han posibilitado que este proyecto saliera adelante ayudando de una manera u otra.

AGRADECIMIENTOS

En primer lugar, a Ramón González del Campo, tutor de este tfg, por guiarme y ayudarme en todo lo que ha estado en su mano. Además, un especial agradecimiento a Rocío Fernández y Clara López por aguantarme y ayudarme siempre que se lo he pedido, sin ellas la aplicación no habría sido la misma.

RESUMEN

ChoreMaster, una aplicación web para la gestión de tareas del hogar

Este proyecto refleja el desarrollo de una aplicación web que trata de facilitar un problema muy frecuente en la vida de los jóvenes que se independizan: la convivencia en pisos compartidos. Para ello, esta aplicación permitirá a sus usuarios organizar las tareas de la casa y los gastos comunes, fuentes de numerosas tensiones, de manera sencilla y eficiente.

Para su realización, se escogieron tecnologías modernas y ampliamente utilizadas como React y Node.js, mejorando así la eficiencia y flexibilidad durante el desarrollo. Previo a este, se realizaron entrevistas para determinar la funcionalidad necesaria para el proyecto y así empezar con el diseño de la aplicación.

En el diseño surgieron dos roles dentro de cada grupo: usuario y administrador. Ambos roles requieren iniciar sesión por parte del usuario, el primero permite a los usuarios sin grupo gestionar sus tareas personales y una vez se une a un grupo gestionar las tareas de la casa, así como los gastos y la lista de la compra. Los administradores pertenecen necesariamente a un grupo y pueden hacer lo mismo que los usuario con grupo y además permite gestionar los datos y miembros del grupo.

Finalmente, tras numerosas dificultades, se llevó el diseño a la realidad dando como resultado una aplicación web que resolvía el problema planteado inicialmente. A pesar de ello, la aplicación aún tiene grandes márgenes de mejora pudiendo implementar nuevas funcionalidades que facilitarían la vida al usuario y mejorarían su experiencia con la aplicación.

Palabras clave

Aplicación web, MySQL, organización, gasto

ABSTRACT

CHOREMASTER, A WEB APPLICATION FOR HOUSEHOLD TASK MANAGEMENT

This project reflects the development of a web application which tries to simplify a common issue faced by young people who move out on their own: living in shared apartments. To tackle this, the application allows users to organize household chores and shared expenses, common sources of tension, in a simple and efficient manner.

For its development, modern and widely used technologies like React and Node.js were chosen, enhancing efficiency and flexibility during the development process. Before starting, interviews were conducted to determine the necessary functionality for the project, which then informed the design of the application.

During the design phase, two roles emerged within each group: user and administrator. Both roles require the user to log in. The user role allows individuals without a group to manage their personal tasks and, once they join a group, to manage household tasks, expenses, and the shopping list. Administrators must belong to a group and can do everything that group users can do, plus they can manage the group's data and members.

Finally, after overcoming numerous challenges, the design was brought to life, resulting in a web application that successfully addressed the initial problem. However, the application still has significant room for improvement, with the potential to implement new features that would further facilitate users' lives and enhance the user experience.

Keywords

Web application, MySQL, organization, expense

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

Capítulo 1 - Introducción

Hoy en día es una realidad que la mayor parte de los jóvenes que se independizan se ven obligados a vivir en pisos compartidos, en algunos casos con amigos, en otros con completos desconocidos. En esa situación la convivencia es crítica, y donde una mala gestión de la casa puede ser una lacra para esta misma.

Sumado a este primer problema, la gente joven que se independiza tiende a tener una agenda muy apretada y poco tiempo libre. Por ello, poder organizarse de manera que se pueda incluir también las tareas de la casa en su rutina es algo fundamental para ahorrar tiempo y, a su vez, mejorar la convivencia en la casa

En este documento desarrollaré el resultado de estas ideas, una aplicación web que permita organizar las tareas de la casa y de uno mismo en una sola aplicación. Empezando desde el estudio de las aplicaciones actuales y cómo esta aplicación se eleva frente a ellas; siguiendo por la elección de tecnologías y todas las etapas del diseño y desarrollo; para terminar con una conclusión y los pasos a seguir en la aplicación

1.1 Motivación

¿Quién no ha escuchado sobre un amigo, familiar o conocido que se ha tenido que cambiar de piso por sus compañeros? Estas situaciones que son relativamente frecuentes en el mundo actual son consecuencia de una mala convivencia y en la mayoría de los casos se pueden evitar. En todas las casas suele haber dos puntos críticos: las tareas de la casa y los gastos compartidos.

Por ello, organizando bien esos dos aspectos de la convivencia esta se simplifica mucho, evitando así diversas discusiones.

1.2 Objetivos

El objetivo principal de este proyecto es el desarrollo de una aplicación web que permita a los miembros de una casa organizar las tareas de esta y organizarse uno mismo con las tareas personales y de la casa, así como, compartir gastos de una manera sencilla.

Durante el desarrollo del proyecto se fueron estableciendo objetivos a lo largo de toda la ruta a seguir:

- Análisis de otras aplicaciones con funcionalidad similar a la propuesta para ver y adoptar ideas, y dar valor añadido frente a sus carencias.
- Pequeñas entrevistas a potenciales usuarios y diseño de la funcionalidad.
- Explorar las posibilidades que ofrece Figma y realizar un prototipo interactivo.
- Desarrollar e implementar el diseño ya realizado, además crear y gestionar una base de datos para la aplicación.
- Redacción de la memoria del TFG explicando todo el estudio, diseño y desarrollo de la aplicación.

1.3 Plan de trabajo

Para poder desarrollar el proyecto dentro de los márgenes de tiempo establecidos, se propuso un cronograma al inicio con una serie de fases. Algunas de las cuales se superponían entre ellas: investigación, diseño, desarrollo, documentación.

Investigación:

Fecha inicio: 10 noviembre Fecha de entrega: 10 febrero

Objetivos:

- Búsqueda y análisis de aplicaciones similares enfatizando en la funcionalidad deseada para la aplicación.
- Análisis y decisión de herramientas y tecnologías usadas a lo largo del proyecto.

- Preparación y realización de entrevistas para el diseño de la funcionalidad.

Diseño:

Fecha inicio: 18 enero Fecha de entrega: 10 junio

Objetivos:

- Creación de casos de uso que muestren la especificación de la funcionalidad.
- Creación de mockups y vistas de la aplicación.
- Creación de un prototipo interactivo de la aplicación.

Desarrollo:

Fecha inicio: 10 abril Fecha de entrega: 15 agosto

Objetivos:

- Creación de la base de datos.
- Desarrollo de las vistas conforme a los diseños y casos de uso.
- Realizar diferentes pruebas a la aplicación y corregir los posibles bugs existentes.

Documentación:

Fecha inicio: 10 noviembre Fecha de entrega: 1 septiembre

Objetivos:

- Tomar apuntes de cada paso realizado y cualquier información que se pueda incluir en la memoria.
- Redacción de una memoria técnica que detalle todos los aspectos del desarrollo del proyecto.

Introduction

Today, it is a reality that most young people who move out on their own are forced to live in shared apartments, sometimes with friends, other times with complete strangers. In such situations, cohabitation is critical, and poor household management can be a burden on everyone involved.

In addition to this initial problem, young people who move out on their own tend to have very tight schedules and little free time. Therefore, being able to organize themselves in a way that includes household chores in their routine is essential to saving time and, at the same time, improving cohabitation in the home.

In this document, I will develop the result of these ideas: a web application that allows organizing household tasks and personal tasks in a single application. It will start with a study of current applications and how this application stands out among them; followed by the choice of technologies and all the stages of design and development; and ending with a conclusion and the steps to follow for the application.

Motivation

Who hasn't heard of a friend, relative, or acquaintance who had to move out because of their roommates? These situations, which are relatively common in today's world, are the result of poor cohabitation and can often be avoided. In every household, there are usually two critical points: household chores and shared expenses.

Therefore, by properly organizing these two aspects, cohabitation is greatly simplified, thus avoiding various conflicts.

Goals

The main objective of this project is to develop a web application that allows household members to organize household chores and their personal tasks, as well as share expenses in a simple way.

Throughout the project's development, objectives were set along the way:

- Analyze other applications with similar functionality to the proposed one to see and adopt ideas and add value by addressing their shortcomings.
- Conduct small interviews with potential users and design the functionality.
- Explore the possibilities offered by Figma and create an interactive prototype.
- Develop and implement the already created design, as well as create and manage a database for the application.
- Write the final project report, explaining the entire study, design, and development of the application.

Work plan

To develop the project within the established time frame, a schedule was proposed at the beginning, with different phases, some of which overlapped: research, design, development, and documentation.

Research:

Start date: November 10 Delivery date: February 10

Objectives:

- Search and analyze similar applications, emphasizing the desired functionality for the application.
- Analyze and decide on the tools and technologies used throughout the project.
- Prepare and conduct interviews for design the functionality.

Design:

Start date: January 18 Delivery date: June 10

Objectives:

- Create use cases that specify the functionality.

- Create mockups and views of the application.
- Create an interactive prototype of the application.

Development:

Start date: April 10 Delivery date: August 15

Objectives:

- Create the database.
- Develop the proposed use cases.
- Develop the views according to the already created designs.
- Conduct various tests on the application and fix any existing bugs.

Documentation:

Start date: November 10 Delivery date: September 1

Objectives:

- Take notes on each step taken and any information that can be included in the final report.
- Write a technical report detailing all aspects of the project's development.

Capítulo 2 - Estado de la cuestión

En este capítulo realizaré un análisis de las aplicaciones ya creadas que tratan de resolver el problema ya explicado en el capítulo anterior. En el análisis se hará un breve resumen de la funcionalidad de la aplicación y nos centraremos en los virtudes y carencias de cada una. Para terminar, explicaré qué aporta el proyecto a este mercado que ya está claramente explotado.

Actualmente hay un gran número de aplicaciones que sirven para organizar las tareas de la casa, los gastos o los quehaceres personales. Debido a ello, intentaré seleccionar algunas representativas por su relevancia en la sociedad o que aportaron ideas relevantes a este proyecto.

2.1 Aplicaciones similares

2.1.1 Any.do

Any.do es una aplicación bastante conocida que permite organizar fácilmente tareas de cualquier índole en un grupo y organizarlas junto a tus propias tareas. Además, permite tener diferentes listas que pueden hacer de lista de la compra como ellos mismo te sugieren.

Puntos fuertes: Las tareas son altamente personalizables y permite tener diferentes grupos de trabajo facilitando la organización individual.

Puntos débiles: La aplicación no permite poner una tarea recurrente para repetir semanalmente.

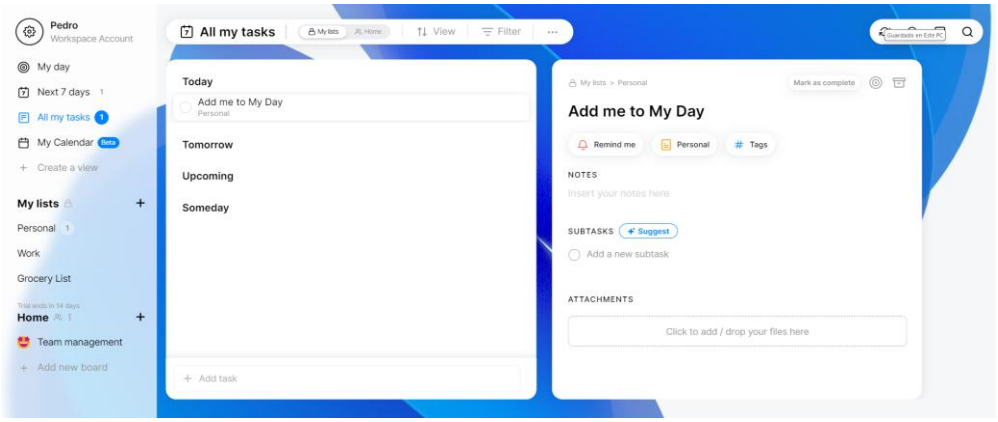


Ilustración 2-1 Pantalla de Any.do

2.1.2 Nipto: tareas del hogar

Nipto es una aplicación móvil pensada para la gestión de tareas exclusivas del hogar. La aplicación plantea las tareas de la casa como un juego en el que vas ganado puntos según las haces, intentando motivar a los usuarios mediante la competición.

Puntos fuertes: El modelo que plantea es muy innovador y sugerente.

Puntos débiles: La aplicación luce bastante infantil y no permite hacerte un horario completo, ya que solo incluye las tareas de la casa.

Ilustración 2-2 Pantalla de Nipto

2.1.3 Tricount

Tricount es una aplicación móvil para gestionar los gastos de un grupo de personas, permitiendo repartir los gastos entre todos e indica las deudas entre los diferentes miembros. La aplicación es muy completa y ha servido de inspiración en la sección de gastos de la aplicación.

Puntos fuertes: Permite agregar y gestionar un gasto de manera sencilla e intuitiva. Actualiza las deudas para que haya los menos pagos posibles.

Puntos débiles: La aplicación es muy limitada ya que solo gestiona gastos

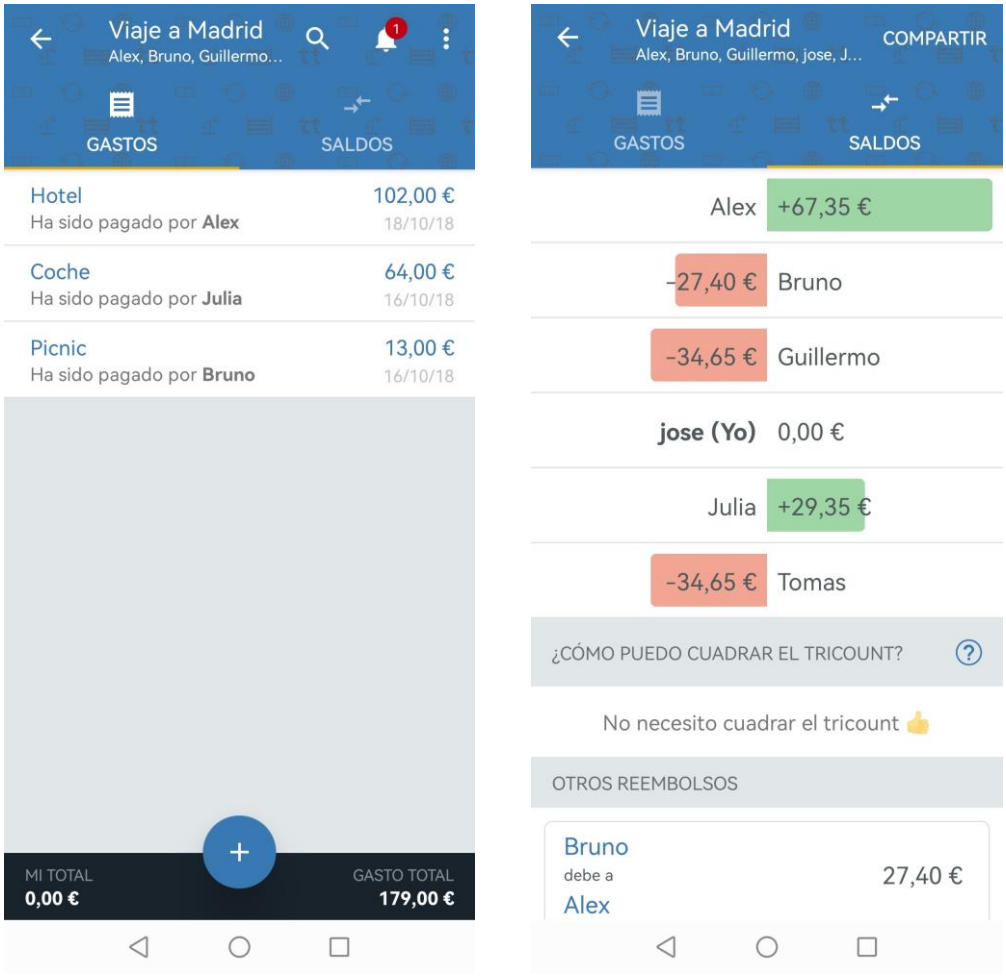


Ilustración 2-3 Pantallas de tricount

2.1.4 Home Tasker

Es una aplicación móvil muy simple que permite crear y distribuir las tareas de la casa entre los miembros. Se diferencia por la división de las tareas de la casa en función de la habitación a la que pertenece.

Puntos fuertes: Tiene una interfaz muy simple que hace que su uso sea muy intuitivo.

Puntos débiles: No permite distribuir las tareas en un horario para organizarte.

Ilustración 2-4 Pantalla de Home Tasker

2.1.5 Family Wall

Family Wall es una de las mejores aplicaciones de este campo. Es muy completa ya que incluye dentro de sus funcionalidades, aparte de gestionar las tareas de la casa, hacer un plan de comidas, la lista de la compra, subir archivos importantes o tener un chat familiar. Esta aplicación ha servido de inspiración en múltiples etapas del desarrollo, especialmente a la hora de diseñar la funcionalidad de la aplicación.

Puntos fuertes: abarca todos los ámbitos que una familia pueda necesitar. Es de las pocas que incluye, aunque de manera limitada, una gestión de gastos familiar con un presupuesto.

Puntos débiles: esta únicamente enfocado a la vida familiar y pierde utilidad en otro tipo de situaciones.

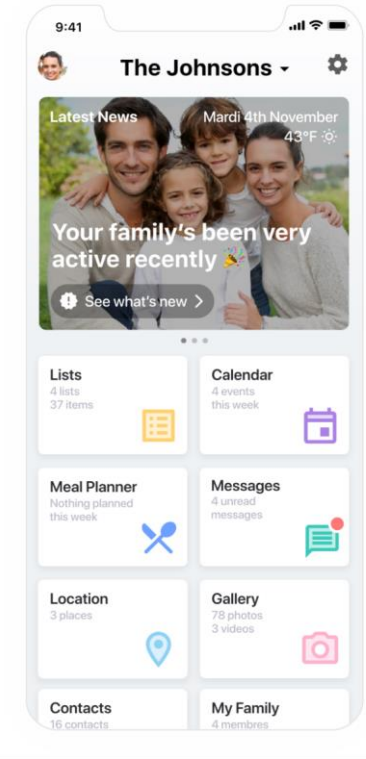


Ilustración 2-5 Pantalla de Family Wall

2.2 Diferenciación de mi aplicación

Como se ha visto en el apartado anterior hay una gran variedad de aplicaciones que pueden cumplir el rol de la que se propone en este proyecto. Una aplicación capaz de gestionar las tareas y gastos de una casa, y a la vez permitir que cada usuario pueda organizar su semana teniendo en cuenta dichas tareas.

El primer punto donde esta aplicación destaca es su usuario objetivo, personas que vivan en un piso compartido. Tal y como se ha apreciado anteriormente, las

aplicaciones se centran más en las familias dejando de lado este tipo de hogares, en los que es igual de necesario si no más este tipo de aplicaciones para organizarse.

En segundo lugar, ninguna aplicación consigue juntar la organización de tareas y gastos. La única que se acerca es Family Wall, pero al hacerlo desde un punto de vista familiar pierde toda su utilidad fuera de este ámbito.

En conclusión, para diferenciarse, mi aplicación resuelve los dos problemas planteados. De esta manera permite gestionar tareas y gastos en una única aplicación. A su vez, se acerca a un público cada vez más numeroso que pasa desapercibido y no se le presta la atención necesaria.

Capítulo 3 - Elección de tecnologías

Durante el desarrollo de este proyecto se han usado muchas tecnologías y herramientas tanto para el diseño, la organización y el desarrollo. A lo largo de este capítulo, las iré detallando minuciosamente. Además, explicaré que aporta el uso de dichas tecnologías, así como su elección frente a otras posibilidades.

Para una mayor claridad voy a dividirlas en tres grupos, frontend, para aquellas que han ayudado en la creación de la interfaz de usuario. Backend, para las que facilitan la lógica detrás de la interfaz y gestionan la comunicación con la base de datos Generales, para aquellas que aportan en otros ámbitos como el diseño o la organización del proyecto.

3.1 Frontend

El frontend es la parte del desarrollo dedicada a crear la parte visible de la aplicación, aquella con la que el usuario interactuará durante el uso de la aplicación. Para esta parte del desarrollo elegí usar React, una biblioteca muy popular en el desarrollo de aplicaciones web.

3.1.1 React

React es una biblioteca muy popular debido a las múltiples facilidades que aporta al desarrollo de interfaces, gracias a su gran cantidad de librerías y herramientas. Además, tiene una gran comunidad que la apoya y da soporte, haciéndola muy atractiva para desarrollar este proyecto. Algunos de sus puntos más fuertes son:

- **Componentes reutilizables**: React permite crear componentes que pueden ser utilizados en distintas partes de la aplicación mejorando la consistencia de esta.
- **Virtual DOM**: el rendimiento de React es uno de sus puntos a destacar, este lo logra gracias a que utiliza un virtual DOM para no tener que renderizar la página completa constantemente.

- **React Hooks**: estos introducen una manera muy sencilla de manejar tanto el estado de la página como las interacciones en ella.

En conclusión, React es una librería que ofrece mucha versatilidad durante el desarrollo y rendimiento a la aplicación final. Pero el motivo final por el cual se escogió React frente a otras opciones como Angular, es por su flexibilidad para poder integrar diferentes herramientas y librerías.

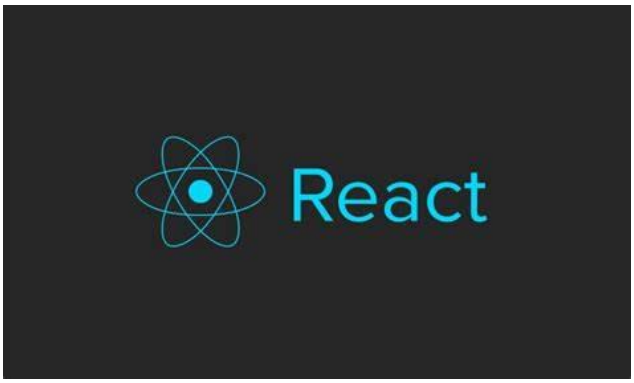


Ilustración 3-1 Logo de React

3.1.2 Formik

Formik es una biblioteca de JavaScript que se encarga de la gestión de formularios en aplicaciones React. Esta ofrece una serie de ventajas muy grandes en el manejo de formularios, validación y en el manejo del estado de estos.

- **Gestión del estado**: Esta biblioteca se encarga de gestionar automáticamente el estado de sus formularios facilitando su uso.
- **Validación personalizable**: Juntando Formik con Yup, una biblioteca de validación de esquemas, se permite validar los diferentes campos de un formulario de manera muy variada y sencilla.
- **Manejo de errores**: Formik facilita en gran medida la gestión y visualización de errores en formularios. Estos se pueden ver de manera clara junto al campo correspondiente, mejorando la experiencia de usuario.

- **Componentes**: La biblioteca proporciona una serie de componentes de alto nivel que facilitan en gran medida la creación de formularios, manteniendo un código más limpio y legible.

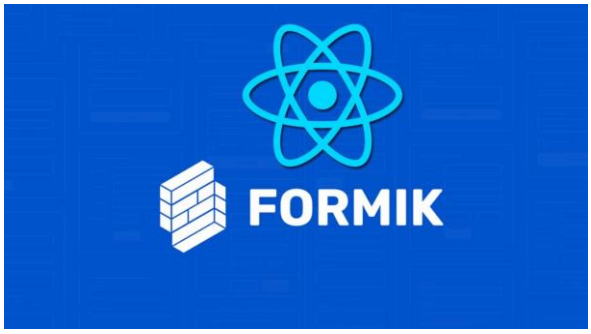


Ilustración 3-2 Logo de Formik

3.1.3 Chakra UI

Chakra UI es una biblioteca de componentes para React que ayuda en la creación de interfaces de usuario. Esta se ha usado en partes específicas de la aplicación donde su uso ha acelerado claramente el desarrollo, estas son algunas de sus ventajas:

- **Facilidad de uso y personalización**: Es una biblioteca muy intuitiva con una API que permite a los desarrolladores personalizar los componentes mediante props y estilos de manera muy sencilla.
- **Componentes responsivos**: los componentes que proporciona Chakra están pensados para adaptarse a diferentes tamaños de pantalla sin tener que escribir código adicional.
- **Accesibilidad**: La biblioteca da una gran importancia a la accesibilidad, cumpliendo así con las pautas de accesibilidad web.

Ilustración 3-3 Logo de chakra

3.2 Backend

El backend es la parte del desarrollo enfocada a la parte no visible de la aplicación, aquella que se encarga de la lógica y la comunicación con la base de datos. Para esta parte del desarrollo decidí utilizar Node.js como entorno de ejecución y como base de datos, una MySQL alojada en un servidor local gestionado por Xampp.

3.2.1 Node.js

Node.js es un entorno de ejecución que permite ejecutar código JavaScript en el lado del servidor. Este destaca por su velocidad y eficiencia que permite manejar múltiples operaciones simultáneamente. Estas son algunos de sus puntos más fuertes:

- **Unificación del lenguaje**: Al permitir usar JavaScript en el backend, se puede desarrollar una aplicación completamente en el mismo lenguaje facilitando la integración y el intercambio de datos.
- **Rendimiento**: Debido a la arquitectura no bloqueante y basada en eventos, Node.js es capaz de manejar múltiples solicitudes simultáneamente con gran eficiencia.
- **Variedad de módulos**: Node.js usa uno de los gestores de paquetes más grande del mundo, Node Package Manager (npm). Esto le permite tener acceso a una vasta cantidad de módulos y herramientas que facilita en gran medida el desarrollo.

- **Escalabilidad**: La capacidad que tiene Node.js de manejar peticiones concurrentes la hace ideal para aplicaciones que pueden tener una alta demanda o un gran escalado.

Ilustración 3-4 Logo de Node

3.2.2 Xampp

Xampp es un software libre que facilita la instalación y configuración de un servidor web local, y además ofrece un sistema de gestión de bases de datos MySQL. Esta herramienta al ser local y muy manipulable es ideal para este proyecto dado su carácter individual, Además Xampp destaca por:

- **Interfaz gráfica**: La herramienta permite gestionar una base de datos de manera muy intuitiva, permitiendo que el desarrollador vea o modifique cualquier elemento de la base de datos de manera muy sencilla.
- **Herramientas útiles**: Aparte del gestor de la base de datos, también se incluye un servidor de correo ideal para realizar pruebas.
- **Portabilidad**: A pesar de ser un servidor local, Xampp ofrece una gran portabilidad, permitiendo tenerlo en un USB. Esta característica fue fundamental en un punto del desarrollo debido a problemas técnicos.

Ilustración 3-5 Logo de Xampp

3.2.3 Sequelize

Sequelize es un Object-Relational Mapping para Node.js que permite comunicarse con una base de datos SQL. Algunos de sus puntos fuertes son:

- **Modelo abstracto**: Esta librería permite trabajar con bases de datos SQL, en este caso MySQL, sin tener la necesidad de escribir consultas. En su lugar, ofrece una serie de métodos en JavaScript que cumple dicha función.
- **Modelos de datos estructurados**: Sequelize facilita la creación de modelos de datos de manera clara y muy visible. Además, permite establecer relaciones entre estos modelos muy fácilmente.
- **Compatibilidad**: Es una biblioteca muy versátil que permite interactuar con diferentes tipos de bases de datos, dotando de mucha flexibilidad a la librería.

Ilustración 3-6 Logo de Sequelize

3.3 Generales

En esta sección detallaré qué herramientas han sido utilizadas, que no han sido usadas directamente en el frontend o backend. En su lugar, han aportado un gran valor tanto para el diseño como la organización del proyecto.

3.3.1 Figma

Figma es una herramienta de diseño muy popular entre los diseñadores gracias a su flexibilidad y a la gran cantidad de recursos que ofrece. La aplicación destaca por:

- **Prototipos interactivos**: Permite crear prototipos interactivos dentro de la aplicación facilitando así la visualización de los flujos de usuario.
- **Componentes y sistemas de diseño**: Figma permite crear componentes que se pueden reutilizar en diferentes partes de la aplicación, además, permite crear sistemas de diseño que aportan consistencia y acelera el proceso de creación.
- **Plugins**: Gracias a la amplia comunidad que tiene Figma, hay una gran cantidad de componentes ya creados por esta.

Ilustración 3-7 Logo de Figma

3.3.2 Miro

Miro es una aplicación web muy versátil que funciona como una pizarra digital. Esta permite realizar todo tipo de tareas, en el caso de este proyecto se ha usado para gestionar las diferentes tareas en cada etapa del proyecto. Además, ha sido muy útil para desarrollar la funcionalidad del producto, a la hora de gestionar y priorizar las ideas.

- **Plantillas**: Hay un gran número de plantillas ya creadas que ahorran mucho tiempo a la vez que ayudan a mantenerse organizado.

Ilustración 3-8 Logo de miro

3.3.3 Github

Github es una plataforma de desarrollo colaborativo muy popular entre desarrolladores por su sistema de control de versiones y de gestión de proyectos.

En el proyecto se han creado dos repositorios, uno con el código referente al frontend y otro al backend.

- **Control de versiones**: El sistema de aplicaciones que incluye la aplicación permite rastrear y restaurar cualquier cambio realizado en la aplicación.
- **Integración en otras aplicaciones**: Github se integra en múltiples aplicaciones de manera muy sencilla. En el caso del proyecto, se ha integrado en VSCode.

Ilustración 3-9 Logo de GitHub

3.3.4 Looka

Looka es una inteligencia artificial que permite a los diseñadores crear logos de una manera sencilla. En el proceso escoges el área en el que será usado, paleta de colores y algunos logos de tu gusto. Juntado la información dada, crea múltiples logos que posteriormente se pueden customizar para alcanzar el resultado deseado.

Ilustración 3-10 Logo de Looka

Capítulo 4 - Diseño de la funcionalidad e interfaz

A lo largo de este capítulo y los siguientes iré detallando la ruta seguida a lo largo de las diferentes fases del desarrollo. Este largo proceso se dividirá en tres: diseño de la funcionalidad y de la interfaz, creación de la base de datos, y desarrollo de diseño.

Describir la funcionalidad que se quiere en una aplicación es un punto fundamental a la hora del diseño, ya que este determina el rumbo del resto. El primer paso fue recabar información de qué ofrecían otras aplicaciones similares, tal y como se cuenta en el Capítulo 2.1. En este proceso se recogieron las cualidades que ofrecía cada aplicación para ir seleccionando aquellas que podían interesar al proyecto.

Una vez planteado un primer enfoque de la funcionalidad de la aplicación, se diseñó una entrevista para hacerla usuarios potenciales. De esta manera se buscaba tener una visión más amplia de lo que podría esperar un usuario de la aplicación y a qué le daba una mayor o menor importancia.

Con una visión más completa, ya se procedió a realizar el diseño de la funcionalidad de este proyecto. Iniciando por determinar los actores, la creación de los casos de uso y los flujos de usuario.

Terminado el diseño de la funcionalidad y con los casos de uso en mente, se inició el proceso de creación de la interfaz gráfica. Para el diseño de la interfaz se utilizó la aplicación Figma, una aplicación de diseño que permite hacer el prototipado de una aplicación y hacer un diseño interactivo que poder mostrar. Tras el primer modelado se mostró a un potencial usuario, para que probara la aplicación y de esta manera mejorar tanto la interfaz como la experiencia de usuario.

4.1 Entrevistas

Antes de empezar a realizar las entrevistas se realizó una selección de los entrevistados y una preparación de las preguntas a realizar a lo largo de estas. Los entrevistados seleccionados fueron: Clara López y Rocío Fernández.

4.1.1 Preparación entrevistas

En esta sección estará detallado el modelo de las entrevistas con las preguntas realizadas a cada una de las entrevistadas:

¿Qué edad tienes?

¿Actualmente a qué te dedicas?

¿Has compartido piso alguna vez?

SI ¿Cuánto tiempo llevas haciéndolo?

NO ¿Piensas que un futuro tendrás que hacerlo?

¿Qué problemas piensas que pueden surgir compartiendo piso? ¿Cuál de todos dirías que es el más importante?

¿Consideras que sería útil tener las tareas gestionadas por una aplicación?

¿Qué es lo más importante que debería tener?

¿Piensas que sería útil poder compartir los gastos entre los compañeros de piso?

¿En tu vida personal utilizas alguna aplicación para organizarte?

SI ¿Qué es lo más importante que incluye dicha aplicación?

SI ¿Podrías de cambiar a una aplicación que te ayudará a gestionar tus tareas y las de la casa todo junto?

NO ¿Te has planteado alguna vez usarla? ¿Por qué?

¿Estarías dispuesto a pagar por un servicio así?

¿En qué dispositivos te gustaría que estuviera disponibles?

¿Consideras que podrías usar una aplicación como esta?

¿De todo lo que hemos hablado qué es lo que consideras lo más importante?

4.1.2 Clara López

¿Qué edad tienes?

Tengo 22 años

¿Actualmente a qué te dedicas?

Ahora mismo estudio un grado de contabilidad y finanzas en la Universidad Rey Juan Carlos y llevo haciendo prácticas todo el año.

¿Has compartido piso alguna vez? ¿Cuánto tiempo llevas haciéndolo?

Si, llevo compartiendo piso desde que vina a Madrid a estudiar hace casi 4 años y lo seguiré haciendo cuando termine la universidad.

¿Qué problemas piensas que pueden surgir compartiendo piso? ¿Cuál de todos dirías que es el más importante?

Pues el yo creo que es la relación persona a persona, y que no se creen malos rollos porque a partir de ahí todo se fastidia. Así que depende bastante de como sea cada uno, porque cada uno tiene sus manías y no es fácil vivir con todo el mundo. Seguramente si digo algo más concreto pues que cada uno sea más o menos ordenado con lo suyo y las cosas de la casa suelen ser problemas, sobre todo al principio.

¿Consideras que sería útil tener las tareas de la casa gestionadas por una aplicación?

Pues la verdad que sí, porque ahora nosotros lo hacemos con una pizarra, pero aun así muchas veces no nos acordamos bien de todo o hay problemas.

¿Qué es lo más importante que debería tener?

Pues tampoco necesita demasiado, sobre todo que sea simple para poder organizarte las tareas, y ya que estamos organizarte en general. Pero tampoco necesita más.

¿Piensas que sería útil poder compartir los gastos entre los compañeros de piso?

Seguramente sí, que, aunque nosotros nunca hemos tenido problema con el dinero, creo que sería muy cómodo.

¿En tu vida personal utilizas alguna aplicación para organizarte? ¿Qué es lo más importante que incluye dicha aplicación?

Ahora mismo no utilizo ninguna, aunque sí que he utilizado en otros momentos de mi vida. Lo más importante seguramente que era muy sencilla y con un vistazo nada más entrar ya sabía que tenía que hacer hoy.

¿Podrías de cambiar a una aplicación que te ayudará a gestionar tus tareas y las de la casa todo junto?

Si ofrece más o menos lo mismo que la que usaba seguramente sí, aunque ahora no uso ninguna.

¿Estarías dispuesto a pagar por un servicio así?

No, habiendo tantas aplicaciones gratuitas no creo que pagara por algo así, al menos ahora.

¿En qué dispositivos te gustaría que estuviera disponibles?

Ahora que estoy todo el día con el ordenador, pues casi diría que es lo más importante, pero también en el móvil.

¿Consideras que podrías usar una aplicación como esta?

Sí, pero siempre tengo el problema de que al final las dejo de usar

¿De todo lo que hemos hablado qué es lo que consideras lo más importante?

Que no me complique la vida, que hay algunas aplicaciones que de tantas cosas que le quieren meter lo complican.

4.1.3 Rocío Fernández

¿Qué edad tienes?

Tengo 18 años

¿Actualmente a qué te dedicas?

Estoy estudiando un doble grado en ingeniería física e ingeniería industrial en la Carlos III

¿Has compartido piso alguna vez? ¿Piensas que un futuro tendrás que hacerlo?

No, nunca he compartido piso, pero seguramente lo haga en un futuro unos años al menos después de terminar la universidad.

¿Qué problemas piensas que pueden surgir compartiendo piso? ¿Cuál de todos dirías que es el más importante?

Pues muchos, desde que no te lleves bien con alguien, que sean desordenados, que no hagan lo que les toca, hay muchas cosas que pueden salir mal. Es difícil decir el más importante, pero para mí que no se invada el espacio del otro, porque, aunque te lleves mal podéis como pasar un poco, pero como alguien sea muy desordenado o no haga sus cosas, creo que ahí ya no se puede pasar y es problema asegurado.

¿Consideras que sería útil tener las tareas gestionadas por una aplicación?

Si

¿Qué es lo más importante que debería tener?

No lo tengo claro porque nunca lo he vivido como tal, pero yo diría poder organizarte todo en un sitio. Y que sea muy cómoda de usar porque si la deja de usar uno de la casa ya no funciona.

¿Piensas que sería útil poder compartir los gastos entre los compañeros de piso?

Mucho la verdad que al final puede facilitar muchas cosas.

¿En tu vida personal utilizas alguna aplicación para organizarte? ¿Te has planteado alguna vez usarla? ¿Por qué?

No, nunca he usado una. Si que me he planteado usarlas, pero me suele dar pereza, incluso empecé a usar una, pero a la semana la deje de usar. Puede que si me obligara a usar por el piso empezara a usarla de verdad.

¿Estarías dispuesto a pagar por un servicio así?

No, si no es gratis sería imposible que la usara.

¿En qué dispositivos te gustaría que estuviera disponibles?

En móvil y ordenador que es lo que más uso.

¿Consideras que podrías usar una aplicación como esta?

No lo tengo claro, porque nunca he usado una parecida, pero puede ser. Depende de si nos organizáramos bien el piso, si no seguramente sí.

¿De todo lo que hemos hablado qué es lo que consideras lo más importante?

Lo más importante pues que hay que conseguir que la gente se quede en la aplicación que yo creo que mucha gente empieza y luego la deja.

4.2 Especificación

4.2.1 Actores

Cada usuario que usa esta aplicación tendrá un rol dentro del grupo al que pertenezca. Este determinará las acciones que puede y no puede realizar, siendo administrador el rol con mayor capacidad de acción y usuario con menor.

- Usuario: Este rol permite al usuario ver las tareas de grupo y distribuir las tareas que le estén asignadas dentro de su horario. También le permite crear tareas personarlas y editarlas como quiera, pudiéndolas organizar junto a sus tareas de grupo. El usuario puede añadir gastos y editar los suyos propios, puede pagar sus deudas como recordar a sus deudores que paguen. Por último, puede añadir productos a la lista de la compra y editar aquellos que haya solicitado él.
- Administrador: Este rol permite realizar las mismas acciones que el usuario. Además, permite añadir y editar las tareas del grupo y, por último, permite administrar los ajustes del grupo, así como, expulsar y hacer administradores a los miembros de este.

4.2.2 Casos de uso

Definir detalladamente la funcionalidad de la aplicación es un paso fundamental para el correcto desarrollo de la aplicación. Tras la investigación inicial, las entrevistas y determinar los actores de la aplicación, se listaron los casos de uso deseados y se

asignaron a administrador o a ambos actores, ya que un administrador tiene la misma funcionalidad base que el usuario normal (Casos de usos detallados en Anexos).

- Usuario: registro, login, crear grupo, unirse a un grupo, añadir tarea personal, editar tarea personal, eliminar tarea personal, añadir tarea de grupo, añadir gasto, editar gasto, pagar deuda, recordar deuda, ver una notificación, añadir un producto a la lista de la compra, editar un producto, eliminar un producto, comprar producto, editar dato personal, log out, eliminar cuenta, dejar grupo.
- Administrador: editar tarea de grupo, eliminar tarea de grupo, expulsar miembro, hacer administrador, eliminar grupo.

4.3 Interfaz

Para el diseño de la interfaz de la aplicación se ha usado Figma, una aplicación que permite el desarrollo de prototipos interactivos. Al empezar es fundamental la paleta de colores seleccionada, la elección final fue una paleta clara con predominancia del azul, usando sus complementarios, como el naranja.

En primer lugar, se realizó un mockup para tener una primera imagen de cómo podía ser la aplicación. Este primer mockup, mostraba la estructura de la aplicación, pero era carente de mucho color, además tenía diferentes pestañas emergentes a la hora de crear gastos o poder editarlos. Para mejorar este primer mockup, se realizó un test de usabilidad. Con los resultados del test se remodeló una parte del mockup para alcanzar el prototipo final.

A lo largo de este proceso de diseño, se buscó un logo acorde con la aplicación, para lograrlo se usó una inteligencia artificial que permite crearlos con ella, Looka. Tras muchas pruebas, finalmente se decidió que el logo de la aplicación sería el siguiente.

Ilustración 4-1 Logo de la aplicación

4.3.1 Test de usabilidad

Un test de usabilidad es el paso a seguir una vez ya se posee una primera versión del diseño, pero aun sin pulir. En este se busca observar como el usuario interactúa con el prototipo y obtener feedback de el para así mejorar tanto la interfaz como la experiencia del usuario al interactuar con la aplicación. Para la realización del test se prepararon una serie de tareas que el usuario debería realizar, mientras iba expresando en voz alto lo que pensaba y sentía. El transcurso del test fue el siguiente:

Regístrate:

A ver pulso el botón registro y será eso supongo. Pues sí. Relleno los campos, y bueno me registro no hay mucha complicación.

¿Qué piensas que tiene cada uno de los botones de la barra de navegación?

A ver el botón de inicio ya lo estoy viendo y parece como un resumen de las tareas o de todo en general. Lo de tareas imagino que será para crear tareas y demás, aunque por lo que veo si hay de grupo y personales, bueno supongo que todo se hará ahí. Horario, será literalmente el horario, no creo que tenga mucho misterio. Y gastos pues me imagino que será tipo tricount o algo así para crear gastos, a ver antes he visto la lista de la compra así que imagino que también estará ahí pero no se. Y perfil pues perfil con los datos y la campana esa tendrá notificaciones o algo del estilo, ¿no?

Intenta crear una tarea de grupo:

Pues para crear una tarea de grupo será ir a tareas y a ver qué hay. Ay que feo no me gusta, todo gris (sugerencia: las tareas de cada de un mismo color). Pero bueno aquí está para añadirla.

Intenta crear una tarea personal:

Pues será en la misma pestaña a bueno un poco más abajo, fácil la verdad.

Intenta crear un gasto:

Me imagino que será en gastos. No entiendo por qué hay un botón, me gusta más como esta en las tareas esta como más integrado.

Intenta añadir un producto a la lista de la compra:

Pues si es aquí en esta página, será más abajo, justo pues también fácil. Aunque los productos son muy grandes podrían ocupar un poco menos.

Intenta editar un tarea personal:

Pues me voy a tareas y pulso editar.

Intenta editar un gasto:

Será igual, voy a gastos. Espera no se está solo lo de pagar qué raro, será pulsando en medio, bueno justo. Pero no me gusta esto habría que cambiarlo.

Intenta editar un producto:

Ah bueno esto también es intuitivo, es sencillo.

Intenta eliminar una tarea personal:

Eso lo he visto antes es muy simple, voy y pulso eliminar, deberías poner una confirmación o algo que si no le puedes dar sin querer.

Intenta cerrar sesión:

Pues me imagino que, yendo a perfil, mmm no me convence del todo el diseño. Pero bueno pulso el botón y listo.

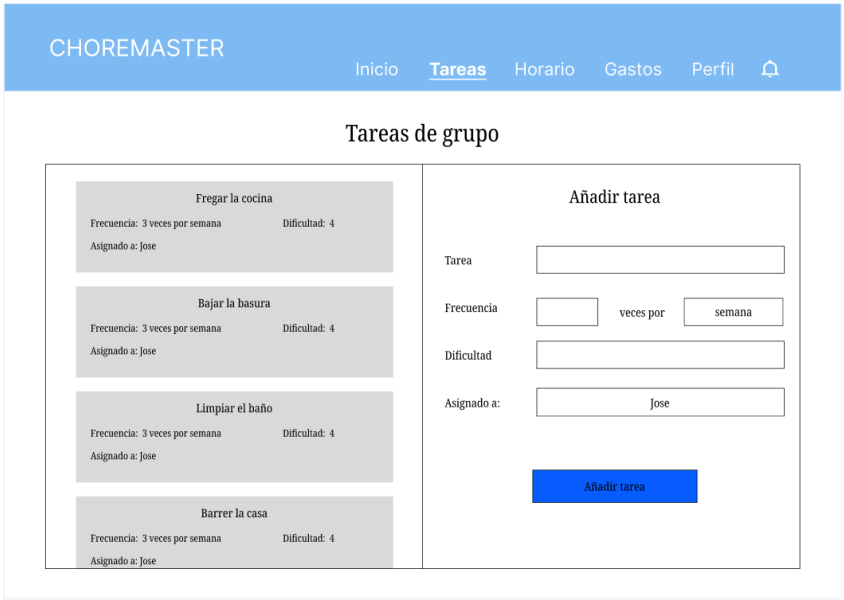
Intenta borrar la cuenta:

Eso estaba al lado de cerrar sesión, así que fácil.

4.3.2 Rediseño

El tras el test de usabilidad ha quedado claro que el prototipo actual es bastante simple y fácil de usar, uno de los prerrequisitos más importantes. El principal problema que se ha encontrado son diseños poco agradables al usuario que empeoran la experiencia de usuario. En esta sección se mostrarán algunas de las secciones tal y como estaban antes del test de usabilidad y tras su rediseño.

- En primer lugar, se asignó a cada usuario un color diferente que cada uno podría elegir. Este sería el color encargado de representar al usuario en cada uno de los apartados de la aplicación. Por ejemplo, las tareas asignadas a dicho usuario estarán en un contenedor de dicho color. El cambio fue el siguiente:



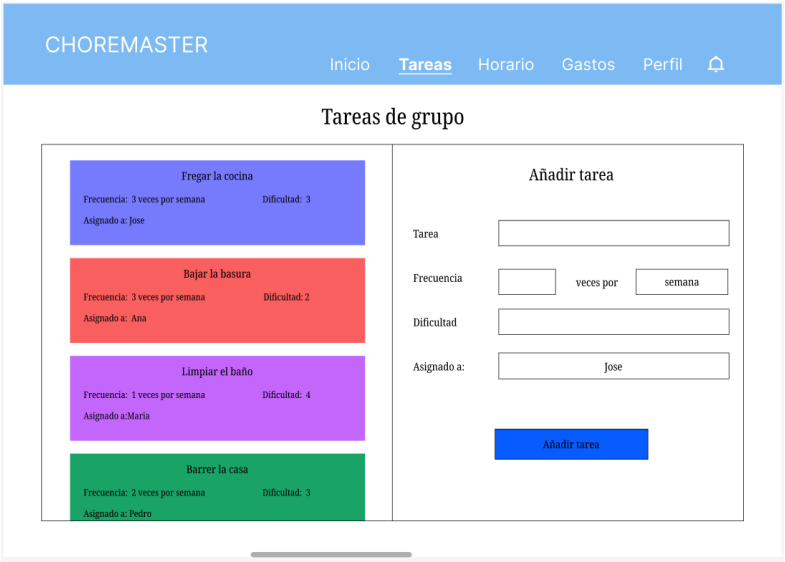


Ilustración 4-2 Cambio diseño pestaña de tareas

- Por otro lado, se rediseñó la parte de la pestaña gastos referente a estos, sin modificar la lista de la compra. En vez de tener un botón de crear una nueva tarea, tiene tres apartados, uno donde marca las deudas actuales, otro con el historial de gastos y uno ultimo para crear un gasto. De esta manera el resultado es más atractivo e incluso simplifica su interacción.

V

Ilustración 4-3 Cambio diseño pestaña gastos

- Aparte de los cambios importantes en el diseño, se han añadido pequeñas mejoras para la experiencia de usuario como puede ser el diseño de los productos en la lista de la compra o redondear botones.

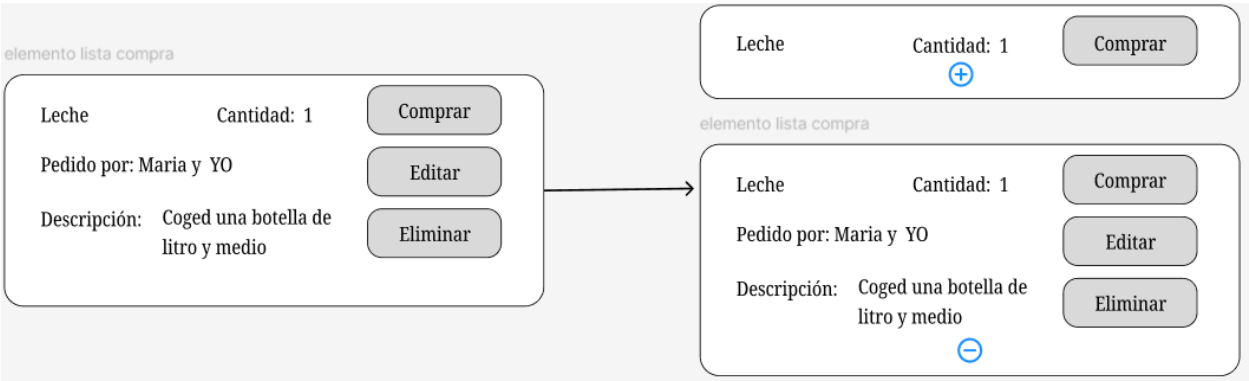


Ilustración 4-4 Cambio diseño en producto

Capítulo 5 - Desarrollo de la aplicación

En este capítulo detallaré la fase del proyecto dedicada al desarrollo del código de la aplicación. En las diferentes secciones se explicará la estructura de directorios de la aplicación, las diferentes vistas y la lógica principal, además el contenido estará dividido en dos secciones diferentes: backend y frontend.

5.1 Backend

El backend se ha desplegado en el puerto 8000 durante el desarrollo de la aplicación. Para su desarrollo se ha aplicado el patrón Modelo-Controlador, el cual separa responsabilidades en un modelo que representa los requisitos y un controlador que actualiza y maneja dicho modelo.

En primer lugar, se realizará una pequeña explicación de la estructura de directorios, para continuar con la explicación de dos partes de la lógica básica para el funcionamiento de la aplicación.

5.1.1 Estructura de directorios

La estructura del backend refleja el patrón modelo controlador mencionado anteriormente y es la siguiente:

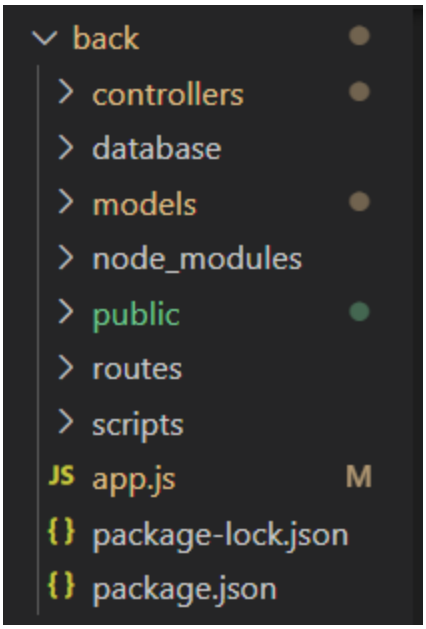


Ilustración 5-1 Estructura directorios backend

- **Controllers**: En este directorio se encuentra el archivo Controller.js que se encarga de la interacción con la base de datos.
- **Database**: En este directorio se encuentra la declaración de la instancia de la base de datos.
- **Models**: En este directorio se encuentran todos los modelos de Sequelize que representan la base datos junto con las relaciones entre estos.
- **Node_modules**: En este directorio se encuentran todas las dependencias del proyecto.
- **Public**: En este directorio se encuentran los archivos públicos del servidor. En concreto es donde se guardarán las fotos de perfil de los usuarios.
- **Routes**: En este directorio se encuentran especificadas las rutas a través las cuales el frontend hace sus peticiones al backend.
- **Scripts**: En este directorio se encuentran los archivos que contienen funciones necesarias para el funcionamiento de la aplicación.

5.1.2 Lógica básica

5.1.2.1 Recordar y desmarcar tareas

En una aplicación de organización es fundamental poder recordar que el usuario ha de hacer una tarea cuando llega su hora. A su vez, también es fundamental que las tareas semanales o mensuales dejen de estar hechas al final de la semana o del mes respectivamente. Para implementar estas funcionalidades se ha utilizado la biblioteca node-cron, esta permite realizar acciones de manera repetitiva.

Haciendo uso de esta biblioteca, cada minuto se llama a una función que hace una consulta comprobando si hay alguna tarea que ha de empezar en dicho momento. En caso de que la consulta devuelva alguna tarea se crea un recordatorio para el usuario en cuestión.

Además, una vez al final de la semana y del mes se llama a otra función que realiza una consulta para marcar todas las tareas como no hechas y, que de esta manera, estén listas de nuevo para la siguiente semana.

5.1.2.2 Tratamiento de imágenes

Las fotos de perfil no son parte fundamental de la aplicación, pero mejoran la experiencia de usuario haciendo la aplicación más personal. Para implementar esta funcionalidad se ha utilizado la biblioteca multer que permite un fácil tratamiento de los archivos, así como filtrar los archivos deseados de los indeseados.

En primer lugar, hay que configurar la carpeta de destino y el nombre del archivo que se va a asignar. A continuación, continuación se crea la instancia y ya se pueden subir los archivos con facilidad, esto se combina con una consulta que actualiza la ruta a la foto de perfil en la base de datos.

5.2 Frontend

5.2.1 Estructura de directorios

La estructura del directorio es fundamental para mantener el código limpio modular y escalable, por ello, el criterio para estructurar los directorios ha sido la

funcionalidad. Así se han agrupado aquellos archivos cuyo código cumple un rol similar en la aplicación. La estructura resultante ha sido la siguiente:

- **BD**: En este directorio están aquellos archivos que se agrupan la interacción con el backend.
- **Components:** En este directorio están los archivos que proporcionan componentes que forman las distintas vistas.
- **Hooks**: En este directorio se encuentran hooks que encapsulan lógica reutilizable que es compartida entre componentes.
- **Pages**: En este directorio están los archivos que representan las páginas completas de la aplicación. Dentro de este directorio se encuentran otros como chores o profile, estos contienen un archivo que representa una página y su archivo .css correspondiente.
- **Utils**: En este directorio están funciones auxiliares que son reutilizadas en diferentes partes del proyecto.

5.2.2 Lógica básica

En el frontend hay mucha lógica que se encarga de manejar los estados y de la interacción entre la interfaz y el backend. En este punto únicamente se desarrollará los diferentes hooks usados y la lógica usada para gestionar las deudas que asegura que entre dos usuarios no haya más de una deuda.

5.2.2.1 AppContext

Este hook, se encarga de aportar la información básica necesaria en toda la aplicación para no tener que pasarla con props entre componentes. Los valores que este hook provee son:

- User: contiene el id del usuario que ha iniciado sesión.
- Group: contiene el id del grupo al que pertenece el usuario.
- Color: contiene el color del usuario.

- Admin: contiene un booleano que muestra si el usuario es o no administrador.

5.2.2.2 AlertContext

Este hook se encarga de gestionar las alertas dentro de la aplicación. Para ello, Se usa el hook que maneja el estado de la alerta como un intermediario entre el resto de los componentes de la aplicación y el componente de la alerta.

El componente Alert, que muestra la alerta hace uso de los parámetros que ofrece el AlertContext. Mediante estos parámetros y la biblioteca Chakra, que ofrece unas alertas prediseñadas se muestra la alerta deseada por el componente que lo solicite.

5.2.2.3 Crear deudas

Gestionar las deudas entre los usuarios es fundamental en la aplicación. Para ello, bastaría con añadir deudas cada vez y eliminarlas al pagarlas, pero podría resultar en una cantidad enorme de datos innecesarios porque toda esa información se podría almacenar en una sola deuda. Para solucionar este problema, se implementado una función que elimina las deudas existentes y crea una nueva con los nuevos datos que se desean.

5.2.3 Vistas

Para crear las diferentes vistas e interfaces de la aplicación se partió de los diseños ya creados en la fase de diseño en Figma. Las interfaces están divididas en cinco páginas: Iniciar sesión, Inicio, Tareas, Horario, Gastos y Perfil.

5.2.3.1 Iniciar de sesión

Al entrar en la aplicación el primer paso que ha de hacer el usuario es iniciar sesión o registrarse en caso de que no tenga cuenta aún. En el caso de que el usuario tenga una cuenta, este ha de introducir su nickname junto con su contraseña. Internamente se procederá a comprobar si las credencias introducidas se corresponden con las de algún usuario, si lo hicieran se lleva al usuario a la página de Inicio.

Ilustración 5-22 Login

En el caso de que el usuario no poseyera una cuenta en la aplicación este debería crear una nueva. Para ello debería pulsar el botón de registro, rellenar los campos correspondientes y pulsar el botón de registro. Internamente se comprobará que el email tiene un formato correcto, esto se hará haciendo uso de la biblioteca Yup junto con Formik, la librería usada para crear los formularios. Si hubiera algún error en cualquier campo este mostraría debajo del recuadro del input indicándoselo al usuario, por otro lado, si todos los campos cumplen con los requisitos se creará la nueva cuenta y se trasladará al usuario a la pestaña de Inicio.

Ilustración 5-33 Registro

5.2.3.2 Inicio

Una vez dentro de la aplicación la primera pestaña que el usuario ve es la pestaña de inicio, esta sirve de resumen a toda la aplicación. En la parte superior de la página se ve un resumen de como llevan las tareas de la semana cada uno de los integrantes mediante una barra de progreso.

Debajo del resumen se pueden apreciar dos tablas, la tabla de la izquierda muestra las próximas tareas de cada usuario y permite recordarle que debe hacer una tarea que se le haya pasado, al hacerlo el usuario recibirá una notificación y un correo recordándoselo. A su lado se encuentra otra tabla, en esta se puede ver las próximas tareas del usuario y una vez que el usuario las termine las podrá marcar como completadas.

Finalmente, en la parte inferior de la página, se encuentra el balance del usuario con sus ingresos, gastos, deudas y dinero que le deben. Por último, a su derecha se encuentra la lista de la compra, en ella se aprecian todos los productos con una pequeña descripción. Además, se permite comprarlos directamente en esta página. Al pulsar el botón comprar, aparece una ventana de dialogo que pide el precio final del producto, y al confirmarlo se crea una deuda entre los usuarios que pidieron el producto con el que lo compra.

Ilustración 5-44 Inicio

5.2.3.3 Tareas

La pestaña de Tareas permite gestionar todos los aspectos de las tareas, tanto las grupales como las personales. En la parte superior de la página, se ve un cuadro con las tareas grupales que está dividido en dos, uno con las diferentes tareas grupales y otro con un formulario para añadir una tarea. En el caso de que el usuario sea un administrador en cada una de las tarjetas verá los botones de editar y eliminar las tareas Si pulsara el botón de editar el formulario de añadir tarea cambiaría por uno de editar la tarea con todos los datos de esta ya rellenados.

En la parte inferior de la página, se encuentran las tareas personales, el formato es similar a las tareas de grupo. Cada usuario verá tanto sus tareas personales como las grupales que tenga asignadas, en todas aparecerá un botón de editar y en las tareas

personales también uno de eliminar. En el caso que el usuario quiera editar una tarea personal verá los campos de nombre, frecuencia y las horas para organizarse, pero si se quiere editar una tarea grupal, solo se podrá modificar las horas.

Ilustración 5-55 Tareas

5.2.3.4 Horario

Para poder organizarse es necesario tener un horario que te permita ver que debes hacer a lo largo de la semana y del mes. Al entrar en la pestaña se ve el calendario del mes y en cada día se pueden ver las tareas que ha de hacer cada día el usuario junto con la hora en la que se ha de hacer.

En caso de que el usuario quiera ver el horario de un día en concreto, simplemente ha de pulsar sobre ese día. Al pulsar, la página cambiará a un horario donde se ve el día seleccionado y las 24 horas del días con las tareas del día en su franja

horaria. Esta página es meramente informativa para el usuario y que pueda ver cómo se está organizando y sus quehaceres del mes.

Ilustración 5-66 Horario

5.2.3.5 Gastos

La pestaña permite gestionar tanto los gastos del grupo como la lista de la compra. En la parte superior de la página se encuentra el cuadro para gestionar los gastos, este dividida en dos partes tal y como es el cuadro de la página de Tareas. En la parte izquierda se ve un resumen de las deudas de cada persona viendo en un diagrama de barras cuanto debe o le deben a cada usuario.

En la parte derecha del cuadro, hay tres secciones: actuales, historial y nuevo gasto. La sección de actuales muestra las deudas de cada usuario y permite tanto pagar la deuda en caso de que sea tuya o recordársela al compañero en caso de que le deban al usuario. En la segunda sección, historial, se ven la lista de los gastos del grupo, si el usuario pulsa un gasto que haya pagado, se verá un formulario que permitirá editar el gasto en caso de que hubiera algún error al añadirlo. Por último, la sección de nuevo gasto consiste en un formulario, que tras rellenarlo correctamente permite añadir un nuevo gasto que actualiza todo lo anterior.

En la parte inferior de la página se encuentra la lista de la compra, esta sigue el mismo formato que los cuadros de Tareas. En la parte de la izquierda se encuentra una

serie de tarjetas cada una asociada a un producto que permite comprar, editar o eliminar un producto que haya pedido el usuario. En la parte derecha se encuentra un formulario que permite al usuario añadir un nuevo producto a la lista de la compra.

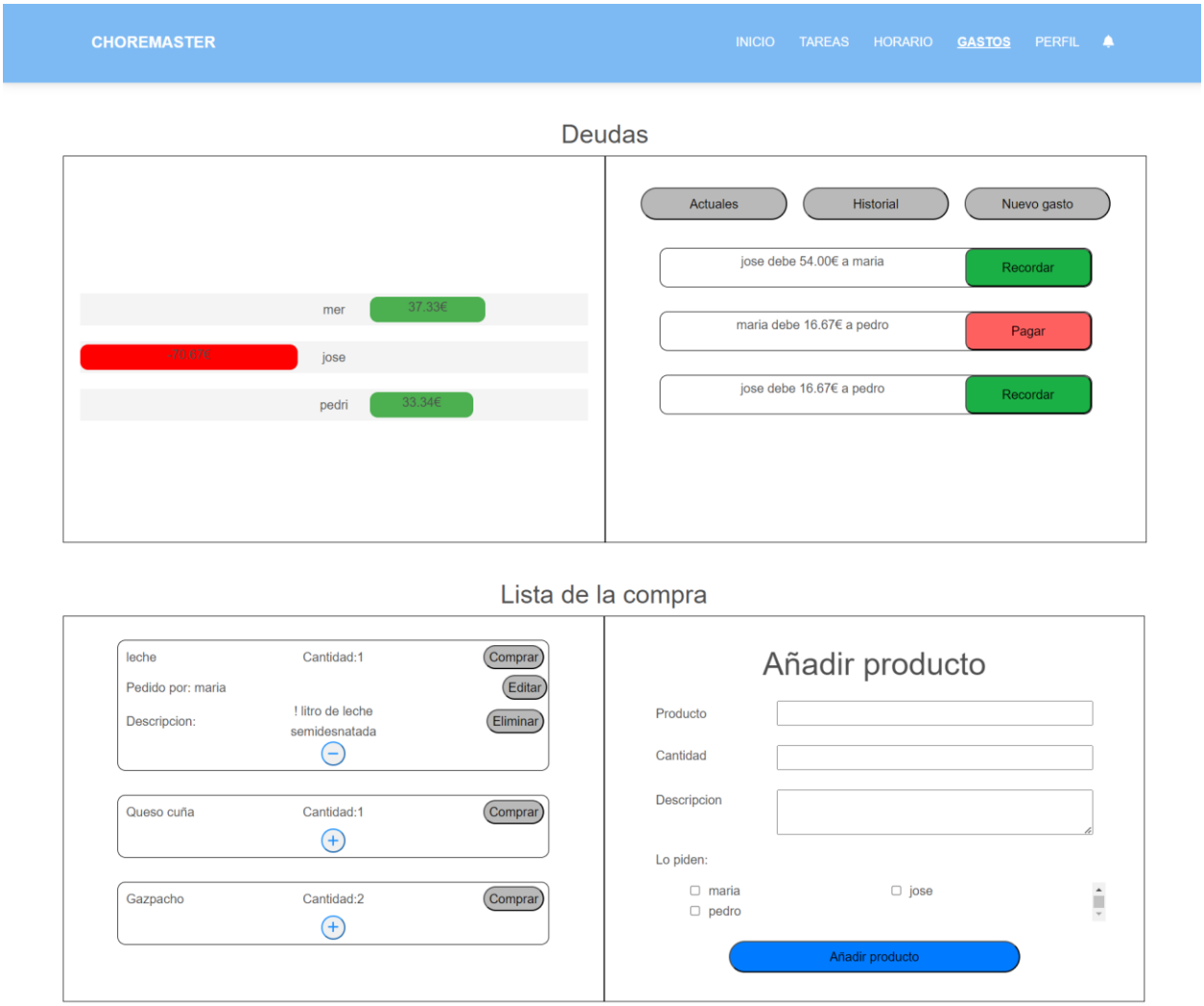


Ilustración 5-77 Gastos

5.2.3.6 Perfil

La pestaña de perfil sirve al usuario para poder visualizar toda su información personal como puede ser el nombre, nickname o contraseña. Como se puede apreciar en la figura, hay un botón que permite cambiar cada uno de los diferentes campos. Además, se puede visualizar la foto de perfil del usuario en la parte superior, permitiendo su modificación como el resto de los campos. Debajo de la información personal, se encuentran los botones de cerrar sesión y eliminar la cuenta. Todas estas acciones

modifican la información del usuario en la base de datos, afectando en algunos casos, como el color, a toda la aplicación.

Debajo de la información personal se encuentra la información de grupo. Esta muestra el nombre del grupo, los miembros y su código de invitación. Para modificar información en esta sección es necesario tener el rol de administrador, teniendo ese rol, se permite modificar el nombre del grupo, expulsar miembros y hacer administrador a otros usuarios. Por último, están los botones de dejar el grupo y eliminar el grupo, siendo el segundo exclusivo de administradores.

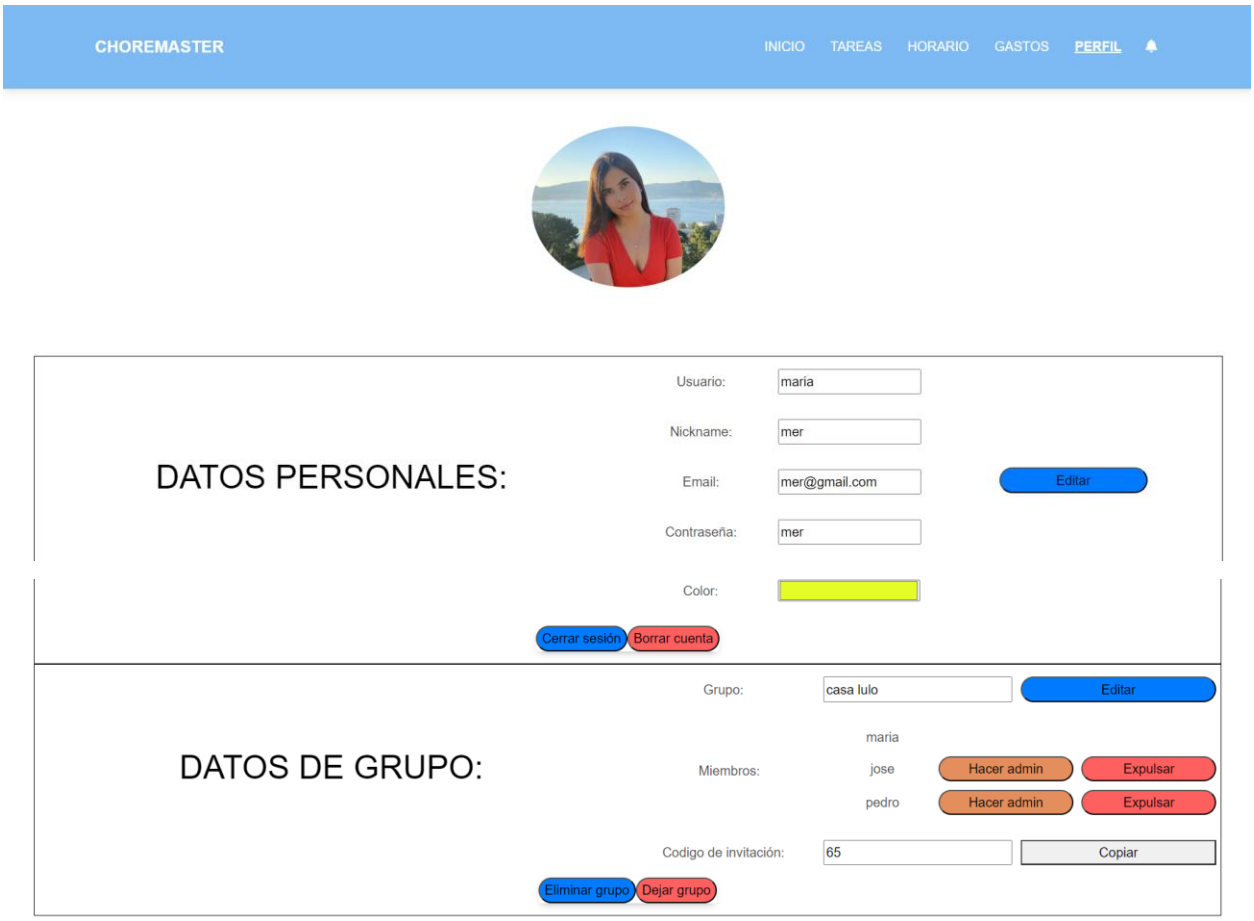


Ilustración 5-88 Perfil

5.2.3.7 Home

Este componente es el encargado de la navegación en la aplicación, este está provisto de una serie de botones que permite desplazarse entre las pestañas. Además, tiene dos elementos extra, el nombre de la aplicación, ChoreMaster; que traslada al

usuario a la pestaña de inicio, y un icono de una campana, este despliega una pequeña pestaña con las notificaciones del usuario.

Ilustración 5-9 Home

5.2.3.8 Notificaciones

Este componente se encarga de mostrar las notificaciones del usuario en cuestión. Este despliega las notificaciones en una pestaña fijada en la parte superior derecha de la pantalla. Desde esta se permite eliminar las notificaciones individualmente usando el icono con forma de X, además para mejorar la experiencia de usuario, se permite eliminar todas las tareas simultáneamente, esto se hace pulsando Eliminar todas. Por último, la pestaña tiene una X en la parte superior derecha que permite cerrar la pestaña.

Capítulo 6 - Conclusiones y trabajo futuro

El desarrollo de este proyecto ha sido un arduo camino en el que no tenía ninguna experiencia. Desde el principio aparecieron numerosas dificultades empezando por el desconocimiento en el uso de la mayoría de las tecnologías usadas como React o Node.js.

Poco a poco se consiguió ir superando los diferentes problemas que aparecieron dejando a su vez una lecciones muy valiosas sobre cómo desarrollar una aplicación web y cómo hay que llevar un proyecto de esta envergadura a lo largo del tiempo. Estos aprendizajes no han hecho más que motivarme a aprender más del tema y a seguir desarrollando aplicaciones para poder aplicarlos.

Viendo en retrospectiva, a lo largo del proyecto se cometieron muchos errores que han complicado el camino y probablemente hayan empeorado el resultado de este, pero han dejado múltiples aprendizajes. A pesar de todos los problemas que surgieron, se consiguió ir cumpliendo los plazos marcados desde el principio y crear una aplicación que cumplía los objetivos principales. Una aplicación sencilla que permitiera integrar en una misma aplicación la gestión de tareas y gastos en un solo lugar.

El único resquemor que me queda en torno al proyecto son algunas ideas sobre nueva funcionalidad que se me fue ocurriendo en la parte final del proyecto, pero por falta de tiempo se ha tenido que quedar en el tintero. Algunas de estas las expondré en la última sección de este documento.

6.1 Trabajo futuro

Al inicio del proyecto se realizó una investigación para determinar qué funcionalidad se debería incluir en la aplicación. Pero durante el transcurso del desarrollo he podido ver diferentes formas de mejorar la aplicación mediante la implementación de diferentes funcionalidades.

Una de las ideas que más aportarían a la aplicación sería permitir que todos pudieran indicar en un horario grupal que días estarían y cuales no, o permitir poner

eventos en un día en concreto. Otra idea sería crear un chat de grupo donde todos pudieran hablar entre ellos.

Viendo el público objetivo de la aplicación está muy claro que también sería importante desarrollar la aplicación para móvil. Los diseños para estos dispositivos se llegaron a realizar, pero no hubo tiempo suficiente para implementar los diseños, esto se detallará en los Apéndices (Diseño de móvil).

Por último, surgió la idea de permitir que cada persona estuviera en más de un grupo. Inicialmente no se pensó en esta idea y aunque se podría implementar de manera sencilla, la idea llegó demasiado tarde.

Conclusions and future work

The development of this project has been an arduous journey in which I had no prior experience. From the beginning, numerous difficulties appeared, starting with my unknow with most of the technologies used, such as React or Node.js.

Gradually, I managed to overcome the various problems that emerged, leaving behind valuable lessons on how to develop a web application and how to manage a project of this magnitud. These learnings have only motivated me to dive deeper into the subject and continue developing applications to apply them.

Looking back, many mistakes were made throughout the project that complicated the process and likely worsened the result. However, they also provided multiple lessons. Despite all the challenges that arose, I was able to meet the deadlines set from the beginning and create an application that fulfilled the main objectives. It's a simple application that integrates task and expense management in one place.

The only regret I have about the project are some ideas for new features that came to mind towards the end of the project. Unfortunately, due to time constraints, they had to be left unfinished. I will present some of these ideas in the final section of this document.

Future Work

At the start of the project, research was conducted to determine what functionality should be included in the application. However, during the development process, I realized various ways to improve the application by implementing different features.

One of the ideas that would greatly enhance the application is to allow everyone to indicate on a group schedule which days they would be at home and which they wouldn't, or to enable event scheduling on specific days. Another idea is to create a group chat where everyone can communicate with each other.

Considering the target audience of the application, it's clear that developing a mobile version would also be important. Designs for these devices were made, but there

wasn't enough time to implement them, which will be detailed in the Appendices (Diseño de móvil).

Lastly, the idea of allowing each person to be part of more than one group emerged. Initially, this idea wasn't considered, and although it could be implemented easily, it came too late in the process.

BIBLIOGRAFÍA

[1] Comunidad de React en DEV [En línea]. Available: https://dev.to/t/react

[2] Developer mozilla «developer.mozilla.org» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs?trk=public_post_comment-text

[3] [En línea]. Available: https://kinsta.com/es/blog/angular-vs-react/

[4] Stackoverflow «https://stackoverflow.com» [En línea] Available: https://stackoverflow.com/questions/78939568/xampp-mysql-not-running

[5] [En línea] Available: https://coliving.com/es/blog/the-ultimate-guide-to-shared-apartments-what-you-need-to-know

[6] [En línea] Available: https://www.viko.net/innovacion/por-que-elegir-figma-para-todo-y-no-solo-para-diseno-de-interfaces

[7] [En línea] Available: https://blog.hubspot.es/website/framework-desarrollo-web

[8] Formik, « https://formik.org » [En línea]. Available: https://formik.org/docs/tutorial

[9] Chakra «https://v2.chakra-ui.com» [En línea]. Available: https://v2.chakra-ui.com/docs/components

[10] [En línea]. Available: https://www.npmjs.com/package/node-cron

APÉNDICES

Apéndice A - Código del proyecto

En el proyecto se ha usado Github para guardar y gestionar el código del proyecto y Figma para crear los diseños y prototipos de la aplicación. A continuación, adjunto los repositorios donde se encuentran estos:

Frontend: front

Backend: back

Diseño: diseño en Figma

Apéndice B - Diseño de móvil

El proyecto inicialmente estaba pensado para que se usará en un ordenador. Pero a su vez, se realizaron diseños para visualizar la aplicación en un dispositivo móvil por si daba tiempo suficiente. A continuación, se adjunta un enlace al diseño en Figma.

Login

Ilustración 6-1 Diseño inicio sesion móvil

Ilustración 6-2 Diseño registro móvil

Inicio

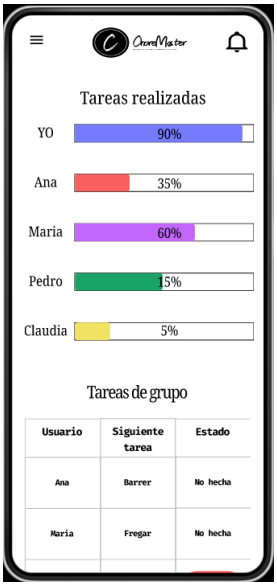


Ilustración 6-3 Diseño resumen tareas móvil

Ilustración 6-4 Diseño balance móvil

Ilustración 6-5 Diseño navegación móvil

Tareas

Ilustración 6-6 Diseño tareas grupales móvil

Ilustración 6-7 Diseño tareas personales móvil

Gastos

Ilustración 6-8 Diseño pestaña gastos móvil

Perfil

Ilustración 6-9 Diseño datos personales móvil

Ilustración 6-10 Diseño datos grupales móvil

Apéndice C - Especificación casos de uso

Registro

- Actores: Usuario.
- Objetivos: Crear una cuenta en la aplicación.
- Precondiciones: No haber iniciado sesión.
- Salidas:
    - Notificación de registro exitoso.
    - Cambio de página inicio.
- Postcondiciones:
    - Cuenta creada correctamente.
- Secuencia normal:
    - Introducir los datos que se pide en el formulario (nombre, nickname, email, contraseña y color a usar).
    - Pulsar botón de registro.

- Secuencia alternativa:
    - Algún problema con el registro aparece notificación que lo indica.

Login

- Actores: Usuario, administrador.
- Objetivos: Iniciar sesión con una cuenta existente.
- Precondiciones: No haber iniciado sesión.
- Salidas:
    - Cambio a página de inicio.
- Postcondiciones:
    - Sesión iniciada.
- Secuencia normal:
    - Introducir su nickname y su contraseña.
    - Pulsar botón de inicio de sesión.
- Secuencia alternativa
    - Error en el inicio de sesión, aparece notificación que lo informa.

Crear un grupo

- Actores: Usuario.
- Objetivos: Crear un nuevo grupo de casa.
- Precondiciones:
    - Haber iniciado sesión.
    - No pertenecer a ningún grupo.
- Salidas:
    - Notificación de grupo creado.
    - Aparece información de grupo.

- Postcondiciones:
    - Grupo creado exitosamente.
    - Usuario se convierte en administrador del grupo creado.
- Secuencia normal:
    - Tras iniciar sesión un usuario sin grupo aparece un cuadro de dialogo.
    - Introducir nombre de grupo.
    - Pulsar crear.
- Secuencia alternativa:
    - Ir a sección de perfil.
    - Pulsar unirme a un grupo.
    - Introducir nombre de grupo.
    - Pulsar crear.

Unirse a un grupo

- Actores: Usuario.
- Objetivos: Unirse a un grupo ya existente.
- Precondiciones:
    - Haber iniciado sesión.
    - No pertenecer a ningún grupo.
- Salidas:
    - Notificación de unirse exitosamente.
    - Aparece información de grupo.
- Postcondiciones:
    - Usuario pertenece al grupo.

- Secuencia normal:
    - Introducir el nombre del grupo o su código.
    - Pulsar unirse.
- Secuencia alternativa:
    - Error al unirse, aparece una notificación avisando.

Añadir tarea personal

- Actores: Usuario, administrador.
- Objetivos: Crear una tarea personal.
- Precondiciones:
    - Haber iniciado sesión.
- Salidas:
    - Notificación tarea creada.
    - Aparece la tarea en la lista de tareas.
    - Se vacía el formulario.
- Postcondiciones:
    - Se crea con éxito la tarea.
- Secuencia normal:
    - Acceder a la pestaña de tareas.
    - Rellenar los campos de la tarea.
    - Pulsar el botón de crear.
- Secuencia alternativa:
    - Error al crear la tarea, aparece notificación que lo indica.
    - Error, datos inválidos, aparece un mensaje que lo indica.

Editar tarea personal

- Actores: Usuario, administrador.
- Objetivos: Editar una tarea personal ya existente.
- Precondiciones:
    - Haber iniciado sesión.
    - Existencia de al menos una tarea personal.
- Salidas:
    - Notificación de tarea editada con éxito.
    - Cambios realizados en la tarjeta de la tarea.
- Postcondiciones:
    - La tarea se edita correctamente en la base de datos.
- Secuencia normal:
    - Acceder a la pestaña de tareas.
    - Buscar la tarea a editar y pulsar su botón editar.
    - Modificar los valores que se quieren editar.
    - Pulsar botón editar.
- Secuencia alternativa:
    - Pulsar el botón cancelar para dejar de editar la tarea.
    - Error al editar, aparece notificación que lo indica.
    - Error, datos inválidos, aparece un mensaje que lo indica.

Eliminar tarea personal

- Actores: Usuario, administrador.
- Objetivos: Eliminar una tarea personal.
- Precondiciones:
    - Haber iniciado sesión.

- Tener al menos una tarea personal asociada al usuario.

- Salidas:
  - Notificación de eliminado con éxito.
  - Desaparece la tarea de la lista de tareas.
- Postcondiciones:
  - Se elimina la tarea de la base de datos.
- Secuencia normal:
  - Acceder a la pestaña tareas.
  - Buscar la tarea a eliminar y pulsar su botón de eliminar.
  - Confirmar la eliminación.
- Secuencia alternativa:
  - Error al eliminar, aparece una notificación que lo indica.

Añadir tarea de grupo

- Actores: Usuario, administrador.
- Objetivos: Crear una tarea para todo el grupo.
- Precondiciones:
  - Haber iniciado sesión.
  - Estar en un grupo.
- Salidas:
  - Notificación de tarea creada exitosamente.
  - Aparece la tarea en la lista de tareas de grupo, y si se la ha asignado el mismo, aparece en su lista personal.
- Postcondiciones:
  - Se crea la tarea en la base de datos.

- Secuencia normal:
    - Acceder a la pestaña de tareas,
    - En el apartado tareas en grupo, rellenar los campos de añadir tarea.
    - Pulsar el botón crear.
- Secuencia alternativa:
    - Error al crear la tarea, aparece una notificación que lo indica.
    - Error, datos inválidos, aparece un mensaje que lo indica.

Editar tarea de grupo

- Actores: Administrador.
- Objetivos: Editar una tarea del grupo.
- Precondiciones:
    - Haber iniciado sesión.
    - Existencia de al menos una tarea del grupo.
- Salidas:
    - Notificación de tarea editada correctamente.
    - Tarea editada de la lista de tareas.
- Postcondiciones:
    - Tarea editada en la base de datos.
- Secuencia normal:
    - Acceder a la pestaña tareas.
    - Buscar la tarea a editar y pulsar el botón editar.
    - Modificar los datos de la tarea en los campo que aparecen.
    - Pulsar el botón editar.

- Secuencia alternativa:
    - Pulsar el valor cancelar para no editar la tarea.
    - Error al editar la tarea, aparece una notificación que lo indica.
    - Error, datos inválidos, aparece un mensaje que lo indica.

Eliminar tarea de grupo

- Actores: Administrador.
- Objetivos: Eliminar una tarea de grupo.
- Precondiciones:
    - Haber iniciado sesión.
    - Existencia de una tarea de grupo.
- Salidas:
    - Notificación de eliminada exitosamente.
    - Desaparece la tarea de la lista, si la tarea estaba asignada al administrador, también desaparecerá de la lista personal.
- Postcondiciones:
    - Se elimina la tarea de la base de datos.
- Secuencia normal:
    - Acceder al apartado de tareas.
    - Buscar la tarea y pulsar el botón de eliminar.
    - Confirmar la eliminación.
- Secuencia alternativa:
    - Error en la eliminación, aparece una notificación que lo indica.

Añadir gasto

- Actores: Usuario, administrador.

- Objetivos: Añadir un gasto.
- Precondiciones:
    - Haber iniciado sesión.
    - Estar en un grupo.
- Salidas:
    - Notificación con gasto creado exitosamente.
    - Balance ajustado.
    - Aparece el nuevo gasto
- Postcondiciones:
    - Se añade el gasto a la base de datos.
    - Se actualizan los balances generales.
- Secuencia normal:
    - Acceder al apartado de gastos.
    - Seleccionar el apartado de nuevo gasto.
    - Rellenar los campos.
    - Pulsar botón crear.
- Secuencia alternativa:
    - Error al crear el gasto, aparece una notificación que lo indica.
    - Error, datos inválidos, aparece un mensaje que lo indica.

Editar gasto

- Actores: Usuario, administrador.
- Objetivos: Editar un gasto del usuario.
- Precondiciones:
    - Haber iniciado sesión.

    - Existencia de un gasto pagado por el usuario
- Salidas:
    - Notificación de gasto creado.
    - Actualización de balance.
    - Actualización de la vista del gasto.
- Postcondiciones:
    - Gasto actualizado en la base de datos.
    - Balance actualizado en la base de datos.
- Secuencia normal:
    - Acceder al apartado de gastos.
    - Seleccionar el apartado de historial.
    - Seleccionar el gasto a editar.
    - Modificar los campos.
    - Pulsar botón editar.
- Secuencia alternativa:
    - Error, datos inválidos, aparece un mensaje que lo indica.
    - Error al editar, aparece una notificación que lo indica.

Pagar deuda

- Actores: Usuario.
- Objetivos: Marcar una deuda como pagado.
- Precondiciones:
    - Haber iniciado sesión.
    - Existencia de una deuda a pagar por el usuario.
- Salidas:

    - Notificación de deuda pagada correctamente.
    - Balance actualizado.
    - Desaparece la deuda de la lista de deudas.
- Postcondiciones:
    - Deuda marcada como pagada en la base de datos.
    - Balance actualizado en la base de datos.
- Secuencia normal:
    - Acceder al apartado de gastos.
    - Seleccionar el apartado deudas.
    - Pulsar el botón pagar.
- Secuencia alternativa:
    - Error al pagar, aparece notificación que lo indica.

Recordar deuda

- Actores: Usuario.
- Objetivos: Recordar una deuda a un usuario.
- Precondiciones:
    - Haber iniciado sesión.
    - Existencia de una deuda que han de pagar al usuario.
- Salidas:
    - Notificación de deuda recordada.
- Postcondiciones:
    - Se crea una notificación al usuario que ha de pagar
- Secuencia normal:
    - Acceder al apartado de gastos.

    - Seleccionar la sección de deudas.
    - Pulsar el botón de recordar de la deuda en cuestión.
- Secuencia alternativa:
    - Error al recordar, aparece una notificación que lo indica.

Ver una notificación

- Actores: Usuario.
- Objetivos: ver las notificaciones del usuario.
- Precondiciones:
    - Haber iniciado sesión.
- Salidas:
    - Aparece una ventana con las notificaciones.
- Postcondiciones:
- Secuencia normal:
    - Se pulsa el botón de la campana
    - (Se elimina la notificación pulsando la X)
- Secuencia alternativa:

Añadir producto a la lista de la compra

- Actores: Usuario.
- Objetivos: añadir un producto a la lista de la compra.
- Precondiciones:
    - Haber iniciado sesión.
    - Estar en un grupo.
- Salidas:
    - Notificación de producto añadido correctamente.

- Se añade el producto a la lista.

- Postcondiciones:
  - Se añade el producto a la lista de la compra del grupo en la base de datos.
- Secuencia normal:
  - Acceder al apartado de gastos.
  - Ir a apartado de nuevo producto y rellenar los campos.
  - Pulsar el botón crear.
- Secuencia alternativa:
  - Error al crear el producto, aparece una notificación que lo indica.
  - Error, datos inválidos, aparece un mensaje que lo indica.

Editar producto

- Actores: Usuario.
- Objetivos: Editar un producto existente.
- Precondiciones:
  - Haber iniciado sesión.
  - Pertenecer a un grupo.
  - Existencia de al menos un producto pedido por el usuario.
- Salidas:
  - Notificación producto editado correctamente.
  - Producto editado en la lista.
- Postcondiciones:
  - Producto editado correctamente en la base de datos.
- Secuencia normal:

    - Acceder al apartado de gastos.
    - Buscar el producto a editar y pulsar su botón editar.
    - Modificar los campos deseados.
    - Pulsar el botón editar.
- Secuencia alternativa:
    - Error al editar, aparece una notificación que lo indica.
    - Error, datos inválidos, aparece un mensaje que lo indica.
    - Pulsar botón cancelar y no editar el producto.
-

Eliminar producto

- Actores: Usuario.
- Objetivos: Eliminar un producto de la lista de la compra.
- Precondiciones:
    - Haber iniciado sesión.
    - Pertenecer a un grupo.
    - Existencia de un producto pedido por el usuario.
- Salidas:
    - Desaparición del producto de la lista.
    - Notificación producto eliminado correctamente.
- Postcondiciones:
    - Producto eliminado de la base de datos.
- Secuencia normal:
    - Acceder al apartado de gastos.
    - Buscar el pedido y seleccionar el botón más.

    - Pulsar el botón eliminar.
    - Confirmar la eliminación.
- Secuencia alternativa:
    - Cancelar la eliminación.
    - Error al eliminar, aparece una notificación que lo indica.

Comprar producto

- Actores: Usuario.
- Objetivos: Comprar un producto de la lista de la compra.
- Precondiciones:
    - Haber iniciado sesión.
    - Pertenecer a un grupo.
    - Existencia de al menos un producto en la lista de la compra.
- Salidas:
    - Notificación que indica la compra del producto.
    - Desaparición del producto de la lista
- Postcondiciones:
    - Pedido marcado como comprado en la base de datos.
    - Creación de un nuevo gasto con los conceptos de la compra.
    - Actualización del balance
- Secuencia normal:
    - Acceder a gastos
    - Buscar el producto y pulsar comprar.
    - Escribir el precio final en el cuadro de dialogo
    - Confirmar compra.

- Secuencia alternativa:
    - Error compra, aparece notificación que lo indica.

Editar dato personal

- Actores: Usuario.
- Objetivos: Editar los datos personales.
- Precondiciones:
    - Haber iniciado sesión.
- Salidas:
    - Modificación del dato en la página.
- Postcondiciones:
    - Modificación de los datos de usuario en la base de datos.
- Secuencia normal:
    - Acceder al apartado perfil.
    - Buscar el dato a cambiar y pulsar el botón cambiar.
    - Modificar el dato deseado.
    - Pulsar el botón aceptar.
- Secuencia alternativa:
    - Error al modificarlo, aparece una notificación que lo indica.

Expulsar miembro

- Actores: Administrador.
- Objetivos: Expulsar a un miembro del grupo.
- Precondiciones:
    - Haber iniciado sesión.
    - Pertenecer a un grupo.

    - Existencia de más miembros.
- Salidas:
    - Eliminación del miembro de la lista.
- Postcondiciones:
    - Modificación del grupo del usuario expulsado a ningún grupo en la base de datos.
- Secuencia normal:
    - Acceder al apartado perfil.
    - Buscar el miembro que se quiere expulsar y pulsar el botón expulsar.
    - Confirmar la expulsión.
- Secuencia alternativa:
    - Error en la expulsión, aparece una notificación que lo indica.

Hacer administrador

- Actores: Administrador.
- Objetivos: Hacer administrador a un miembro del grupo.
- Precondiciones:
    - Haber iniciado sesión.
    - Pertenecer a un grupo.
    - Existencia de más miembros del grupo.
- Salidas:
    - Notificación que indica que el usuario es ahora administrador.
- Postcondiciones:
    - Modificación del usuario a administrador en la base de datos.
- Secuencia normal:

    - Acceder al apartado de perfil.
    - Buscar el miembro y pulsar el botón hacer admin.
    - Confirmar la acción.
- Secuencia alternativa:
    - Cancelar la acción.
    - Error al hacer administrador, aparece una notificación que lo indica.

Log out

- Actores: Usuario, administrador.
- Objetivos: cerrar sesión.
- Precondiciones:
    - Haber iniciado sesión.
- Salidas:
    - Volver a la página de inicio de sesión.
- Postcondiciones:
    - Se cierra la sesión.
- Secuencia normal:
    - Acceder al apartado de perfil.
    - Pulsar el botón cerrar sesión.
    - Confirmar la acción.
- Secuencia alternativa:
    - Cancelar la acción.

Eliminar cuenta

- Actores: Usuario, administrador.

- Objetivos: Eliminar la cuenta del usuario.
- Precondiciones:
    - Haber iniciado sesión.
- Salidas:
    - Vuelves a la página de inicio de sesión.
    - Notificación que indica que se eliminado correctamente.
- Postcondiciones:
    - Usuario eliminado de la base de datos.
    - Se cierra la sesión.
- Secuencia normal:
    - Acceder al apartado de perfil.
    - Pulsar el botón eliminar cuenta.
    - Confirmar la acción.
- Secuencia alternativa:
    - Cancelar la acción.
    - Error en la eliminación, aparece una notificación que lo indica.

Dejar grupo

- Actores: Usuario
- Objetivos: Dejar el grupo.
- Precondiciones:
    - Haber iniciado sesión.
    - Pertenecer a un grupo.
- Salidas:
    - Desaparece la información del grupo.

    - Notificación que indica que has dejado el grupo.
- Postcondiciones:
    - El grupo del usuario cambia a no grupo en la base de datos.
- Secuencia normal:
    - Acceder al apartado de perfil.
    - Pulsar el botón dejar grupo.
    - Confirmar la acción.
- Secuencia alternativa:
    - Cancelar la acción.
    - Error al dejar el grupo, aparece una notificación que lo indica.

Eliminar grupo

- Actores: Administrador.
- Objetivos: Eliminar el grupo.
- Precondiciones:
    - Haber iniciado sesión.
    - Pertenecer a un grupo.
- Salidas:
    - Desaparición de la información del grupo.
    - Notificación de eliminación de grupo con éxito.
- Postcondiciones:
    - Modificación de los miembros del grupo a no grupo.
    - Eliminación de las tareas, gastos y productos del grupo.
- Secuencia normal:
    - Acceder al apartado de perfil.

    - Pulsar el botón de eliminar grupo.
    - Confirmar la acción.
- Secuencia alternativa:
    - Cancelar la acción.
    - Error al eliminar el grupo, aparece una notificación que lo indica.

Apéndice D - Arquitectura de la base de datos

En este capítulo se desarrollarán los detalles sobre la estructura de la base de datos creada en este proyecto. Esta base de datos se ha creado haciendo uso de la aplicación xampp, que proporciona una interfaz gráfica muy simple con la que poder manejar todos los aspectos de la base de datos. Para la interacción entre la base de datos y la aplicación se decidió usar Sequelize, una librería que permite esta interacción sin hacer uso de consultas.

En Sequelize hubo que definir 8 modelos, cada modelo hace referencia a una tabla de la base de datos: user, group, chore, debtors, debts, expenses, notification, timetable. Entre estos modelos se establecen asociaciones que representan las relaciones existentes en la base de datos, y se aprecian en el siguiente diagrama:

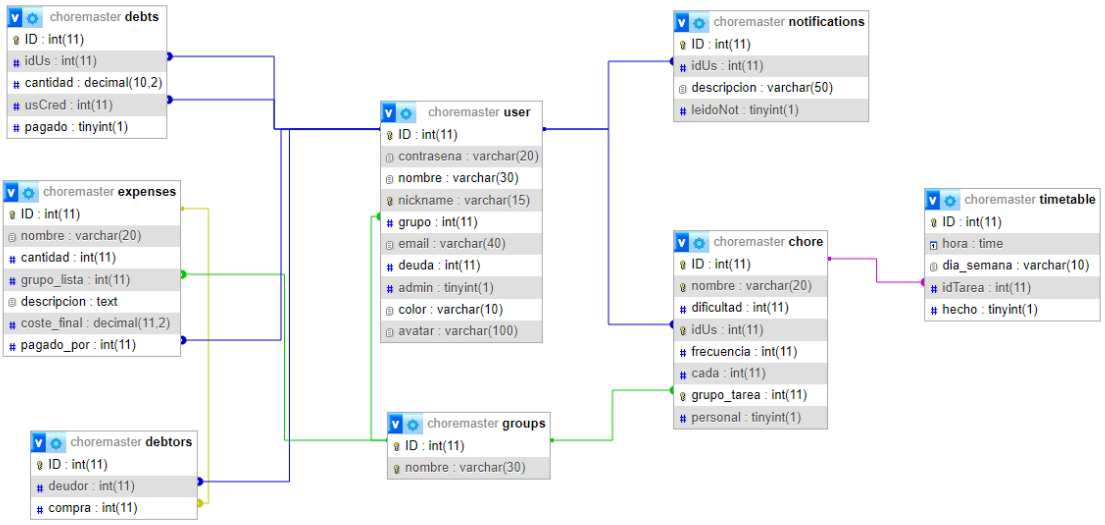


Ilustración 9-11 Diagrama Entidad-Relación de la base de datos

User

Este modelo representa a los usuarios guardando todos los datos sobre ellos.

Atributos

- ID: es la clave primaria de la tabla.
- Contraseña: campo de tipo varchar que guarda la contraseña.
- Nombre: campo de tipo varchar que guarda el nombre del usuario.
- Nickname: campo de tipo varchar que guarda el usuario para el inicio de sesión.
- Grupo: campo numérico que referencia el grupo al que pertenece el usuario. Este no podrá ser nulo, en caso de que el usuario no pertenezca a un grupo el valor será 0.
- Email: campo de tipo varchar que guarda el correo del usuario.
- Deuda: campo numérico que guarda el balance actual del usuario.
- Color: campo de tipo varchar que representa el color asociado al usuario.
- Avatar: campo de tipo varchar que guarda la ruta a la foto de perfil.

Asociaciones

El modelo users tiene 7 relaciones con otros modelos:

- OneToMany: debts, debtors, expenses, chores, notifications.
- ManyToOne:
    - Grupo → groups (id)

Groups

Este modelo representa a los grupos guardando todos los datos sobre ellos.

Atributos

- ID: es la clave primaria de la tabla. La clave preestablecida para no pertenecer a ningún grupo es 0.
- Nombre: campo varchar que guarda el nombre del grupo.

Asociaciones

El modelo users tiene 7 relaciones con otros modelos:

- OneToMany: user, expenses, chores.

Chore

Este modelo representa a las tareas que se pretenden organizar.

Atributos

- ID: es la clave primaria de la tabla.
- Nombre: campo de tipo varchar que guarda el nombre asignado a la tarea.
- Dificultad: campo numérico que asigna la dificultad de la tarea.
- IdUs: campo numérico que hace referencia a un usuario. Este será el usuario asignado para realizar la tarea. El campo no podrá ser nulo.
- Frecuencia: campo numérico que guarda la frecuencia de la tarea en el plazo especificado en el campo cada.
- Cada: campo numérico que guarda el plazo en el que se ha de repetir la tarea. Tiene tres valores posibles: 0 si la tarea se debe hacer semanalmente, 1 si la tarea se ha de hacer mensualmente, 2 si la tarea solo se debe hacer una vez.
- Grupo_tarea: campo numérico que hace referencia a un grupo en concreto. Este representa al grupo que pertenece la tarea.
- Personal: campo booleano que indica si la tarea es personal o grupal.

Asociaciones

El modelo chore tiene 7 relaciones con otros modelos:

- OneToMany: timetable.
- ManyToOne:
    - Grupo_tarea → groups (id)

- IdUs → user (id)

Timetable

Este modelo representa los horarios de las tareas.

Atributos

- ID: es la clave primaria de la tabla.
- Hora: campo de tipo "time" que almacena la hora a la que se debe realizar la tarea dada.
- Dia_semana: campo de tipo varchar que almacena el día de la semana, mes o día concreto que se ha de realizar la tarea. La comprobación de tipo se hará en el frontend.
- IdTarea: campo de tipo numérico que hace referencia a una tarea. Este representa la tarea a la que está asociada la hora expresada en los campos anteriores.
- Hecho: campo de tipo booleano que muestra si la tarea asociada se ha realizado o no.

Asociaciones

El modelo timetable tiene una única asociación:

- ManyToOne:
  - IdTarea → chore (id)

Notifications

Este modelo representa las notificaciones dentro de la aplicación.

Atributos

- ID: es la clave primaria de la tabla.

- IdUs: campo numérico que hace referencia a un usuario. Este representa el usuario que debe recibir dicha notificación.
- Description: campo de tipo varchar que guarda explica el motivo de la notificación.
- LeidoNot: campo de tipo booleano que indica si se ha leído la notificación o no.

Asociaciones

El modelo notification tiene una única relación:

- ManyToOne:
    - IdUs → user (id)

Expenses

Este modelo representa los gastos concretos de cada grupo, en este modelo se incluyen los productos de la lista de la compra.

Atributos

- ID: es la clave primaria de la tabla.
- Nombre: campo de tipo varchar que indica el concepto por el cual se ha generado el gasto.
- Cantidad: campo de tipo numérico que en el caso de los productos de la lista de la compra indica cuantos productos se han de comprar.
- Grupo_lista: campo de tipo numérico que hace referencia a un grupo. Este es el grupo al que pertenece el gasto dado.
- Description: campo de tipo varchar que muestra una breve descripción ya sea de lo que se espera comprar o del gasto añadido.
- Coste_final: campo de valor decimal que indica la cantidad exacta del gasto que hay que pagar.

- Pagado_por: campo de valor numérico que hace referencia a un usuario. Este indica el usuario que ha pagado el gasto, en caso de que nadie lo haya pagado aún será nulo.

Asociaciones

El modelo users tiene 4 relaciones con otros modelos:

- OneToMany: debts, debtors.
- ManyToOne:
  - Grupo_lista → groups (id)
  - Pagado_por → user (id)

Debts

Este modelo representa a las deudas entre los usuarios. Entre dos usuarios nunca podrá existir más de una deuda, debido a que en el caso de que hubiera una deuda existente y llegará una nueva deuda entre ambos usuarios la deuda inicial se actualizaría sumando o restando la nueva cantidad.

Atributos

- ID: es la clave primaria de la tabla.
- IdUs: campo de tipo numérico que hace referencia a un usuario. Este representa el usuario que debe pagar la deuda.
- Cantidad: campo de tipo decimal que representa la deuda que hay entre los usuarios representados por idus y usCred.
- UsCred: campo de tipo numérico que hace referencia a un usuario. Este representa el usuario al que le deben de pagar la deuda.
- Pagado: campo de tipo booleano que representa si la deuda esta pagada o no.

Asociaciones

El modelo debts tiene 2 relaciones con otros modelos:

- ManyToOne:
    - IdUs → user (id)
    - UsCred → user (id)

Debtors

Este modelo representa a los usuarios guardando todos los datos sobre ellos.

Atributos

- ID: es la clave primaria de la tabla.
- Compra: campo de tipo numérico que hace referencia a un gasto. Este gasto es el que ha de pagar el usuario que se hace referencia en el siguiente campo.
- Deudor: campo de tipo numérico que hace referencia a un usuario. Este es uno de los usuarios que han de pagar el gasto en cuestión.

Asociaciones

El modelo debtors tiene 2 relaciones con otros modelos:

- ManyToOne:
    - Compra → expenses (id)
    - Deudor → user (id)