ComuNest

Alumno: Cristian Porras Torres

Ciclo: Desarrollo de Aplicaciones Web

Memoria DAW

IES Abastos. Curso 2023/24. Grupo 7W. 24 de Mayo de 2024

Tutor: Jose María Carceller Pérez

ÍNDICE

[1-Introducción 1](#_Toc167200838)

[2-Contexto 1](#_Toc167200839)

[2.1-Identificación 1](#_Toc167200840)

[2.2-Justificación 1](#_Toc167200841)

[2.3-Objetivos del proyecto 1](#_Toc167200842)

[2.3.1-Objetivos principales 1](#_Toc167200843)

[2.3.2-Objetivos específicos 2](#_Toc167200844)

[2.3.3-Objetivos de ampliación 2](#_Toc167200845)

[3-Diseño 2](#_Toc167200846)

[3.1-Guía de estilos 2](#_Toc167200847)

[3.1.1-Paleta de colores 2](#_Toc167200848)

[3.1.2-Fuente y formatos 3](#_Toc167200849)

[3.2-Roles de usuario 4](#_Toc167200850)

[3.3-Base de datos 4](#_Toc167200851)

[3.3.1-Explicación BD 4](#_Toc167200852)

[3.3.2-Diseño Lógico 5](#_Toc167200853)

[3.3.3-Diseño entidad relación 6](#_Toc167200854)

[4-Tecnologías 6](#_Toc167200855)

[4.1-Front end 6](#_Toc167200856)

[4.1.1-Elección de framework 6](#_Toc167200857)

[4.1.2-Manejo de sesiones 6](#_Toc167200858)

[4.1.3-Enrutado 7](#_Toc167200859)

[4.1.4-Framework de CSS 7](#_Toc167200860)

[4.2-Back end 7](#_Toc167200861)

[4.2.1-Tecnología elegida 7](#_Toc167200862)

[4.2.2-Elección de framework 8](#_Toc167200863)

[4.2.3-Bcrypt 8](#_Toc167200864)

[4.2.4-mysql2 8](#_Toc167200865)

[4.2.4-Elección de gestor de base de datos 9](#_Toc167200866)

[4.3-Herramientas 9](#_Toc167200867)

[4.3.1-Visual Studio Code 9](#_Toc167200868)

[4.3.2-Node Package Manager 9](#_Toc167200869)

[4.3.3-Vue.js Devtools 10](#_Toc167200870)

[4.3.4-Thunderclient 10](#_Toc167200871)

[4.3.5-Git 10](#_Toc167200872)

[4.3.6-phpMyAdmin 11](#_Toc167200873)

[4.3.7-Docker Desktop 11](#_Toc167200874)

[4.3.8-Vue CLI 11](#_Toc167200875)

[5.Desarrollo 12](#_Toc167200876)

[5.1-Interfaz de Programación de Aplicaciones 12](#_Toc167200877)

[5.2-Estructura de los archivos 12](#_Toc167200878)

[5.2.1-Estructura servidor/cliente 12](#_Toc167200879)

[5.3-Desarrollo backend 13](#_Toc167200880)

[5.3.1-Creación de la aplicación en Express.js 13](#_Toc167200881)

[5.3.2-Estructura del controlador 14](#_Toc167200882)

[5.3.3-Conexión con la base de datos 15](#_Toc167200883)

[5.3.4-Rutas 16](#_Toc167200884)

[5.4-Desarrollo en frontend 17](#_Toc167200885)

[5.4.1-Creación de la app Vue 17](#_Toc167200886)

[5.4.2-Estructura de las vistas 18](#_Toc167200887)

[5.4.3-Rutas 20](#_Toc167200888)

[5.5-Control de versiones 21](#_Toc167200889)

[5.5.1-Git Flow 21](#_Toc167200890)

[5.5.2-Ramas 21](#_Toc167200891)

[5.6-Testing y depuración 22](#_Toc167200892)

[5.6.1-Cliente 22](#_Toc167200893)

[5.6.2-Servidor 22](#_Toc167200894)

[6-Temporalidad 22](#_Toc167200895)

[6.1-Tabla de tiempos 22](#_Toc167200896)

[6.2-Listado de commits en la rama main 23](#_Toc167200897)

[7-Despliegue 23](#_Toc167200898)

[7.1-Configuración inicial 23](#_Toc167200899)

[7.2-Instalación del Entorno de Desarrollo 24](#_Toc167200900)

[7.3-Despliegue de la API 24](#_Toc167200901)

[7.4-Configuración de la Base de Datos 24](#_Toc167200902)

[7.5-Despliegue del Cliente 24](#_Toc167200903)

[8-Conclusiones 24](#_Toc167200904)

[8.1-Conclusión sobre la elección de tecnologías 24](#_Toc167200905)

[8.2-Conclusiones sobre el proyecto 25](#_Toc167200906)

[9-Referencias 25](#_Toc167200907)

[10-Anexo 1](#_Toc167200908)

[10.1-Anexo A 1](#_Toc167200909)

# 1-Introducción

En respuesta a la necesidad de simplificar la gestión financiera de un grupo o familia, se ha concebido y desarrollado una plataforma web específica para la gestión de gastos. El objetivo principal de esta herramienta es simplificar y mejorar la forma en que los grupos y familias administran sus finanzas, facilitando la división de gastos y el registro de cada transacción mensual con facilidad.

# 2-Contexto

## 2.1-Identificación

El programa toma la forma de una plataforma web intuitiva y accesible diseñada para brindar soluciones efectivas a los desafíos financieros comunes que enfrentan amigos, compañeros de cuarto, familiares u otras entidades que comparten gastos. La herramienta proporciona una infraestructura digital donde los usuarios pueden administrar sus finanzas compartidas de manera eficiente y transparente, simplificando la tarea de dividir gastos y mantener registros detallados de las transacciones mensuales.

## 2.2-Justificación

La razón de esta medida surge de la clara necesidad en muchos entornos de optimizar la gestión de los gastos compartidos. La falta de un sistema organizado y centralizado a menudo conduce a confusión, malentendidos y conflictos relacionados con el dinero entre miembros del grupo o de la familia. La plataforma web tiene como objetivo abordar estos problemas proporcionando soluciones digitales para promover la transparencia, la colaboración y la equidad en la gestión financiera compartida.

## 2.3-Objetivos del proyecto

### 2.3.1-Objetivos principales

El objetivo principal de este proyecto consiste en la creación de un ambiente digital propicio para la gestión financiera compartida entre diversos grupos, familias, compañeros de vivienda, parejas y otros entornos afines. Se busca proporcionar a estos distintos colectivos una plataforma accesible y funcional, donde cada individuo tenga la capacidad de registrar detalladamente sus gastos de manera precisa. Este entorno, caracterizado por su interfaz diseñada de manera sencilla e intuitiva, tiene como propósito facilitar la tarea de registro de gastos, garantizando así una gestión financiera más eficiente y transparente para todos los involucrados.

### 2.3.2-Objetivos específicos

Otros objetivos específicos incluyen implementar nuevas funciones para mejorar la experiencia del usuario. Estos incluyen la capacidad de crear múltiples "casas" para administrar múltiples entornos simultáneamente, lo que permite una organización más eficiente de los gastos compartidos en diferentes entornos. Además, se presenta la posibilidad de marcar si un mes ha sido resuelto, brindando a los usuarios una herramienta para rastrear claramente su situación financiera para cada período. También integrado un sistema de búsqueda que facilita la localización de grupos y sus miembros. Esto permite a los usuarios acceder de forma rápida y eficiente a información relevante sobre los grupos creados y acceder a ellos si se requiere.

### 2.3.3-Objetivos de ampliación

Como objetivos de ampliación, se ha considerado agregar funciones adicionales para enriquecer la utilidad de la plataforma. Esto incluye implementar un sistema para administrar los gastos fijos, que son gastos que ocurren regularmente todos los meses. Esta función brindaría a los usuarios una manera conveniente de registrar y monitorear estos gastos predefinidos, simplificando así la gestión financiera diaria. Además, de integra el cálculo automático de la deuda total mensual y los sobrepagos entre los participantes. Este cálculo automatizado proporcionaría a los usuarios una visión clara de los saldos pendientes, lo que facilitaría una mejor comprensión y equilibrio de la asignación de cargos compartidos.

# 3-Diseño

## 3.1-Guía de estilos

### 3.1.1-Paleta de colores

Para la selección de la paleta de colores se han priorizado colores simples e intuitivos para el usuario. Colores claros que se puedan ver con cualquier tipo de iluminación y no haya cabida a la confusión del usuario. Para que el usuario sea consciente de que el mes ha sido resuelto se ha establecido el color verde con bastante transparencia para que no sea un color muy intrusivo. Así como su contraposición de un mes no resuelto, que en este caso es la tarjeta roja con un porcentaje muy alto de transparencia. La paleta de colores de la página, mayoritariamente blanco y gris muy oscuro ha sido seleccionada para destacar los botones o las acciones funcionales, ya que, la prioridad del proyecto es la utilidad.

![Forma

Descripción generada automáticamente]() Color seleccionado para el mes resuelto / #e5ffe5

![Forma

Descripción generada automáticamente con confianza media]()

Color seleccionado para el mes no resuelto / #ffe5e5

![Imagen que contiene Forma

Descripción generada automáticamente]()

Color de fondo / #ffffff



Color header y footer / #212529

![Forma, Cuadrado

Descripción generada automáticamente]()

Color botones login y signup / #ffc107



Color botones añadir gasto, mes, funcionalidad de volver atrás y botón de envío de formulario/ #0d6efd



Color de las tarjetas de casa en la búsqueda / #e0e0e0



Colores de los bordes de las tarjetas de los meses / #ff0000 y #00ff00

### 3.1.2-Fuente y formatos

Se eligió "Times New Roman" por su amplia disponibilidad y familiaridad para los usuarios. Es una de las fuentes predeterminadas en muchos navegadores web, lo que significa que los usuarios estarán familiarizados con su apariencia y, a menudo, la tendrán instalada en sus dispositivos. Esta sensación de familiaridad ayuda a aumentar la comodidad y satisfacción de los usuarios al interactuar con la plataforma porque se reducen las posibles barreras de entrada asociadas con la adaptación a una nueva fuente. Además, "Times New Roman" es conocido por su legibilidad y claridad, lo que lo hace adecuado para presentar información en un entorno digital. Su diseño clásico y atemporal garantiza una visualización clara del contenido, lo cual es especialmente importante para las plataformas web utilizadas para gestionar finanzas compartidas, donde la precisión y la comprensión de la información son cruciales. También el único formato que aplicamos es “bold” en la tabla de los gastos de cada mes, se ha seleccionado este formato en los títulos de la tabla para que destaquen y sea más intuitivo para el usuario el ver a que sección pertenece cada uno de los apartados.

## 3.2-Roles de usuario

Cuando un usuario accede a la aplicación sin estar registrado, se le asigna automáticamente el rol de "invitado". En este estado, las funcionalidades de la aplicación están limitadas y el usuario no podrá acceder a todas las características disponibles. Esto se implementa con el propósito de proteger la privacidad y seguridad de los datos, así como también para fomentar la participación plena dentro de la aplicación. Una vez que el usuario se registra y realiza el inicio de sesión, se concede acceso completo a todas las funcionalidades de la aplicación. Esto incluye la capacidad de realizar acciones como agregar, editar o eliminar datos. Con esta estrategia de asignar roles basados en el estado de autenticación del usuario se permite garantizar una experiencia de usuario coherente y segura. También si el usuario en cuestión es administrador podrá seleccionar cualquiera de las casas del buscador y eliminarlas para aseguras así el uso correcto de estas.

## 3.3-Base de datos

### 3.3.1-Explicación BD

Cuando el usuario se registra, se implementan sus datos en la tabla de usuario. Al crear un grupo, en este caso llamados casa, se añade una fila a la tabla casa y los usuarios que se unan o quien la haya creado se añadirán a la tabla “casa\_usuario” para poder así gestionar varios usuarios por casa o varias casas por usuario. En cada casa encontramos varios meses para tener el seguimiento de las finanzas y la gestión económica de la casa en cuestión, pudiendo así resolverlo o no cuando esté todo solventado. Así cada mes tiene su colección de gastos, donde encontramos al usuario que lo ha ejecutado, la casa en la que se ha producido y el mes al que pertenece.

### 3.3.2-Diseño Lógico

Tabla usuario:

id\_user (clave primaria)

nombre

email

pass

nivel

Tabla casa:

id\_casa (clave primaria)

pass

nombre

Tabla casa\_user:

id\_casa (clave externa referenciando id\_casa en la tabla casa)

id\_user (clave externa referenciando id\_user en la tabla usuario)

Tabla mes:

id\_mes (clave primaria)

nombre

id\_casa (clave externa referenciando id\_casa en la tabla casa)

resuelto

Tabla gasto:

id\_gasto (clave primaria)

nombre

descripcion

importe

id\_user (clave externa referenciando id\_user en la tabla usuario)

id\_casa (clave externa referenciando id\_casa en la tabla casa)

id\_mes (clave externa referenciando id\_mes en la tabla mes)

### 3.3.3-Diseño entidad relación

Diagrama

Descripción generada automáticamente

# 4-Tecnologías

## 4.1-Front end

### 4.1.1-Elección de framework

Vue 3 destaca frente a otros frameworks o JavaScript vanilla para proyectos pequeños por varias razones. En comparación con JavaScript vanilla, Vue 3 ofrece una estructura más organizada y una sintaxis más clara, lo que simplifica el desarrollo y el mantenimiento del código. Además, Vue 3 proporciona características avanzadas como la reactividad y la gestión del estado de manera más intuitiva, lo que reduce la complejidad y el tiempo de desarrollo. En comparación con otros frameworks, Vue 3 se destaca por su curva de aprendizaje más suave y su menor cantidad de configuración inicial, lo que lo hace ideal para proyectos pequeños donde se valora la agilidad y la eficiencia en el desarrollo. Además, la flexibilidad de Vue 3 permite a los desarrolladores integrar fácilmente bibliotecas y herramientas adicionales según sea necesario, adaptándose a los requisitos específicos del proyecto sin agregar una sobrecarga innecesaria.

### 4.1.2-Manejo de sesiones

“localStorage” se destaca como una herramienta eficiente para el manejo de sesiones en aplicaciones web. Al almacenar datos de manera persistente en el navegador del usuario, asegura la continuidad de la sesión incluso después de cerrar y abrir el navegador, proporcionando una experiencia de usuario sin interrupciones. Su implementación simple y directa agiliza el proceso de desarrollo, mientras que su seguridad integrada garantiza la protección de los datos del usuario. En resumen, “localStorage” ofrece una combinación única de persistencia, simplicidad y seguridad, lo que lo convierte en una elección confiable y efectiva para el manejo de sesiones en aplicaciones web.

### 4.1.3-Enrutado

Cuando se trata del enrutamiento en aplicaciones Vue 3, “vue-router” destaca como una herramienta esencial. Permite la navegación fluida entre vistas, garantizando una experiencia de usuario consistente. Su sintaxis declarativa simplifica la definición de rutas y componentes asociados, ideal para proyectos de gran escala. Además, “vue-router” ofrece características avanzadas, como rutas anidadas y protección de rutas, adaptándose a necesidades complejas. Su integración fluida con Vue.js y otras herramientas del ecosistema asegura una experiencia de desarrollo coherente.

### 4.1.4-Framework de CSS

Bootstrap destaca como un framework de CSS preferido para diversos proyectos debido a su amplia gama de componentes predefinidos y estilos personalizables, facilitando la creación eficiente de interfaces atractivas y receptivas. Su estructura CSS flexible y modular simplifica la organización del código, mientras que su sistema de cuadrícula receptiva garantiza una experiencia de usuario consistente en todos los dispositivos. En comparación con otros frameworks, Bootstrap ofrece una amplia adopción y soporte en la comunidad, así como plantillas y temas listos para usar, acelerando el proceso de desarrollo y asegurando un aspecto profesional y pulido para el proyecto.

## 4.2-Back end

### 4.2.1-Tecnología elegida

La elección de Node.js como tecnología de backend proporciona una serie de ventajas distintivas para el desarrollo de aplicaciones web. Al permitir el uso de JavaScript en ambos lados del stack, facilita la coherencia y la colaboración entre los equipos de desarrollo frontend y backend. Su modelo de Entrada/Salida basado en eventos lo hace especialmente adecuado para aplicaciones en tiempo real y de alta concurrencia, garantizando un rendimiento óptimo incluso bajo cargas de trabajo intensas. Además, el extenso ecosistema de módulos de Node.js, impulsado por npm, ofrece una amplia gama de herramientas y soluciones preexistentes que aceleran el proceso de desarrollo.

### 4.2.2-Elección de framework

Elegir Express.js sobre Node.js directamente para el desarrollo de aplicaciones web ofrece varias ventajas clave. Express.js simplifica el desarrollo con su facilidad de uso y una potente capacidad de middleware (software que se ejecuta entre la solicitud del cliente y la respuesta), que permite integrar funciones adicionales como autenticación y compresión de datos de manera eficiente. Su flexibilidad permite una adaptación fácil a las necesidades específicas del proyecto, respaldada por una gran comunidad y soporte activo. Además, Express.js garantiza un rendimiento óptimo, asegurando una ejecución rápida y eficiente incluso en entornos exigentes.

### 4.2.3-Bcrypt

Bcrypt es una librería utilizada para la encriptación de contraseñas en aplicaciones web, especialmente cuando se desarrolla una API con Express.js. Bcrypt implementa un algoritmo de hashing que convierte una contraseña en una cadena de caracteres irreconocibles, que es lo que se almacena en la base de datos en lugar de la contraseña en texto plano. Esta técnica de hashing es crucial para la seguridad, ya que, en caso de una brecha de datos, las contraseñas almacenadas no se verán comprometidas. La librería bcrypt ofrece varias ventajas: primero, genera hashes únicos incluso para contraseñas idénticas debido a la incorporación de un valor de “salt” aleatorio en el proceso de hashing, lo que dificulta los ataques de fuerza bruta. Segundo, bcrypt permite ajustar el coste computacional del hashing a través de un factor de trabajo, aumentando la dificultad para descifrar las contraseñas con el aumento de la potencia de computación.

### 4.2.4-mysql2

Es una librería para Node.js que permite la comunicación con bases de datos MySQL. Facilita la realización de consultas y transacciones de forma asincrónica, lo que mejora el rendimiento de las aplicaciones. Además, proporciona soporte para promesas de conexiones, lo que permite una gestión eficiente de múltiples conexiones a la base de datos. Su uso básico implica la creación de una conexión a la base de datos y la ejecución de consultas para obtener o modificar datos.

### 4.2.4-Elección de gestor de base de datos

Utilizar MySQL como motor de base de datos para una web pequeña ofrece varias ventajas clave. MySQL es conocido por su rendimiento, fiabilidad y escalabilidad, lo que garantiza un funcionamiento eficiente incluso con cargas de trabajo moderadas. Además, al ser de código abierto y contar con una amplia adopción en la comunidad, ofrece una gran cantidad de recursos y herramientas disponibles para facilitar el desarrollo y la administración de la base de datos. Su compatibilidad con diversas tecnologías y plataformas permite una integración sin problemas con otros componentes de la infraestructura de la aplicación.

## 4.3-Herramientas

### 4.3.1-Visual Studio Code

Visual Studio Code (VSCode) destaca como un editor de texto preferido por su combinación de características potentes, facilidad de uso y una amplia comunidad de usuarios y extensiones. Su interfaz intuitiva y altamente personalizable permite a los desarrolladores adaptar el entorno de trabajo a sus necesidades específicas, facilitando la productividad y la eficiencia en el desarrollo de software. Además, VSCode ofrece una integración sin problemas con una variedad de lenguajes de programación, herramientas de desarrollo y servicios en la nube, lo que lo convierte en una opción versátil para una amplia gama de proyectos y flujos de trabajo. Su modelo de extensibilidad permite a los usuarios ampliar y personalizar aún más sus capacidades a través de una amplia selección de extensiones disponibles en el mercado, que van desde mejoras en la productividad hasta integraciones con tecnologías específicas. En resumen, la combinación de características potentes, facilidad de uso y una comunidad activa.

### 4.3.2-Node Package Manager

Node Package Manager, o “npm”, es una herramienta esencial en el ecosistema de Node.js que se utiliza para gestionar las dependencias de un proyecto, así como para instalar, compartir y distribuir paquetes de código JavaScript. Con npm, los desarrolladores pueden fácilmente agregar, actualizar y eliminar dependencias de sus proyectos, lo que simplifica el proceso de desarrollo y asegura la consistencia en las versiones de las bibliotecas utilizadas. Además, npm proporciona acceso a un vasto repositorio de paquetes de código abierto, lo que permite a los desarrolladores aprovechar soluciones preexistentes y acelerar el desarrollo de sus aplicaciones.

### 4.3.3-Vue.js Devtools

La extensión para navegador Vue.js Devtools es esencial para desarrollar con Vue.js, ya que facilita el proceso de desarrollo y depuración. Permite inspeccionar y depurar componentes Vue en tiempo real, proporcionando una visión detallada del estado de la aplicación y la estructura del árbol de componentes. Además, facilita el seguimiento de mutaciones de estado, eventos y rendimiento, lo que ayuda a identificar y solucionar problemas de manera eficiente. Esta herramienta optimiza el desarrollo al proporcionar una experiencia más fluida y eficiente, mejorando la calidad y estabilidad del código, lo que se traduce en una mejor experiencia para los usuarios finales. En resumen, Vue.js Devtools es esencial para el desarrollo con Vue.js, proporcionando funcionalidades avanzadas que facilitan la depuración y optimización de aplicaciones Vue.

### 4.3.4-Thunderclient

Utilizar la extensión Thunderclient para hacer pruebas con la API en desarrollo ofrece ventajas significativas. Proporciona una interfaz intuitiva y fácil de usar para enviar y recibir solicitudes HTTP de manera eficiente, simplificando el proceso de desarrollo y prueba. Además, ofrece características avanzadas como guardar y compartir colecciones de solicitudes, gestionar variables de entorno y escribir scripts, permitiendo una mayor automatización y personalización en las pruebas. Esto facilita la identificación y solución de problemas antes de implementar la API en producción, mejorando la eficiencia y colaboración en el equipo de desarrollo.

### 4.3.5-Git

Utilizar Git como controlador de versiones ofrece numerosas ventajas frente a otras opciones disponibles. Git es ampliamente reconocido por su eficiencia, escalabilidad y robustez, lo que lo convierte en una opción popular para el control de versiones en proyectos de desarrollo de software. Su arquitectura distribuida permite trabajar de forma descentralizada, lo que resulta en una mayor flexibilidad y autonomía. Además, Git ofrece un historial completo de cambios en el código, lo que facilita la identificación de errores o problemas de manera eficiente. Su capacidad para realizar ramificaciones y fusiones de forma rápida y sencilla permite el desarrollo experimentar con nuevas características y realizar cambios en paralelo de manera segura. Además, Git proporciona una variedad de herramientas y comandos que facilitan la gestión y el seguimiento de las versiones del código, lo que garantiza la integridad y la consistencia del proyecto a lo largo del tiempo. En resumen, Git es una herramienta fundamental para el desarrollo de software, ofreciendo eficiencia y control de versiones en proyectos de cualquier tamaño y complejidad.

### 4.3.6-phpMyAdmin

“phpMyAdmin” es una herramienta de administración de bases de datos MySQL basada en web que ofrece una interfaz gráfica para gestionar bases de datos de manera eficiente. Permite a los usuarios realizar una amplia gama de tareas, como crear, eliminar y modificar bases de datos y tablas, ejecutar consultas SQL, importar y exportar datos, y gestionar usuarios y privilegios. Su popularidad radica en su facilidad de uso, su capacidad para trabajar con múltiples bases de datos y su amplia gama de funciones, lo que lo convierte en una opción preferida para administrar bases de datos MySQL, especialmente para usuarios que prefieren una interfaz gráfica en lugar de la línea de comandos.

### 4.3.7-Docker Desktop

Docker es una plataforma de código abierto que simplifica la creación, implementación y ejecución de aplicaciones dentro de contenedores. Estos contenedores ofrecen un entorno aislado y consistente para ejecutar aplicaciones, incluyendo bases de datos, con todas las dependencias necesarias. Lo que hace a Docker ideal para desplegar bases de datos es su portabilidad y consistencia; los contenedores funcionan de la misma manera en cualquier entorno donde se ejecuten, ya sea en desarrollo, pruebas o producción. Además, Docker simplifica la configuración de bases de datos al proporcionar imágenes preconfiguradas y listas para usar que se pueden implementar fácilmente con unos pocos comandos. También permite escalar las bases de datos de manera eficiente utilizando la misma imagen de contenedor en múltiples instancias. Al compartir recursos del sistema operativo subyacente, Docker garantiza una utilización eficiente de los recursos de hardware y una mayor densidad de implementación. Además, proporciona herramientas de gestión centralizadas que facilitan la monitorización del rendimiento, la gestión de la configuración y la implementación automatizada de bases de datos en entornos de producción.

### 4.3.8-Vue CLI

Vue CLI es una herramienta de línea de comandos que simplifica la creación y gestión de proyectos Vue.js. Proporciona una estructura de proyecto predefinida, permitiendo la creación rápida de aplicaciones Vue.js. Vue CLI facilita el desarrollo local con un servidor de desarrollo incorporado, y simplifica la compilación y empaquetado de la aplicación para su distribución en producción. Además, es altamente personalizable y extensible mediante plugins y configuraciones avanzadas, lo que permite adaptar el entorno de desarrollo a las necesidades específicas del proyecto. En resumen, Vue CLI agiliza el proceso de desarrollo de aplicaciones Vue.js al proporcionar una interfaz intuitiva y herramientas integradas para el desarrollo, compilación y gestión del proyecto.

# 5.Desarrollo

## 5.1-Interfaz de Programación de Aplicaciones

Utilizar una API, o Interfaz de Programación de Aplicaciones, ofrece numerosas ventajas en el desarrollo de software. Una API permite la comunicación entre diferentes componentes de software o sistemas, permitiendo que se intercambien datos y funcionalidades de manera segura y eficiente. Funciona como un conjunto de reglas y protocolos que define cómo interactuar con un sistema o servicio específico. Cuando un sistema hace una solicitud a una API, esta procesa la solicitud y devuelve una respuesta con los datos solicitados o realiza una acción específica. Esto permite a los desarrolladores acceder y utilizar funcionalidades complejas sin tener que conocer todos los detalles internos de cómo funciona el sistema subyacente. Además, las APIs facilitan la integración de servicios externos en una aplicación, lo que permite a los desarrolladores aprovechar características y datos de otras plataformas para enriquecer su propia aplicación. Por ejemplo, una aplicación de clima puede utilizar una API de servicios meteorológicos para obtener pronósticos actualizados.

## 5.2-Estructura de los archivos

### 5.2.1-Estructura servidor/cliente

La elección de la estructura de carpetas API/Cliente se fundamenta en una organización eficiente del código en proyectos web. Esta división clara entre el lado del servidor, donde se aloja la lógica de la API, y el lado del cliente, donde se desarrolla la interfaz de usuario, promueve una modularidad y reutilización del código que facilita la gestión y el mantenimiento a medida que el proyecto crece. La carpeta "api" alberga los archivos relacionados con la API, mientras que la carpeta "client" contiene los elementos del cliente, permitiendo una colaboración efectiva entre equipos y una arquitectura limpia y escalable que se adapta a las necesidades en evolución del proyecto.

Texto

Descripción generada automáticamenteEscala de tiempo

Descripción generada automáticamente con confianza baja 5.2.2-Carpetas servidor 5.2.3-Carpetas Cliente

## 5.3-Desarrollo backend

### 5.3.1-Creación de la aplicación en Express.js

Texto

Descripción generada automáticamente

Este fragmento de código configura una aplicación Express para manejar solicitudes HTTP, incluyendo análisis de cuerpos JSON, manejo de CORS, y enrutamiento de solicitudes. También sirve archivos estáticos y maneja solicitudes no encontradas (en este caso, redirigiendo a una página de error 404), proporcionando una base sólida para el desarrollo de aplicaciones web con Express.js. También el módulo path simplifica la manipulación de rutas de archivos y directorios, proporcionando utilidades para construir y manipular rutas de manera consistente entre diferentes sistemas operativos.

### 5.3.2-Estructura del controlador

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

* Recepción de la Solicitud: Cuando el cliente envía una solicitud HTTP para eliminar una casa (probablemente a través de una solicitud DELETE o POST), el controlador borrarCasa se activa para manejar esta solicitud.
* Extracción y Validación de Datos: El controlador extrae el id\_casa del cuerpo de la solicitud (req.body). Este paso es crucial para identificar qué casa necesita ser eliminada. En una implementación más robusta, también se podría incluir validación para asegurar que id\_casa es válido y está presente.
* Interacción con la Base de Datos: Utilizando la función db.query, el controlador ejecuta una consulta SQL para eliminar la entrada correspondiente de la base de datos. La consulta utiliza una sintaxis parametrizada (DELETE FROM casa WHERE id\_casa = ?) para prevenir inyecciones SQL, lo que añade una capa de seguridad.
* Manejo de la Respuesta de la Base de Datos: La función de callback en db.query maneja la respuesta de la base de datos. Si ocurre un error (err), se envía una respuesta JSON con { exito: 0 } y un estado HTTP 200. Si la operación es exitosa, se envía una respuesta JSON con { exito: 1 } y un estado HTTP 200. Este manejo binario del éxito o fracaso de la operación proporciona retroalimentación inmediata al cliente.
* Manejo de Excepciones: El bloque try...catch envuelve toda la lógica para capturar cualquier error inesperado que pueda surgir fuera del contexto de la consulta de la base de datos. Si ocurre una excepción, se registra el error en la consola y se envía una respuesta JSON con un mensaje de error y un estado HTTP 500, indicando un error interno del servidor.

### 5.3.3-Conexión con la base de datos

Texto

Descripción generada automáticamente

* Importación del Módulo MySQL: Primero, se importa el módulo mysql2, que es una librería para interactuar con bases de datos MySQL en Node.js.
* Creación de la Conexión a la Base de Datos:Se utiliza mysql.createConnection para crear una conexión a la base de datos. Aquí, se proporcionan los detalles de conexión como host, puerto, usuario, contraseña y el nombre de la base de datos.
* Establecimiento de la Conexión: La función db.connect se llama para establecer la conexión con la base de datos. Si hay un error durante este proceso, se lanza una excepción con el mensaje de error. Si la conexión es exitosa, se imprime "Base de datos conectada" en la consola.
* Exportación del Objeto de Conexión: Finalmente, el objeto de conexión db se exporta para que pueda ser utilizado en otros módulos de la aplicación, incluyendo los controladores.

### 5.3.4-Rutas

Texto

Descripción generada automáticamente

* Importación de Express y Creación de un Router: Se importa el módulo express y se utiliza su método Router() para crear un nuevo router.
* Importación del Controlador: Se importa el módulo del controlador (index.controller) que contiene las funciones que manejan las diferentes rutas.
* Asociación de Rutas con Funciones del Controlador: Se definen las rutas y se asocian con las funciones del controlador correspondientes utilizando métodos como router.get() y router.post(). Por ejemplo, router.get('/', controller.index) significa que cuando se haga una solicitud GET a la ruta raíz '/', se ejecutará la función index del controlador.
* Exportación del Router: Se exporta el router para que pueda ser utilizado por la aplicación principal de Express.

En resumen, este archivo de enrutador organiza las rutas de la aplicación y las relaciona con las funciones del controlador adecuadas, lo que facilita el manejo de las solicitudes HTTP entrantes y el enrutamiento adecuado hacia las funciones correspondientes del controlador.

## 5.4-Desarrollo en frontend

### 5.4.1-Creación de la app Vue

Texto

Descripción generada automáticamente

Aquí se adjunta un fragmento del html publico en el vue y a raíz de este div se monta la aplicación en vue.

Texto

Descripción generada automáticamente

En este fragmento de código, se compone la aplicación Vue instanciando la aplicación con createApp(App), utilizando el enrutador Vue para la navegación con .use(router), y montando la aplicación en el DOM con .mount('#app'). Además, se importan el componente principal de la aplicación (App.vue), el enrutador Vue (router), y las librerías necesarias para estilos y funcionalidades adicionales de Bootstrap. Este enfoque sigue el patrón de diseño de Vue 3 para organizar y configurar la aplicación de manera eficiente, estableciendo el punto de entrada en el DOM y gestionando la navegación entre las diferentes vistas de la aplicación.

### 5.4.2-Estructura de las vistas

En Vue 3, las vistas están compuestas por tres partes principales: el template, el script y los estilos.

Pantalla de computadora

Descripción generada automáticamente con confianza media

* Template: En el template se define la estructura y el contenido HTML de la vista. Aquí se utilizan directivas como v-model para establecer la vinculación bidireccional entre los datos y la interfaz de usuario, y @click para manejar eventos de clic. También se pueden utilizar expresiones y directivas condicionales como v-if para controlar la visibilidad de elementos en función de condiciones específicas.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

* Script (Variante Setup): El script define la lógica y el comportamiento de la vista. En la variante setup, se utiliza la sintaxis simplificada para definir variables reactivas utilizando la función ref y para importar funcionalidades de Vue como useRouter para el enrutamiento. Aquí se define la lógica para el manejo de eventos, como el envío de datos al servidor y la gestión de respuestas. Además, se pueden utilizar condicionales y bucles para controlar el flujo de la aplicación.

Texto

Descripción generada automáticamente

* Estilos (Scoped): Los estilos definen la apariencia y el diseño de la vista. En este caso, los estilos están marcados como scoped, lo que significa que solo se aplicarán a los elementos dentro del componente actual y no se filtrarán a otros componentes. Aquí se pueden definir reglas de estilo CSS, como colores, márgenes y tamaños de fuente, que se aplicarán únicamente a los elementos dentro del componente.

### 5.4.3-Rutas

En Vue 3, el sistema de rutas se compone de tres partes principales: la configuración de rutas, la definición de las vistas y la creación del router.

Texto

Descripción generada automáticamente

* Configuración de Rutas: En la configuración de rutas, se define un array de objetos que representan las rutas de la aplicación. Cada objeto contiene propiedades como path, que especifica la URL de la ruta, name, que define un identificador único para la ruta, y component, que indica el componente Vue asociado a la ruta.

Texto

Descripción generada automáticamente

* Definición de Vistas: Las vistas son componentes Vue que representan las diferentes páginas de la aplicación. Cada ruta está asociada a un componente Vue específico, que se importa desde su archivo correspondiente. Estos componentes definen la estructura y el contenido HTML de cada página, así como la lógica y el comportamiento asociado.

Texto

Descripción generada automáticamente

* Creación del Router: El router se crea utilizando la función createRouter, que toma como argumentos un objeto de opciones de configuración y el array de rutas definido anteriormente. En este caso, se utiliza createWebHistory para indicar que se utilizará el modo de historial basado en la API de historial del navegador. Finalmente, se exporta el router para que pueda ser utilizado en la aplicación principal.

## 5.5-Control de versiones

### 5.5.1-Git Flow

Git Flow facilita la gestión de versiones, lo que es crucial para mantener un proyecto web estable y funcional. Al definir un proceso claro para la publicación de versiones estables, se puede garantizar que el código base esté siempre en un estado coherente y listo para su despliegue. Esto da la confianza de que se puedan lanzar nuevas versiones del proyecto de manera eficiente y sin problemas. Otra ventaja del Git Flow es que permite experimentar con nuevas funcionalidades y corregir errores sin arriesgar la estabilidad del código. Al trabajar en ramas separadas, se pueden probar nuevas ideas y realizar cambios sin preocupación por romper la aplicación principal.

### 5.5.2-Ramas

Texto

Descripción generada automáticamente

## 5.6-Testing y depuración

### 5.6.1-Cliente

En cuanto a la gestión, Vue Devtools permite inspeccionar y gestionar el estado de los componentes Vue en tiempo real. Esto incluye la visualización de la estructura del árbol de componentes, la inspección de propiedades y datos, y la edición en vivo de estado y props. Esto facilita la identificación y resolución de problemas relacionados con el estado de los componentes. Para el testing, Vue Devtools ofrece herramientas para realizar pruebas interactivas y depuración en tiempo real. Por ejemplo, se pueden realizar cambios en los datos de un componente y observar cómo afectan al estado y a la vista en tiempo real. Además, Vue Devtools proporciona una consola integrada que muestra mensajes de error y advertencias, lo que facilita la depuración de problemas de rendimiento o errores de código.

### 5.6.2-Servidor

Al utilizar el debugger de Node.js junto con Thunderclient para depurar y hacer testing de una API de Express, se puede examinar el estado interno y el flujo de ejecución del código desarrollado en tiempo real. El debugger de Node.js te permite colocar puntos de interrupción en tu código para detener la ejecución en puntos específicos y examinar el estado de las variables en ese momento. Por otro lado, Thunderclient te permite enviar solicitudes HTTP a los endpoints de la API y observar cómo se manejan esas solicitudes en tiempo real, lo que te permite probar y verificar el comportamiento de la API en diferentes escenarios de uso.

# 6-Temporalidad

## 6.1-Tabla de tiempos

|  |  |  |
| --- | --- | --- |
| **TAREA** | **TIEMPO ESTIMADO** | **TIEMPO REAL** |
| Investigar tecnologías | 2semanas | 2semanas |
| Importar dependencias de los frameworks y estructura de carpetas | 5 h | 4 h |
| Diseño y creación BD  + despliegue BD | 6 h | 4 h |
| Pantalla invitados | 2 días | 2 días |
| Pantalla home | 1 semana | 1 semana |
| Vista de Casa | 1 semana | 4 días |
| Vista de Gastos | 1 semana | 1 semana |
| Búsqueda | 1 día | 2 días |
| Login + Registro  +Login Casa | 1 día | 3 días |
| Añadir mes +  Añadir gasto | 1día | 5 h |
| Memoria | 2 meses | 2 meses |
| Despliegue | 1 día | 1 día |
| Crear presentación | 4 h | 2.5 h |

## 6.2-Listado de commits en la rama main

Se adjunta, en el Anexo A\*, imágenes de los últimos commits de cada rama al fusionarlas con main. Disponen de las fechas en las que se realizaron las fusiones a main.

# 7-Despliegue

## 7.1-Configuración inicial

Primero, se provisiona una máquina virtual en Oracle Cloud con una imagen de Ubuntu. Se configura el acceso mediante claves SSH para asegurar la conexión a la VM y se abren los puertos necesarios para el funcionamiento del servidor web y de la base de datos, específicamente los puertos 22 para SSH, 80 para HTTP, 443 para HTTPS y 3306 para MySQL.

## 7.2-Instalación del Entorno de Desarrollo

Se actualiza el sistema operativo de la VM y se procede a instalar Node.js y npm para gestionar las dependencias del proyecto. También MySQL como sistema de gestión de bases de datos y phpMyAdmin como interfaz de administración de MySQL.

## 7.3-Despliegue de la API

Se clona el repositorio del proyecto desde GitHub a la máquina virtual. Una vez en el servidor, las dependencias del proyecto se instalan usando npm. Se configura el entorno para que la API de Express pueda conectarse a la base de datos MySQL. Esto incluye la configuración de las variables de entorno necesarias, como las credenciales de la base de datos y otros parámetros de configuración.

## 7.4-Configuración de la Base de Datos

Se crea las bases de datos y las tablas necesarias en MySQL. Se implementan los datos iniciales mediante archivos de volcado SQL. Se continúa configurando phpMyAdmin para facilitar la administración de la base de datos.

## 7.5-Despliegue del Cliente

Se clonan los archivos del repositorio. Estos archivos se copiaron al directorio correspondiente en la máquina virtual. Se configura el servidor Apache para servir estos archivos estáticos y se maneja el enrutamiento hacia la API.

# 8-Conclusiones

## 8.1-Conclusión sobre la elección de tecnologías

Con respecto a la elección de tecnologías en el proyecto, se ha priorizado un modelo de desarrollo más moderno y seguro, optando por una arquitectura API/Cliente. Este enfoque permite una separación clara entre el backend y el frontend, lo que mejora la mantenibilidad y escalabilidad del proyecto. Para el backend, se ha elegido Express.js debido a su flexibilidad, rendimiento óptimo y la amplia comunidad de desarrolladores que proporciona soporte y recursos. Express facilita la creación de APIs robustas y eficientes, lo cual es esencial para la comunicación entre el servidor y el cliente. En cuanto al frontend, se ha optado por Vue 3, un framework progresivo de JavaScript que destaca por su simplicidad, reactividad y capacidad para construir interfaces de usuario dinámicas y eficientes. Vue 3 introduce mejoras significativas en rendimiento y en la estructura de código con Composition API, haciendo el desarrollo más intuitivo y potente. La combinación de Express y Vue 3 permite construir aplicaciones web modernas, seguras y de alto rendimiento, asegurando una experiencia de usuario fluida y una gestión eficaz del flujo de datos entre el cliente y el servidor.

## 8.2-Conclusiones sobre el proyecto

En general, el progreso del proyecto ha sido satisfactorio. La experiencia adquirida durante el desarrollo ha permitido optimizar considerablemente los tiempos y procesos. Aunque la incorporación de nuevas tecnologías al principio supuso un desafío, con el avance del proyecto se ha vuelto más fluido y eficiente. Si se tuviera que iniciar el proyecto nuevamente, el proceso sería considerablemente más ágil, ya que se contaría con un conocimiento más profundo de las herramientas y tecnologías empleadas. En resumen, a pesar de los desafíos iniciales, ha habido una evolución positiva a lo largo del desarrollo del proyecto.

# 9-Referencias

* Bootstrap. (n.d.). Bootstrap documentation. Recuperado de <https://getbootstrap.com/>
* Vue.js. (n.d.). Vue.js documentation. Recuperado de <https://vuejs.org/>
* Node.js. (n.d.). Node.js documentation. Recuperado de <https://nodejs.org/en>
* Express. (n.d.). Express.js documentation. Recuperado de <https://expressjs.com/es/>
* CJ Developer. (n.d.). Gitflow tutorial. Recuperado de <https://github.com/cjadeveloper/gitflow-tutorial>
* Thunder Client. (n.d.). Thunder Client documentation. Recuperado de <https://www.thunderclient.com/>
* Oracle. (n.d.). Documentación de Oracle. Recuperado de <https://www.oracle.com/es/>
* Apache Software Foundation. (s. f.). Documentación de Apache. Recuperado de <https://httpd.apache.org/>

# 10-Anexo

## 10.1-Anexo A

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Captura de pantalla de un videojuego

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza mediaCaptura de pantalla de computadora

Descripción generada automáticamente