PiggyBank – App de gestión y administración de ingresos y gastos

Memoria de Proyecto Final de Máster

Máster Universitario en Desarrollo de Sitios y Aplicaciones Web

Informática, Multimedia y Telecomunicación

Autor: Omar Adolfo Álvarez Hernández

Consultor: Carlos Caballero González

Profesor: César Pablo Córcoles Briongos y Julià Minguillón Alfonso

08 de junio de 2020

Créditos/Copyright

Dedicatoria/Cita

“He fallado más de 9000 tiros en mi carrera. He perdido casi 3000 juegos. 26 veces han confiado en mí para tomar el tiro que ganaba el juego y lo he fallado. He fracasado una y otra vez en mi vida y eso es por lo que tengo éxito”

Michael Jordan

Abstract

Este proyecto consiste en la creación de una aplicación web para administrar y gestionar los ingresos y gastos propios de cada usuario. En dicha aplicación, los usuarios pueden añadir todo tipo de gastos y ganancias que reciben, y a cambio la aplicación le proporciona recordatorios de pagos, estadísticas y la visualización de los ahorros acumulados y el dinero disponible para afrontar cada mes. PiggyBank ayuda al usuario a poder ahorrar, ya que le permite establecer metas de ahorro y gestiona la distribución de la economía de cada mes junto al aporte de consejos y buenas prácticas para poder obtener beneficios.

Esta aplicación cuenta con un diseño responsivo y adaptable a cualquier tamaño de pantalla permitiendo acceder desde cualquier dispositivo.

La aplicación se ha implementado utilizando Angular como framework JavaScript para el front-end y Laravel como framework PHP para el back-end.

Palabras claves: ingresos, pagos, ganancias, ahorros, gastos, Laravel, Angular

Abstract (english version)

This project consists of the creation of a web application to administer and manage the income and expenses of each user. In this application, users can add all kinds of expenses and earnings they receive, and in return the application provides payment reminders, statistics and the visualization of accumulated savings and money available to face each month. PiggyBank helps users to save by allowing them to set savings goals and manage the distribution of each month's savings along with advice and best practices to help them make a profit.

This application has a responsive design and can be adapted to any screen size allowing access from any device.

The application has been implemented using Angular as a JavaScript framework for the front-end and Laravel as a PHP framework for the back-end.

Keywords: income, payments, earnings, savings, expenses, Laravel, Angular

Agradecimientos, Notaciones y Convenciones

Agradecer a mis padres, a mi hermano, a mi abuela y al resto de mi familia por siempre ayudarme en todo y aportarme confianza y ánimo para lograr mis objetivos. Además, agradecerles mucho su paciencia y generosidad ya que sin ellos esto no hubiera sido posible.

Agradecer a mis amigos y a mi pareja por siempre animarme y ayudarme, ya que de esta forma han posibilitado siempre que tenga la fuerza suficiente para alcanzar mis metas.

Índice

Figuras y tablas

Índice de figuras

Índice de tablas

1. Introducción/Prefacio

1.1 Contexto

El Banco de España demuestra que la tasa de ahorro en nuestro país es una de las más bajas dentro de la zona euro no alcanzando el 5% de la renta bruta, por lo que los españoles no están preparados para enfrentar problemas económicos que puedan ocurrir de forma eventual, lo que provoca que tendamos a endeudarnos en el futuro con el fin de contrarrestar estos problemas de ahorro.

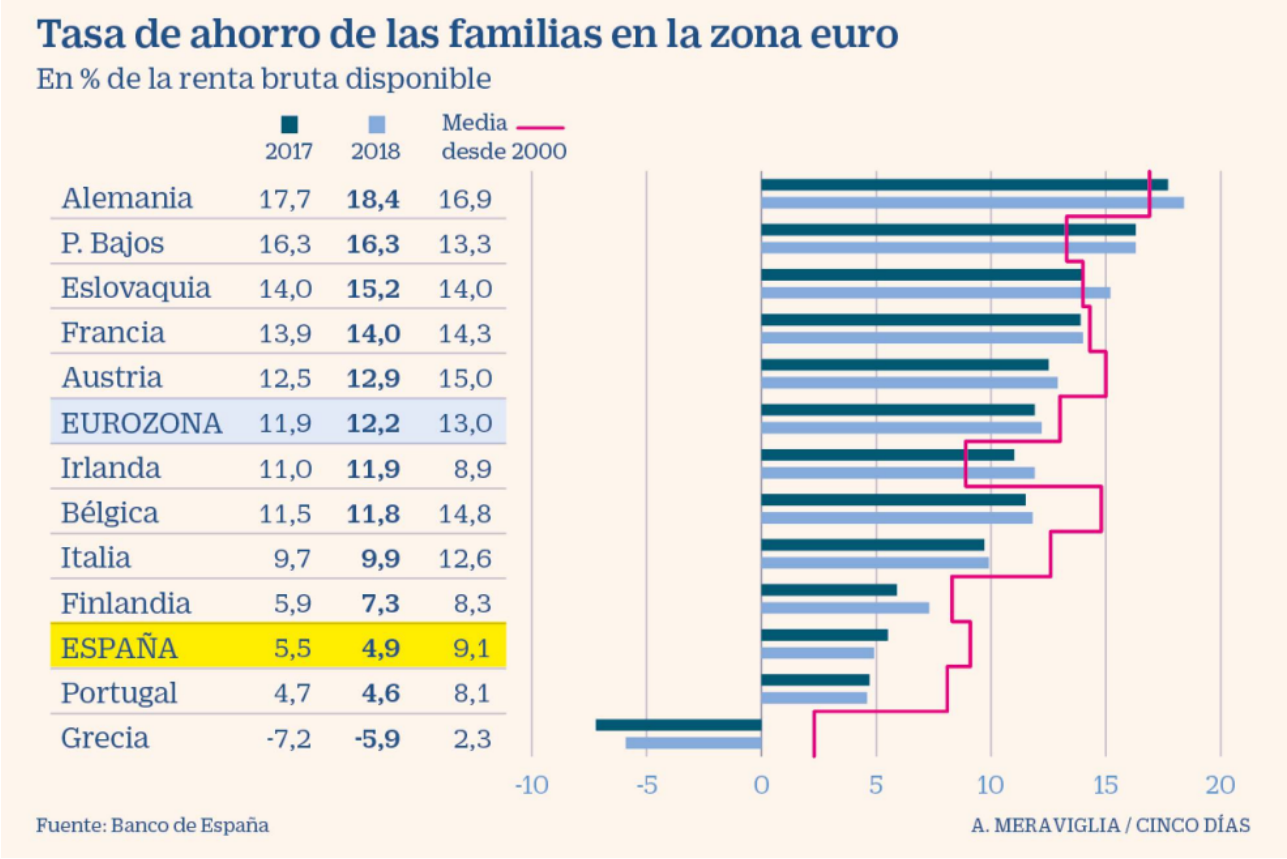


Ilustración 1: Tasa de ahorros de las familias de la zona euro

1.2 Justificación

Con el objetivo de solucionar estos problemas nace el desarrollo de este proyecto cuyo objetivo es ayudar al usuario a poder tener un mayor control sobre los gastos que tiene y a su vez, recomendarle los ahorros que debería efectuar mensualmente con el fin de alcanzar metas que se proponga cada año.

Aunque ya existen aplicaciones que tienen como fin permitir al usuario controlar sus gastos y ahorrar dinero, esta aplicación tiene como diferencia con respecto a otras, evitar un control sobre las propias cuentas bancarias del usuario que, muchas veces llevan a la desconfianza de introducir datos tan personales en cuentas de terceros, y sustituir dicho mecanismo por un control más manual e interactivo,

donde sea el usuario quien gestione los gastos que ya conoce y poder marcarlos como ya pagados, además de poder establecer metas de ahorro y ver cómo se desarrolla su economía a lo largo de los meses.

2. Descripción/Definición/Hipótesis

En este Trabajo Final de Máster se propone el desarrollo de una aplicación web que permita gestionar y administrar los ingresos y gastos con el fin de poder alcanzar unos ahorros mensuales que cumplan las expectativas de ahorro anual de un determinado usuario. Dicha aplicación está destinada para cualquier persona que mensualmente reciba unos ingresos y tenga que cumplir con una serie de gastos, por lo tanto, hablamos de un rango de amplitud en cuanto a usuarios muy grande.

Las funcionalidades principales de la aplicación son:

- Administrar los gastos e ingresos mensuales de forma personal aplicando el método de ahorro 50/30/20 (porcentaje de gastos primarios / gastos secundarios / ahorros) como uno de los métodos para ahorrar más importantes y utilizados en la actualidad.
- Seguimiento de los gastos y recibos pagados cada mes.
- Conocer el dinero disponible que tiene el usuario para ajustarse a los ahorros que desea realizar en el mes actual.
- Historial de los gastos realizados.
- Seguimiento de metas de ahorros propuestas.
- Recibir consejos de como se está realizando la administración del dinero.
- Recordatorio de pagos.

Con todo lo comentado, la aplicación tiene como fin primordial que el usuario en todo momento pueda administrar su economía de forma que siempre tenga controlado todos los beneficios y pérdidas que reciben sus cuentas bancarias, así como todos los gastos que realiza a lo largo de su vida económica teniendo la aplicación como fuente histórica de los mismos.

3. Objetivos

A continuación, se presentan los objetivos principales que se pretenden cumplir durante el desarrollo del proyecto, y los objetivos secundarios que buscan reforzar y mejorar el núcleo de la aplicación que tiene como objetivo principal aportar al usuario una herramienta que les ayude a ahorrar.

3.1 Principales

Los objetivos principales son:

- Informar al usuario de los ahorros acumulados.
- Informar al usuario del dinero disponible para el mes en que se encuentre.
- Permitir introducir gastos y ganancias.
- Permitir establecer metas de ahorros para incitar al usuario a ahorrar.
- Gestionar los ingresos y pagos de cada mes aplicando el método 50/30/20.
- Mantener un histórico de los gastos y ganancias.
- Registrar aquellos pagos que han sido efectuados.
- Documentar el desarrollo de la aplicación.
- Realizar una aplicación adaptable a cualquier dispositivo para poder realizar gestiones en cualquier momento y lugar.

3.2 Secundarios

Los objetivos secundarios son:

- Listar a modo de recordatorio los gastos pendientes del mes en el que se encuentre.
- Ofrecer estadísticas sobre: la evolución de los gastos y ganancias del año actual, total de gastos y ganancias, y gastos donde hemos empleado mayor cantidad de dinero.
- Ofrecer consejos y buenas prácticas que ayudan a llevar una mejor administración del dinero de cada mes.
- Establecer un sistema de autenticación basado en roles: usuario y administrador.

4. Contenidos

La aplicación cuenta con una página de bienvenida desde donde podemos acceder al inicio de sesión de esta. En ella, los usuarios se autentican en el sistema ya sea con el rol de usuario o con el de administrador. En caso de que el usuario no esté autenticado en la aplicación, puede acceder a la pantalla de registro donde será registrado con el rol de usuario ya que, sólo los administradores pueden crear una vez hayan iniciado sesión, otros usuarios administradores.

En la aplicación es necesario comentar una serie de detalles para tener en cuenta, con el fin de comprender mejor la aplicación:

- Tanto los gastos como las ganancias pueden estar englobados en diferentes tipos que permiten a la aplicación aplicar diferentes funcionalidades con ellos. En un primer momento, los tipos de gastos que se han establecido son: anuales, mensuales primarios, mensuales secundarios e importantes. De esta forma, la aplicación puede realizar diferentes funcionalidades con los tipos establecidos como son: recordatorios de gastos mensuales que aún no se han pagado, recordatorios anuales para recordar gastos que se han realizado el año anterior, metas de ahorros donde no se incorporan gastos importantes, ya que se establece que son gastos con una suma alta de ahorros y con un fin como puede ser comprar una casa y por ello, no deben influir en los objetivos de ahorros de un determinado año, y por último, las plantillas que tienen como fin administrar los gastos e ingresos de un determinado mes utilizan los gastos tanto mensuales primarios como mensuales secundarios para poder aplicar el método 50/30/20 con ellos. Por otro lado, los tipos de ganancias que se han establecido en un primer momento son: mensuales y extras. Las ganancias mensuales se utilizan para la gestión de los ingresos mensuales a través de las plantillas, y las ganancias extras no se utilizan con ninguna funcionalidad específica, sino que se incorporan para poder acoger a cualquier ganancia no mensual. Cada tipo está acompañado de una descripción que permite al usuario saber su finalidad.

El contenido de cada una de las páginas web es el siguiente:

- No autenticados
    - Página de bienvenida de la aplicación.
    - Inicio de sesión
    - Registro de usuarios
    - Olvido de contraseña
- Autenticado como usuario
    - Escritorio: donde podemos encontrar el dinero disponible para el mes actual, el total de ahorros acumulados, el porcentaje de gastos pagados de tipo mensual, recordatorios anuales de gastos que se han realizado con un año de diferencia y recordatorios mensuales de los gastos de tipo mensual que aún no se han pagado.

  - Gastos: donde el usuario agrega los gastos que tiene junto a los pagos de cada uno de ellos.
  - Ganancias: donde el usuario agrega las ganancias que tiene junto a los ingresos de cada una de ellas.
  - Objetivos: donde el usuario puede establecer metas de ahorros que desea cumplir para un determinado año.
  - Plantillas: donde el usuario agrega gastos y ganancias para un determinado mes de un año.
  - Estadísticas: donde se muestra los ingresos y pagos del año actual, total de ganancias y pagos, y los gastos donde invierte más dinero.
  - Ayuda: donde se adjunta el manual de usuario de la aplicación.
- Autenticado como administrador
  - Usuarios: listado de los usuarios registrados en la aplicación.
  - Tipos de gastos: listado de tipos de gastos que podemos seleccionar dentro del panel de usuarios a la hora de crear un gasto.
  - Tipos de ganancias: listado de tipos de ganancias que podemos seleccionar dentro del panel de usuarios a la hora de crear una ganancia.
  - Buenas prácticas: listado de buenas prácticas que se aplican dentro de las plantillas de los usuarios.
  - Ayuda: donde se adjunta el manual de administrador de la aplicación.

5. Metodología

La metodología elegida para llevar a cabo la aplicación ha sido el desarrollo en cascada debido que, cuenta con una serie de etapas bien diferenciadas y donde cada una de ellas contempla un paso vital dentro del desarrollo, con el fin de conseguir como producto final una aplicación web que cumpla los objetivos que hemos establecido. Las etapas de este modelo como vemos en la ilustración 2 son:

1. Análisis de requisitos: se establecen los requisitos y objetivos del presente proyecto junto a la planificación de este.
2. Diseño: en esta fase se eligen las tecnologías y herramientas que se utilizan para el desarrollo de la aplicación junto a los prototipos de las pantallas que permitan ver una imagen inicial de cómo va a ser la aplicación.
3. Implementación: en esta fase se comienza a desarrollar mediante código todas las funcionalidades que hemos descrito.
4. Verificación: se realizan pruebas en la aplicación con el fin de verificar que la aplicación funciona como hemos planteado desde un principio.
5. Mantenimiento: esta fase se reserva después de la finalización de este trabajo y se llevaría a cabo en el supuesto de que la aplicación se decidiese publicar en la web y que cualquier usuario la desease utilizar.

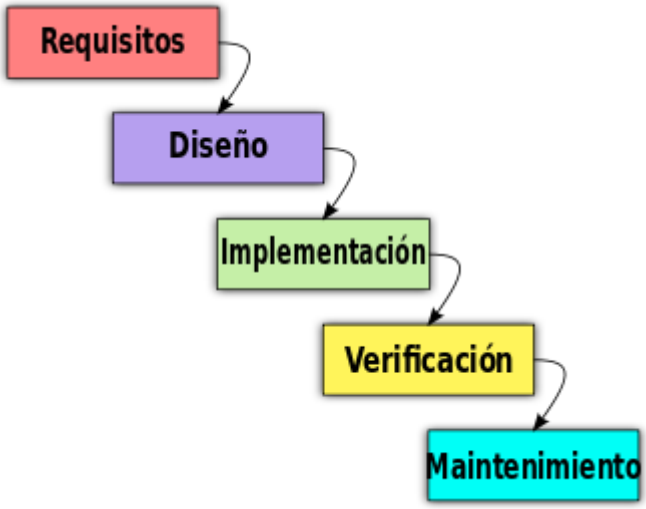


Ilustración 2: Modelo en cascada

6. Arquitectura de la aplicación/sistema/servicio

La arquitectura de la aplicación consta de las siguientes partes:

- Cliente: implementado con el framework JavaScript de Angular. Desde Angular se hacen las llamadas a la API REST implementada en el siguiente punto. Con este framework conseguimos una aplicación web SPA, es decir, un sitio web de una sola página cuyo objetivo es ofrecer a los usuarios una experiencia más fluida.
- Servidor: implementado con el framework PHP de Laravel. Gracias a Laravel, el cliente puede conectarse a la API RESTful que desarrollaremos con este framework y que, a su vez, éste se conectará con la base de datos.
- MySQL es el sistema de gestión de bases de datos utilizado.

La conexión cliente-servidor se realiza mediante llamadas HTTP securizadas mediante JWT desde Angular a la API Rest desarrollada en Laravel. La tecnología JWT permite añadir tokens que permitan autenticar al usuario contra la API Rest dentro de las cabeceras de las llamadas, de forma que, sólo se pueda utilizar la API si los usuarios están registrados dentro del sistema.

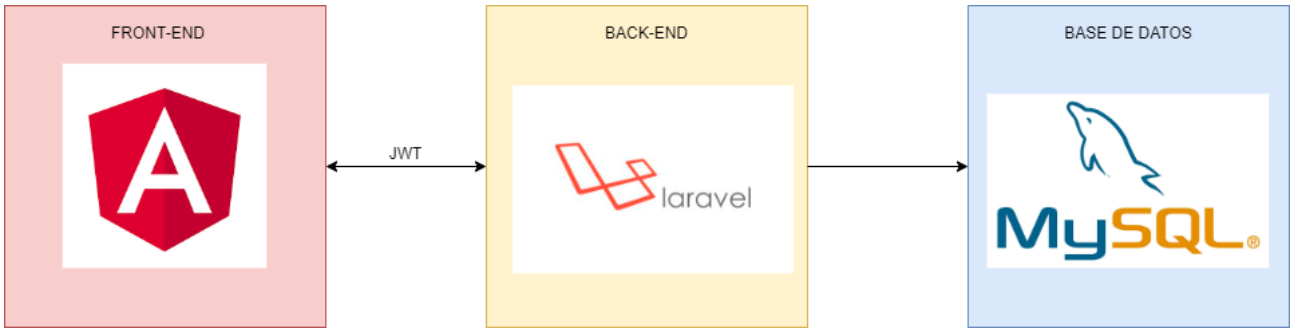


Ilustración 3: Arquitectura de la aplicación

7. Plataforma de desarrollo

Los recursos tecnológicos utilizados son los siguientes:

	SOFTWARE
Memoria del proyecto	Microsoft Word
Planificación	Gantt Project
Diagramas	Draw.io
Casos de uso	Creately
IDE	Visual Code
Gestión de control de versiones	Sourcetree
Servicio en la nube	Google Drive
Imágenes sin derechos de autor	Pixabay
Prototipos a bajo nivel	Balsamiq
Llamadas http	Postman
Vídeos	OBS Studio
Gestión de tareas	Trello
SFTP	FileZilla

Tabla 1: Recursos tecnológicos de software

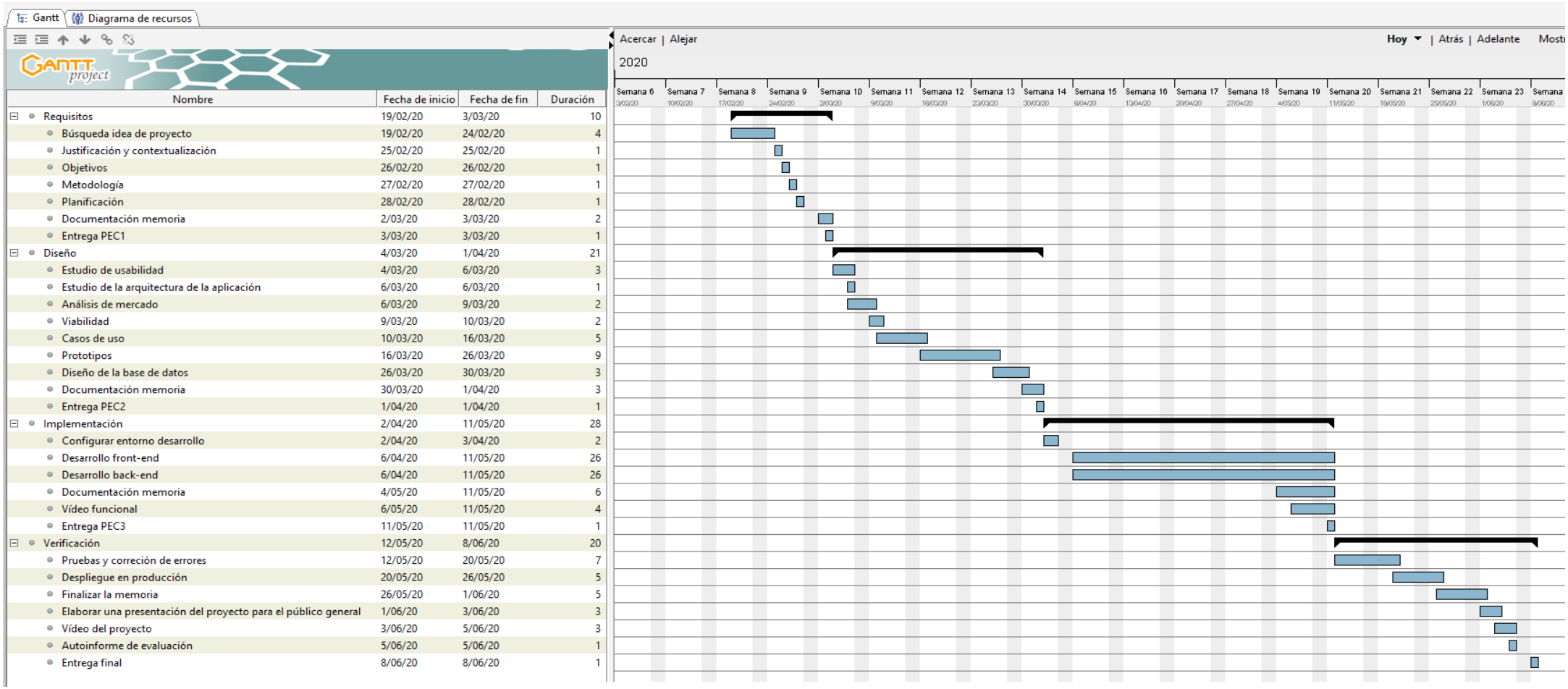
	APLICACIÓN
Repositorio de control de versiones	Github
Servidor de aplicaciones web en desarrollo	XAMPP

Tabla 2: Recursos tecnológicos de la aplicación

	HARDWARE
Ordenador	Intel i5 7500 3.4 GHZ, 8GB de RAM, 1 TB HDD
Sistema operativo	Windows 10

Tabla 3: Recursos tecnológicos de hardware

8. Planificación



Nombre	Fecha de inicio	Fecha de fin	Duración
Requisitos	19/02/20	3/03/20	10
Búsqueda idea de proyecto	19/02/20	24/02/20	4
Justificación y contextualización	25/02/20	25/02/20	1
Objetivos	26/02/20	26/02/20	1
Metodología	27/02/20	27/02/20	1
Planificación	28/02/20	28/02/20	1
Documentación memoria	2/03/20	3/03/20	2
Entrega PEC1	3/03/20	3/03/20	1
Diseño	4/03/20	1/04/20	21
Estudio de usabilidad	4/03/20	6/03/20	3
Estudio de la arquitectura de la aplicación	6/03/20	6/03/20	1
Análisis de mercado	6/03/20	9/03/20	2
Viabilidad	9/03/20	10/03/20	2
Casos de uso	10/03/20	16/03/20	5
Prototipos	16/03/20	26/03/20	9
Diseño de la base de datos	26/03/20	30/03/20	3
Documentación memoria	30/03/20	1/04/20	3
Entrega PEC2	1/04/20	1/04/20	1
Implementación	2/04/20	11/05/20	28
Configurar entorno desarrollo	2/04/20	3/04/20	2
Desarrollo front-end	6/04/20	11/05/20	26
Desarrollo back-end	6/04/20	11/05/20	26
Documentación memoria	4/05/20	11/05/20	6
Vídeo funcional	6/05/20	11/05/20	4
Entrega PEC3	11/05/20	11/05/20	1
Verificación	12/05/20	8/06/20	20
Pruebas y correción de errores	12/05/20	20/05/20	7
Despliegue en producción	20/05/20	26/05/20	5
Finalizar la memoria	26/05/20	1/06/20	5
Elaborar una presentación del proyecto para el público general	1/06/20	3/06/20	3
Vídeo del proyecto	3/06/20	5/06/20	3
Autoinforme de evaluación	5/06/20	5/06/20	1
Entrega final	8/06/20	8/06/20	1

Ilustración 4: Diagrama de Gantt

9. Proceso de trabajo/desarrollo

El proceso de desarrollo ha sido el siguiente:

- Creación de la base de datos: la base de datos fue creada dentro de phpMyAdmin con el nombre de piggy-bank. Una vez creada, se utilizó Laravel para la creación de las tablas y relaciones basándonos en el diagrama de entidad-relación que se define en el capítulo 10. Este framework proporciona una herramienta muy útil para el propósito mencionado llamado migraciones, donde establecemos por cada tabla, un fichero donde definimos el tipo de cada columna, la clave primaria y las claves foráneas que tiene con otras tablas. Con las migraciones creadas y con los datos de conexión a la base de datos establecidos de forma correcta dentro del archivo .env de Laravel, podemos ejecutar el comando php artisan migrate para poder crear las tablas dentro de nuestra base de datos.

- Creación de la API con Laravel: gracias a Laravel podemos definir la API de nuestra aplicación. Para ello, creamos por cada conjunto de funcionalidades de nuestra aplicación, un fichero controlador donde definimos los servicios web que éste va a contener. Cada servicio web puede ejecutar un método REST según su propósito: obtenemos datos mediante GET, creamos objetos mediante POST, modificamos objeto mediante PUT y eliminamos objetos mediante DELETE. Una vez hayamos creado nuestros controladores, debemos establecer las rutas dentro de nuestro fichero web.php para que el cliente pueda realizar sus peticiones consultando las rutas que establecemos en este archivo. Las rutas están agrupadas por roles de forma que cada rol puede consultar sólo las rutas a las que tiene permiso. El encargado de comprobar que cada rol consulte las rutas correspondientes es el middleware de jwt que se encarga de validar si el usuario tiene un token válido y el rol correspondiente para acceder a una determinada ruta.

- Creación del cliente con Angular. Dentro del cliente tenemos dos carpetas diferenciadas:
    - Shared: contiene los elementos compartidos de la aplicación. En ella tenemos los siguientes elementos:
        - Componente home que contiene el menú superior llamado navbar y un menú lateral llamado sidebar. En este componente se establece todas las opciones de los dos menús y a las rutas que accedemos una vez pulsemos alguna de las opciones.
        - Componente loader que se encarga de mostrar información útil al usuario de que hay una acción, en nuestro caso las llamadas al servidor, que se está ejecutando y debe esperar.

- Directiva decimal que se utiliza en los inputs de los formularios donde debemos añadir una cantidad y se encarga de asegurarse de que el formato es el que conocemos de euros y céntimos separados por un punto.
- Los guards de la aplicación se encargan de no permitirnos acceder a rutas de la aplicación donde el rol permitido no sea el nuestro además de validar el token para comprobar que estamos autenticados.
- Interceptor que se encarga de añadir el token jwt a la cabecera de cada llamada a la api debido que es necesario en aquellos casos que las rutas estén securizadas.
- Los modelos utilizados.
- Servicios
- Validadores de campos de formulario

- Vistas: contiene todas los componentes que conforman las vistas de la aplicación.

Por último, en un primer momento se comenzó a desarrollar el back-end y el front-end por separado y posteriormente una vez finalizados, se decidió añadir el proyecto de Angular dentro de la carpeta resources de Laravel y a la hora de compilar el código, copiar los archivos compilados de la carpeta dist de Angular dentro de la carpeta public de Laravel. De esta forma, tenemos todo el proyecto junto y podemos acceder al cliente a través de la raíz del proyecto de Laravel. Para finalizar, si queremos acceder al proyecto de Angular, debemos ir a la carpeta frontend que se encuentra dentro de la carpeta resources.

```
Route::get('/', function () {
    return View::make('index');
});
```

Ilustración 5: Ruta que renderiza el index de Angular

10. Diagramas UML

10.1 Diagrama E/R

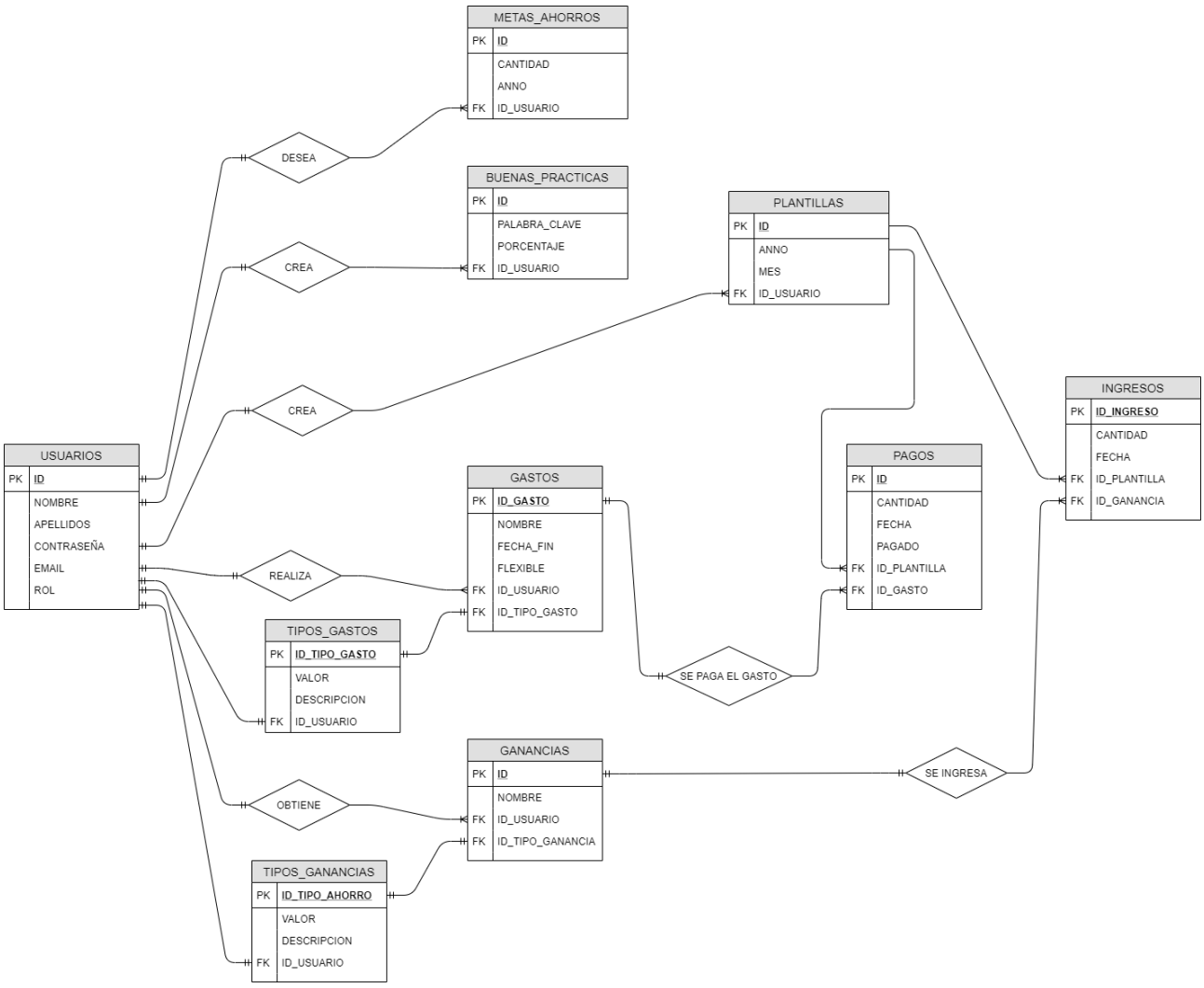


Ilustración 6: Diagrama E/R

Dentro del modelo anterior hay algunos detalles a comentar:

- Los tipos de ganancias y gastos tienen como clave foránea el id de usuario de forma que sólo aquellos usuarios administradores que hayan creado un determinado tipo sean los únicos que pueden editarlo o eliminarlo. El resto de los administradores pueden ver todos los tipos de la aplicación, pero de esta forma sólo pueden modificar los suyos. Esto se ha hecho como medida de seguridad para evitar que otros administradores eliminen tipos que se usan para determinadas funcionalidades de la aplicación.

- Tanto los ingresos como los pagos tienen como clave foránea el id de las plantillas, ya que dentro de la pantalla de detalle de las plantillas podemos añadir registros de las dos entidades, puesto que las propias plantillas están constituidas tanto por gastos como por ganancias del mes en cuestión.
- Tanto los gastos como las ganancias tienen un id correspondiente al tipo al que pertenecen debido que podemos diferenciarlos en diferentes tipos.
- El campo flexible dentro de la tabla de gastos se utiliza para que los usuarios puedan marcar aquellos gastos que se mantienen a lo largo de un mes creciendo como puede ser la gasolina o la compra, y de esta forma, no se incluye dicho gasto marcado como flexible dentro de los recordatorios de pagos mensuales.
- La estructura que se ha decidido utilizar permite evitar la repetición de campos y una mayor capacidad de trato con los datos.

10.2 Casos de usos

Se han diseñado los casos de usos de dos de las acciones más importantes del sistema:

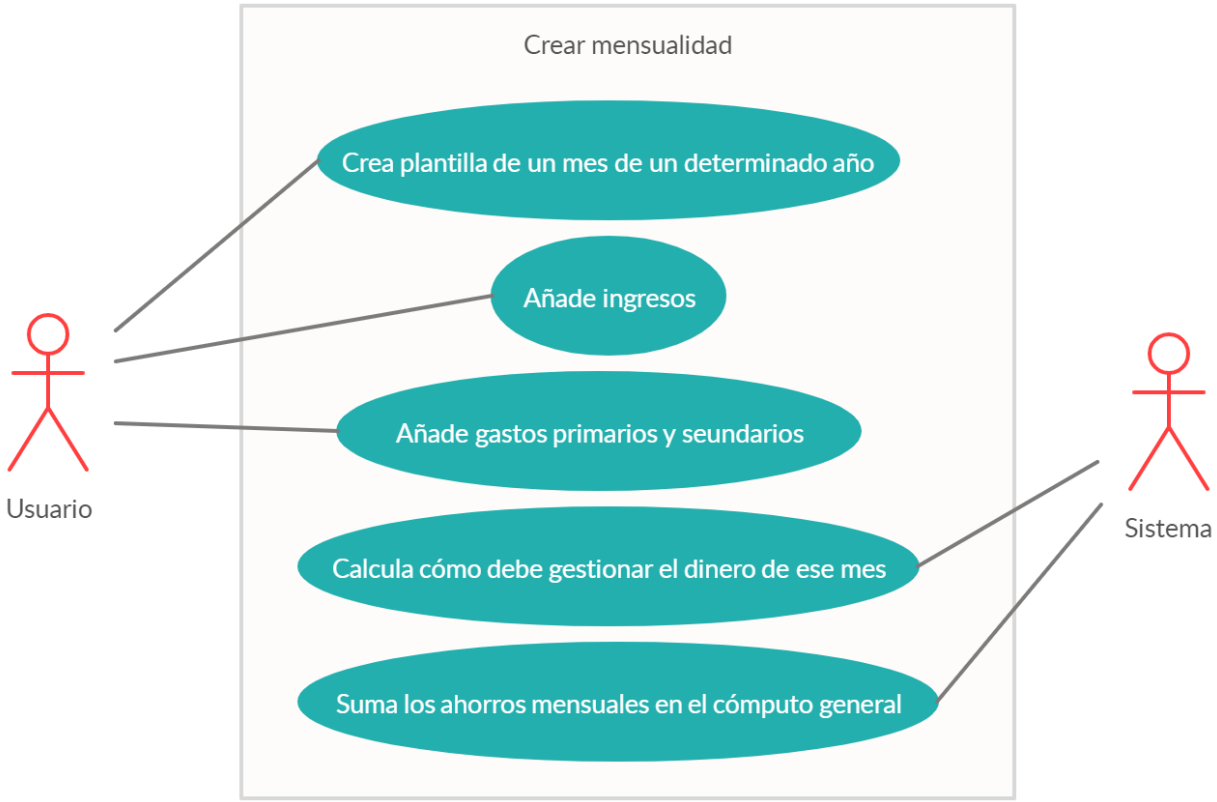


Ilustración 7: Caso de uso crear mensualidad

Caso de uso: Crear mensualidad

Descripción

El usuario crea el listado de gastos para un determinado mes de un año y el sistema le calcula como debe gestionar la economía de ese mes.

Actores

- Usuario
- BBDD

Precondiciones

- El usuario ha iniciado sesión en la aplicación.
- El usuario ha accedido a la pantalla.

Postcondiciones

- El usuario dentro del escritorio puede ver el dinero disponible de ese mes.
- El usuario dentro del escritorio puede ver el total de ahorros de gastos mensuales.

Flujo del proceso

1. El usuario accede a la pantalla de gastos mensuales.
2. El usuario crea una plantilla para un determinado mes de un año.
3. El usuario añade los ingresos de ese mes.
4. El usuario añade los gastos diferenciando entre primarios y secundarios.
5. El sistema calcula aplicando el método 50/30/20 la gestión de ese mes.
6. El sistema suma los ahorros del usuario al cómputo general de ahorros.

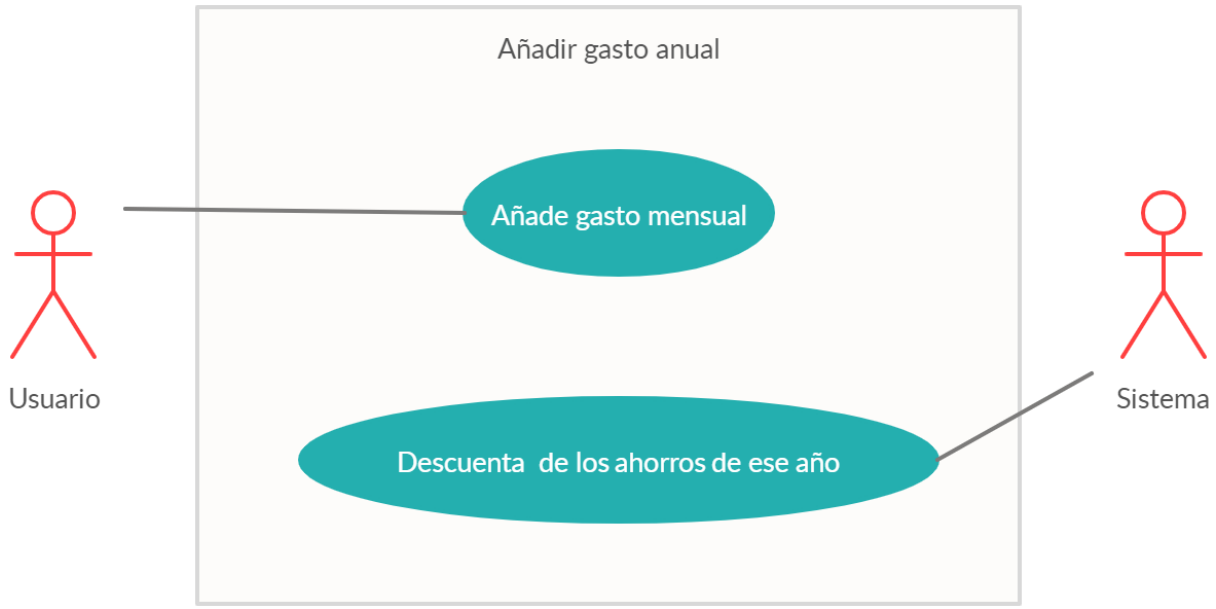


Ilustración 8: Caso uso añadir gasto anual

Caso de uso: Añadir gasto anual

Descripción

El usuario añade un gasto anual.

Actores

- Usuario
- BBDD

Precondiciones

- El usuario ha iniciado sesión en la aplicación.
- El usuario ha accedido a la pantalla de gastos anuales.

Postcondiciones

- El cómputo general de ahorros de ese año se reduce con el gasto anual que se haya introducido.

Flujo del proceso

1. El usuario accede a la pantalla de gastos anuales.
2. El usuario añade un gasto anual introduciendo el nombre, valor, año, mes y si desea que se le recuerde el siguiente año.
3. El sistema descuenta el valor del ahorro anual.

11. Prototipos

La aplicación se va a enfocar desde el punto de vista del Responsive Web Design, y por ello, debemos establecer los puntos de ruptura. Con los tres que trabajaremos son:

- Escritorio: mayor o igual de 1024px
- Tablet: 768px
- Móvil: 425px

11.1 Lo-Fi

Se han creados wireframes comunes que se repiten entre las diferentes pantallas de la aplicación con el fin de evitar crear un wireframe por cada pantalla de la aplicación y conseguir tener prototipos que muchas pantallas compartirán. Los wireframes creados son:

- Pantalla home

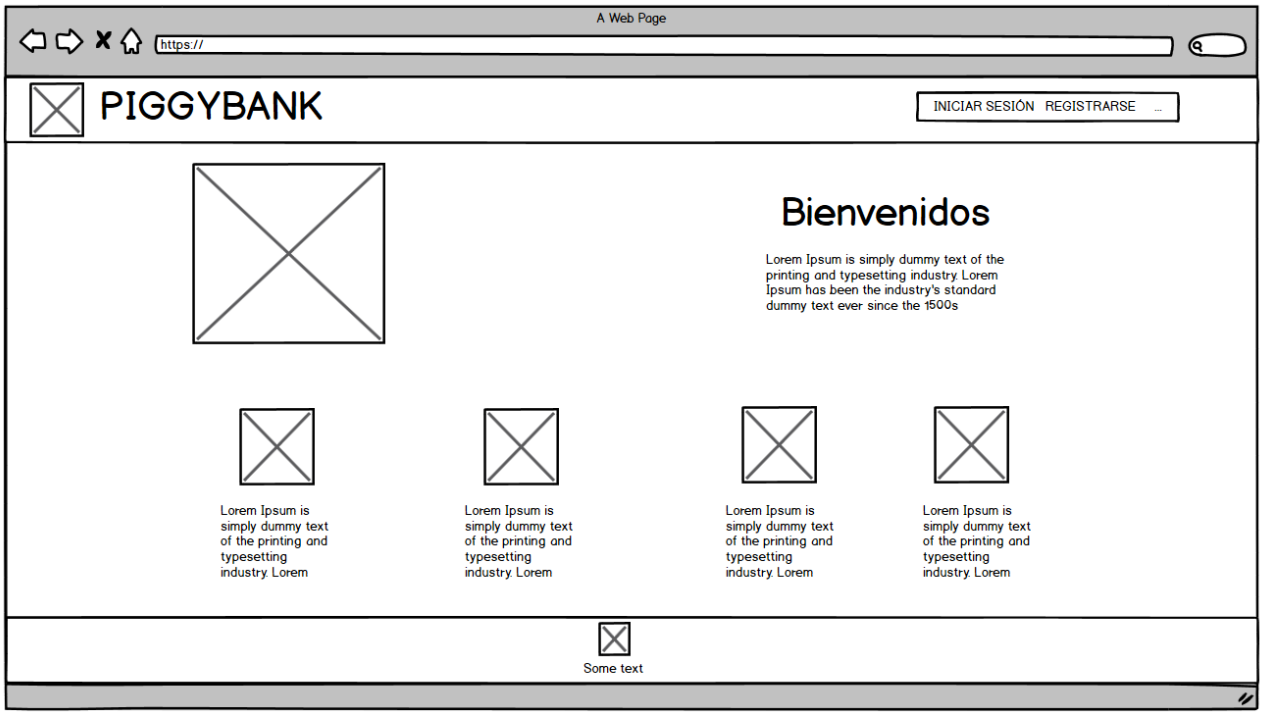


Ilustración 9: Wireframe home escritorio

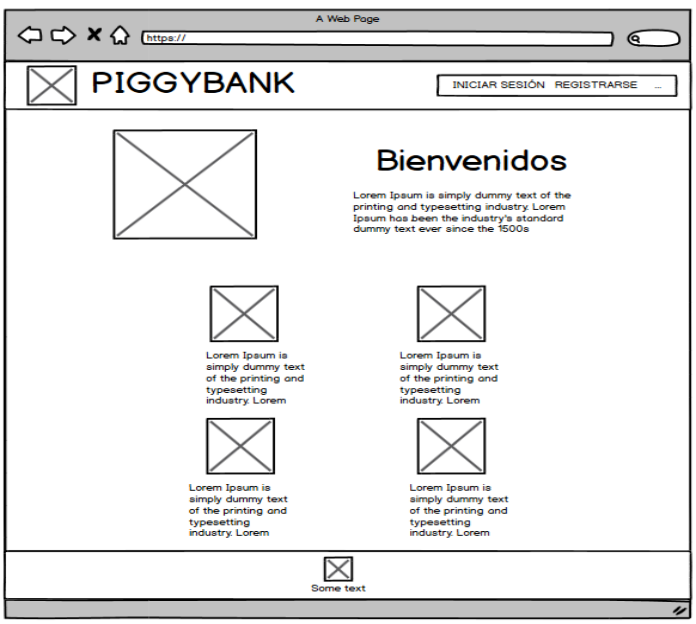


Ilustración 10: Wireframe home tablet

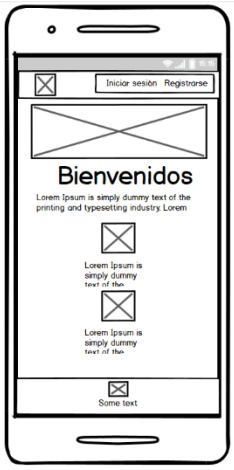


Ilustración 11: Wireframe home móvil

- Pantalla inicio. Reutilizable para las pantallas de registrar usuario y ha olvidado su contraseña, cambiando sus respectivos textos y campos.

Ilustración 12: Wireframe inicio de sesión escritorio

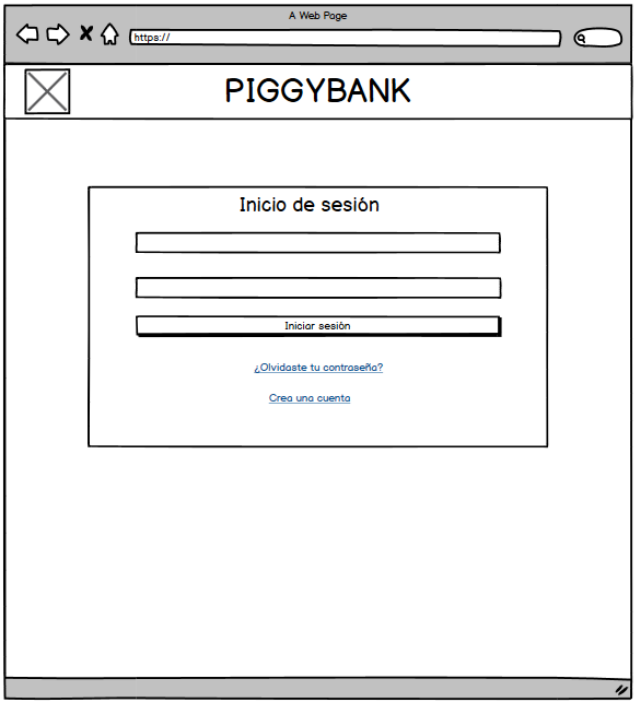


Ilustración 13: Wireframe inicio tablet

Ilustración 14: Wireframe inicio móvil

- Pantalla de escritorio

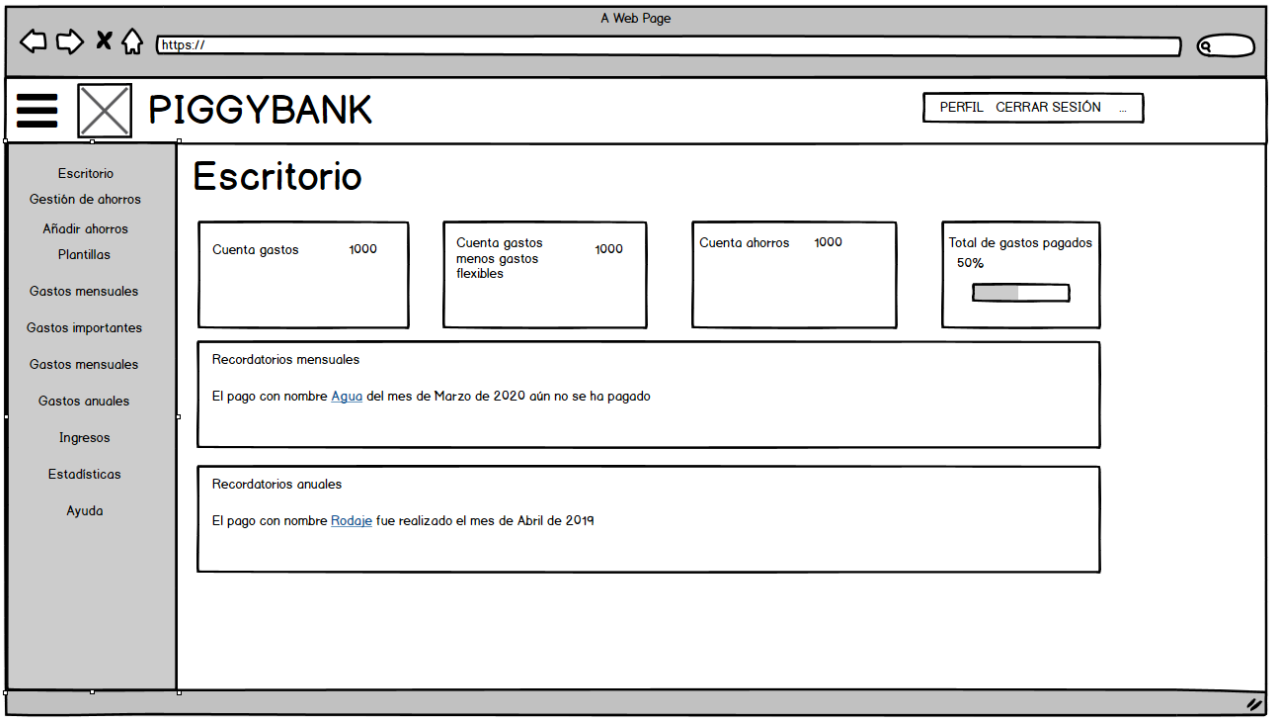


Ilustración 15: Wireframe dashboard escritorio

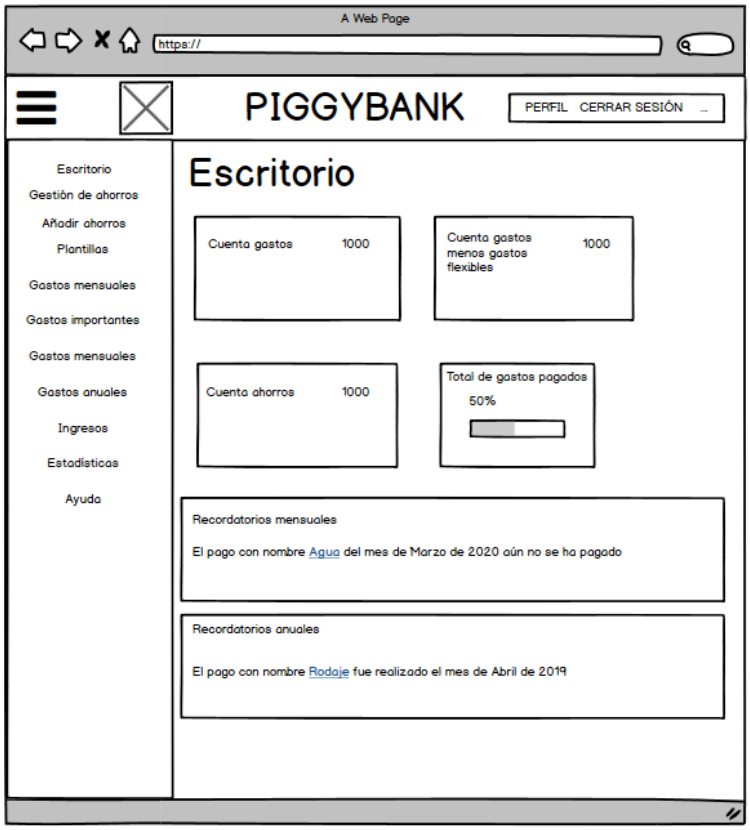


Ilustración 16: Wireframe dashboard tablet

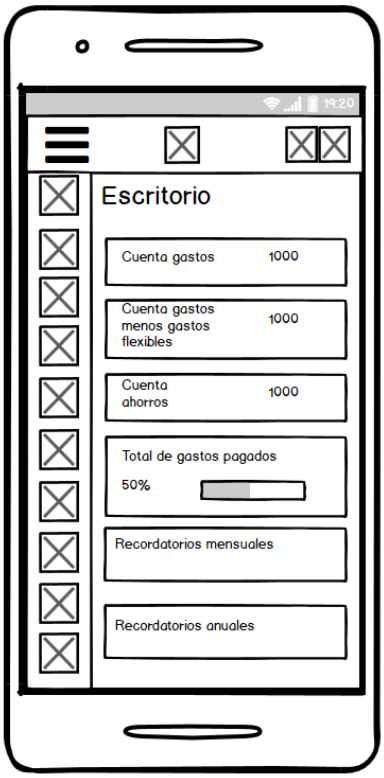


Ilustración 17: Wireframe dashboard móvil

- Pantalla de estadísticas

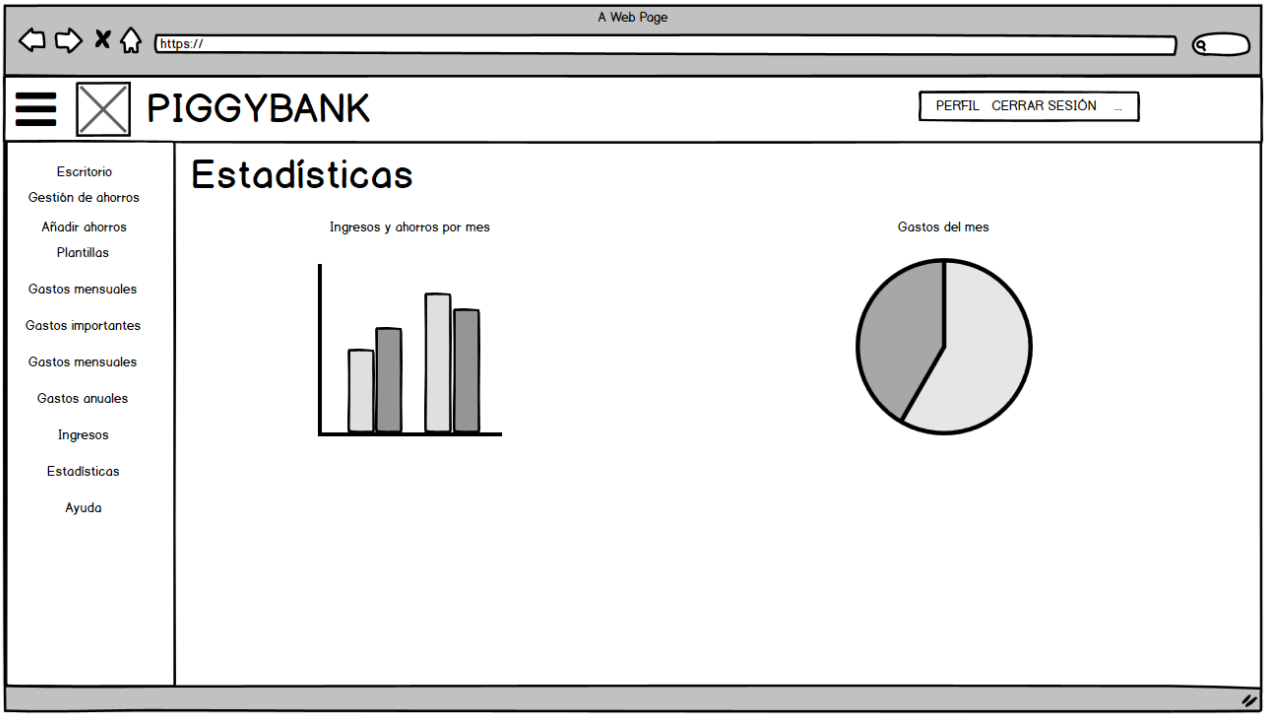


Ilustración 18: Wireframe estadísticas escritorio

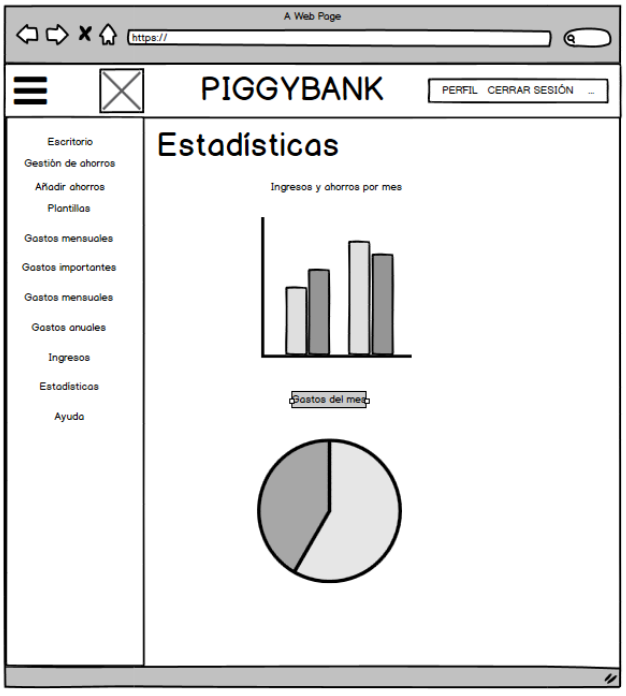


Ilustración 19: Wireframe estadísticas tablet

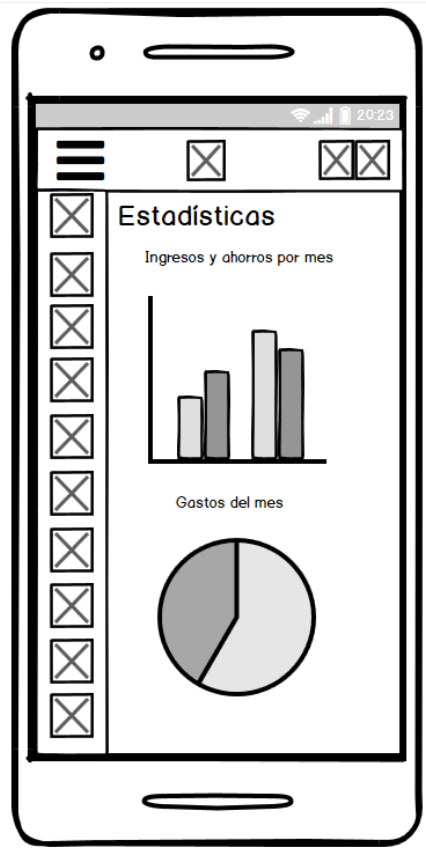


Ilustración 20: Wireframe estadísticas móvil

- Pantalla con tabla y buscador. Se utiliza en gestión de ahorros, añadir ahorros, gastos mensuales, gastos anuales, gastos importantes, ingresos, administrador de usuarios, buenas prácticas, tipos de gastos y tipos de ahorros.

Ilustración 21: Wireframe tabla escritorio

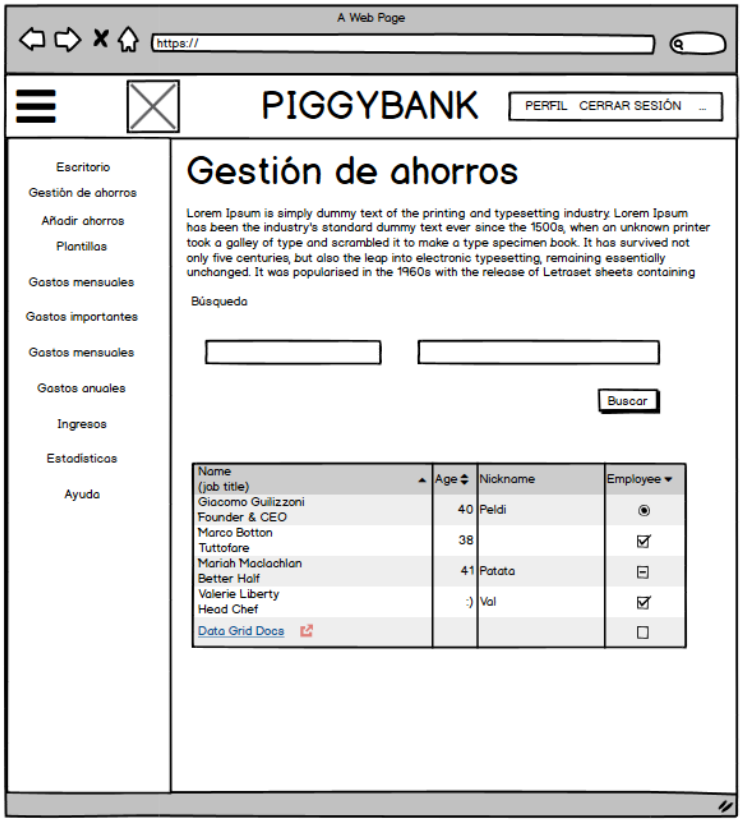


Ilustración 22: Wireframe tablas tablet

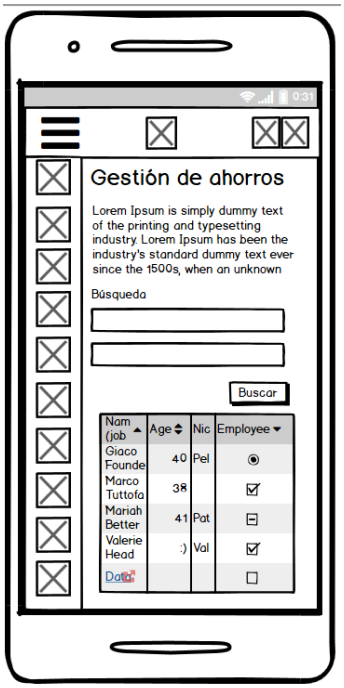


Ilustración 23: Wireframe tablas móvil

- Pantalla de plantillas

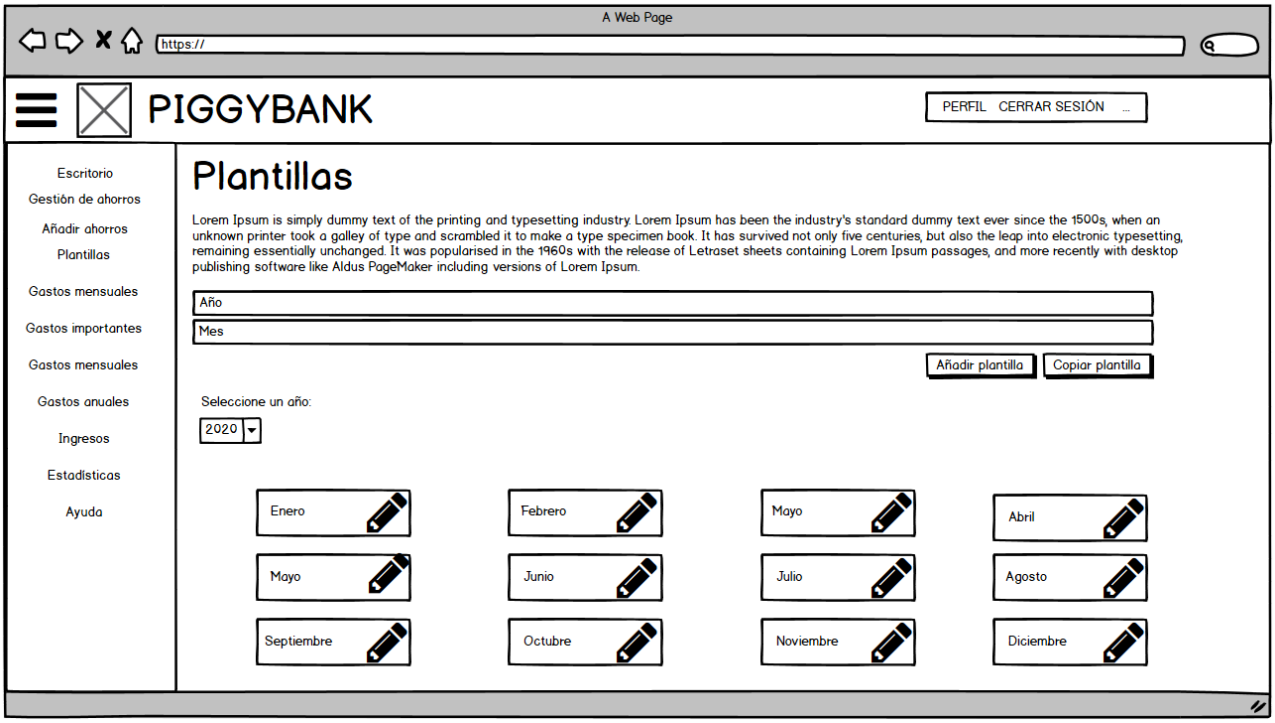


Ilustración 24: Wireframe plantillas escritorio

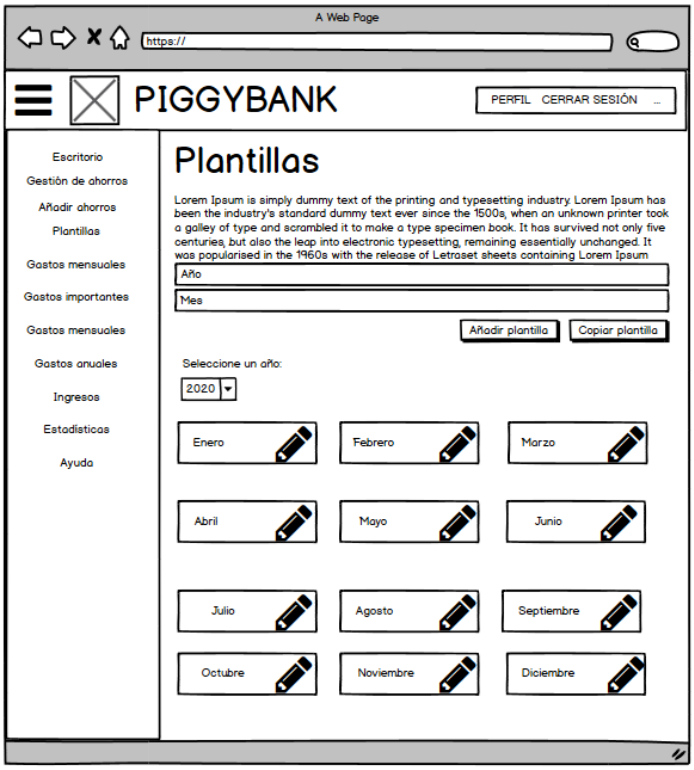


Ilustración 25: Wireframe plantillas tablet

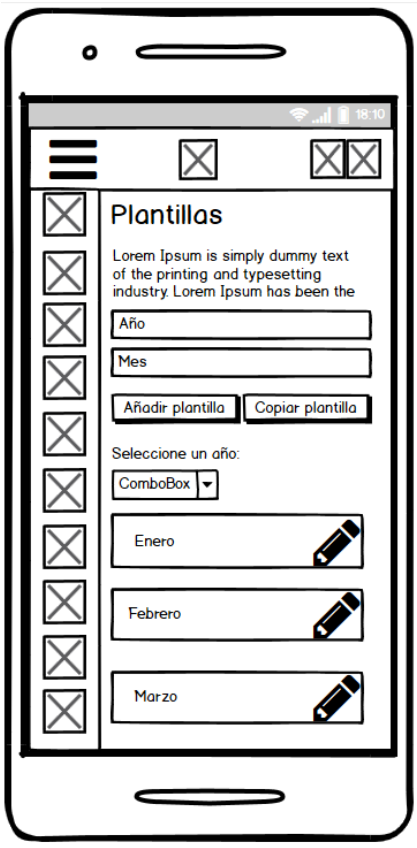


Ilustración 26: Wireframe plantillas móvil

- Pantalla de crear plantilla

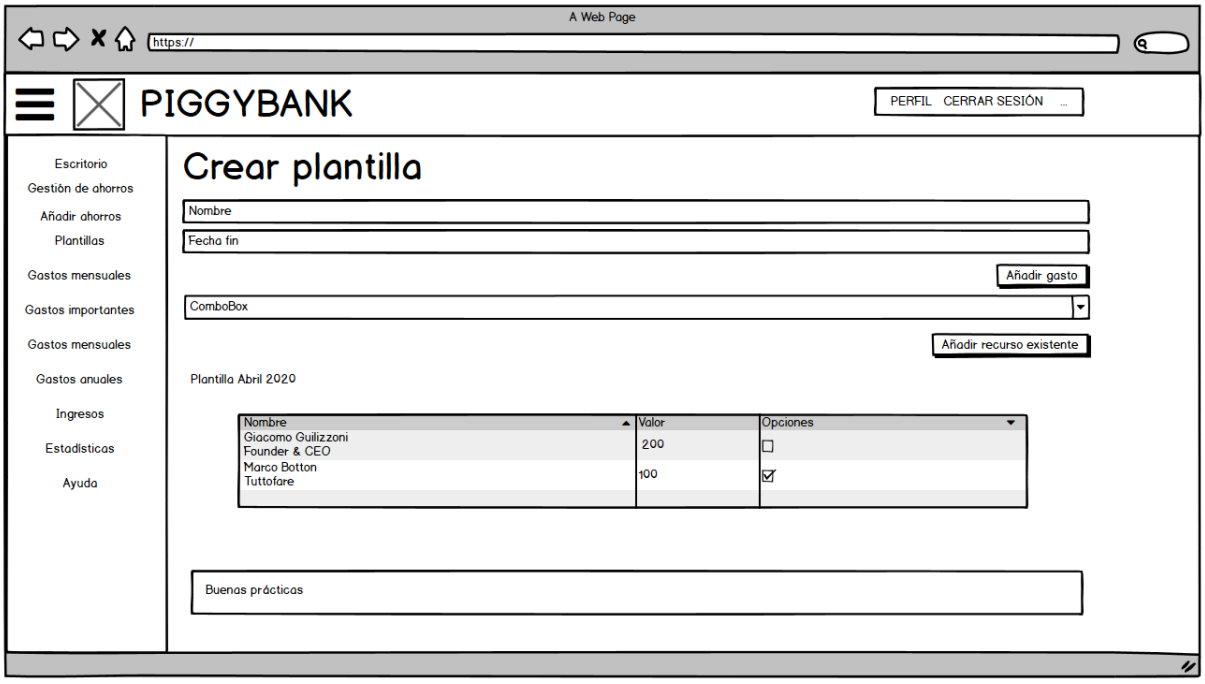


Ilustración 27: Wireframe crear plantilla escritorio

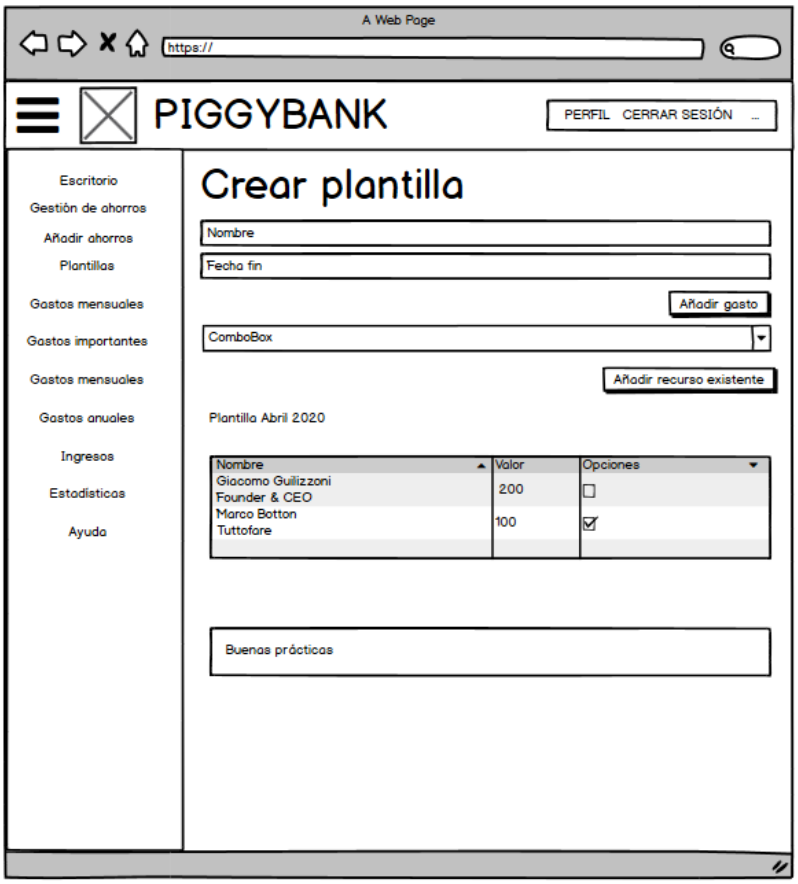


Ilustración 28: Wireframe crear plantilla tablet

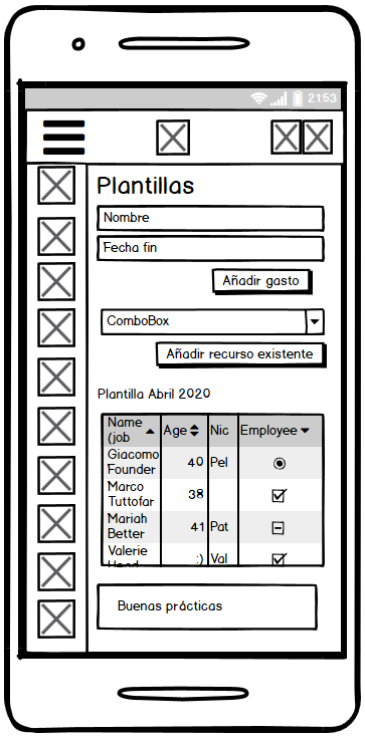


Ilustración 29: Wireframe crear plantilla móvil

11.2 Hi-Fi

Los prototipos han sido desarrollados en Angular con el fin de ahorrar tiempo en la fase de implementación, y con ello tener una base muy amplia de donde partir en la próxima fase de desarrollo. Las pantallas desarrolladas son las mismas que el apartado anterior ya que con ello tenemos todos los tipos de páginas de los que consta la aplicación.

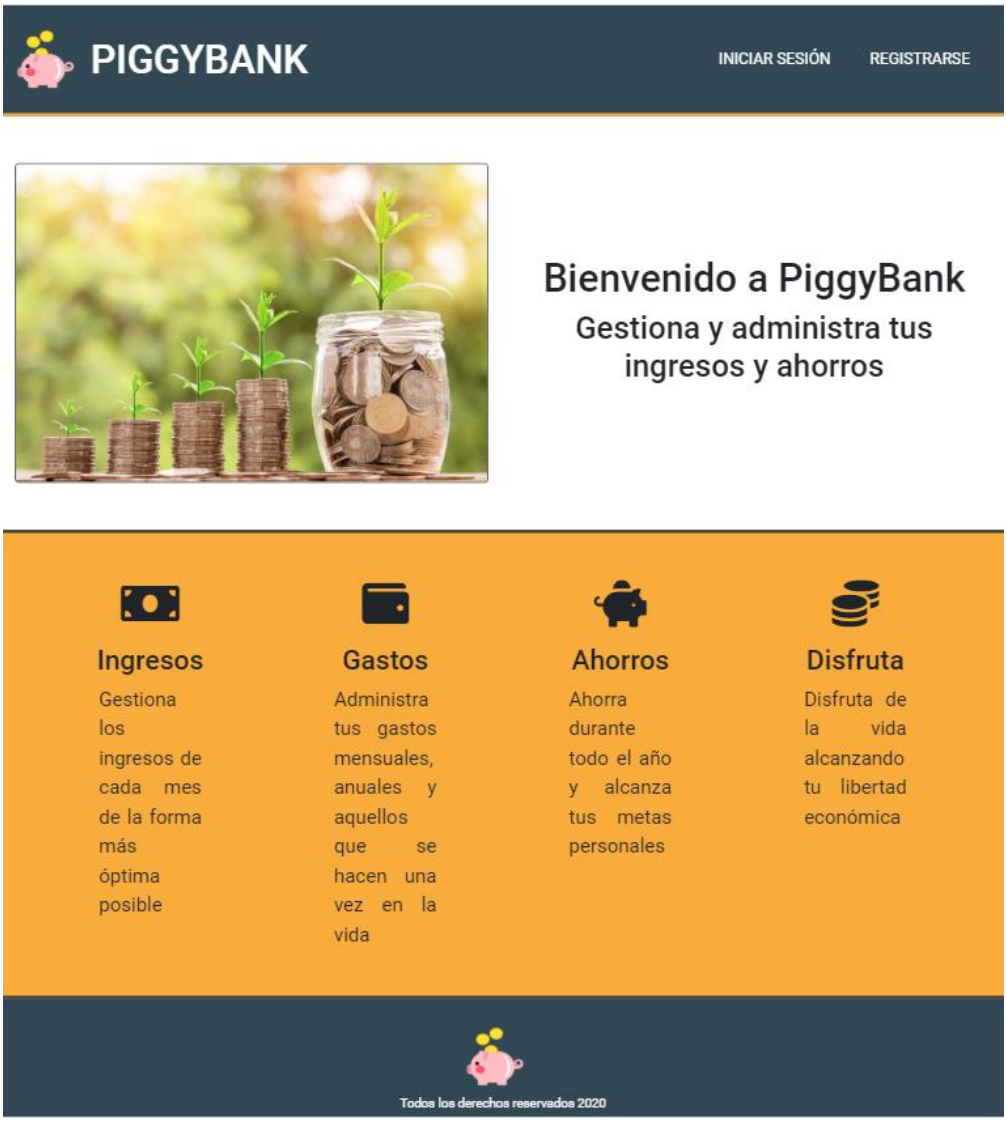


Ilustración 30: Pantalla de bienvenida

Ilustración 31: Pantalla de inicio sesión

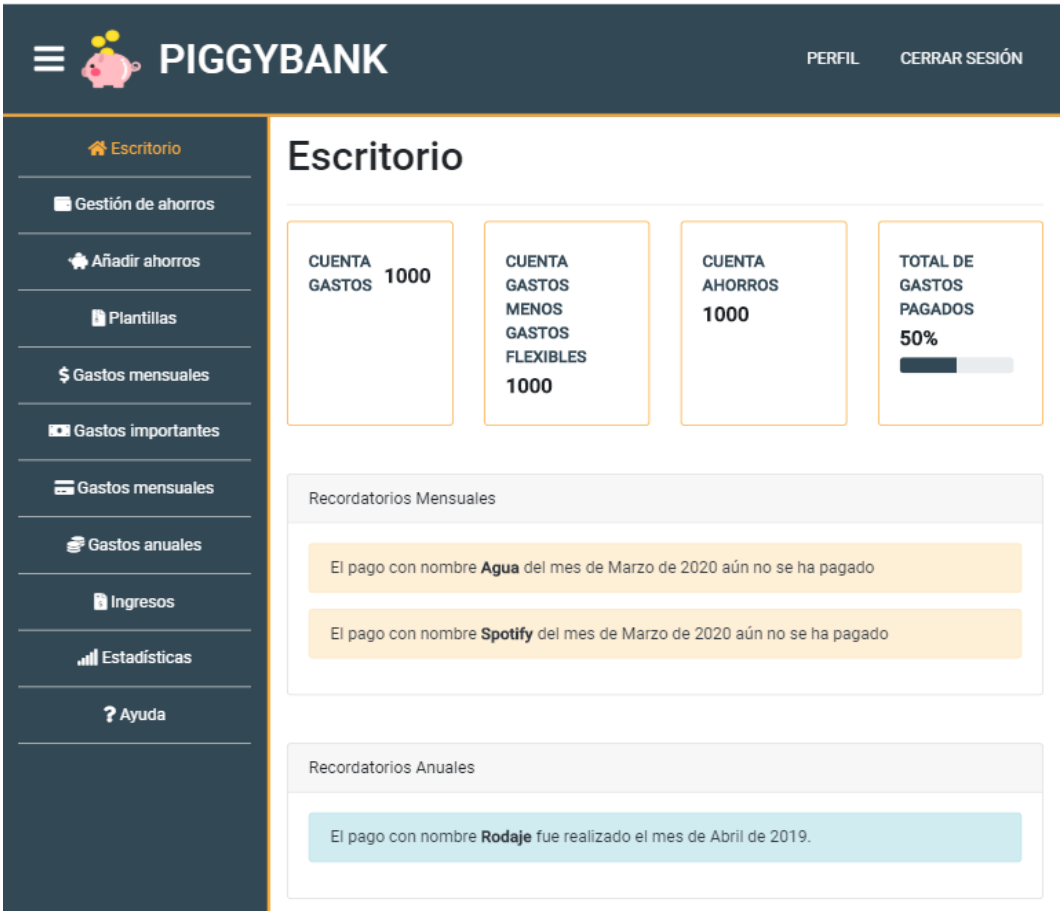


Ilustración 32: Pantalla de escritorio

Ilustración 33: Pantalla con tabla y búsqueda

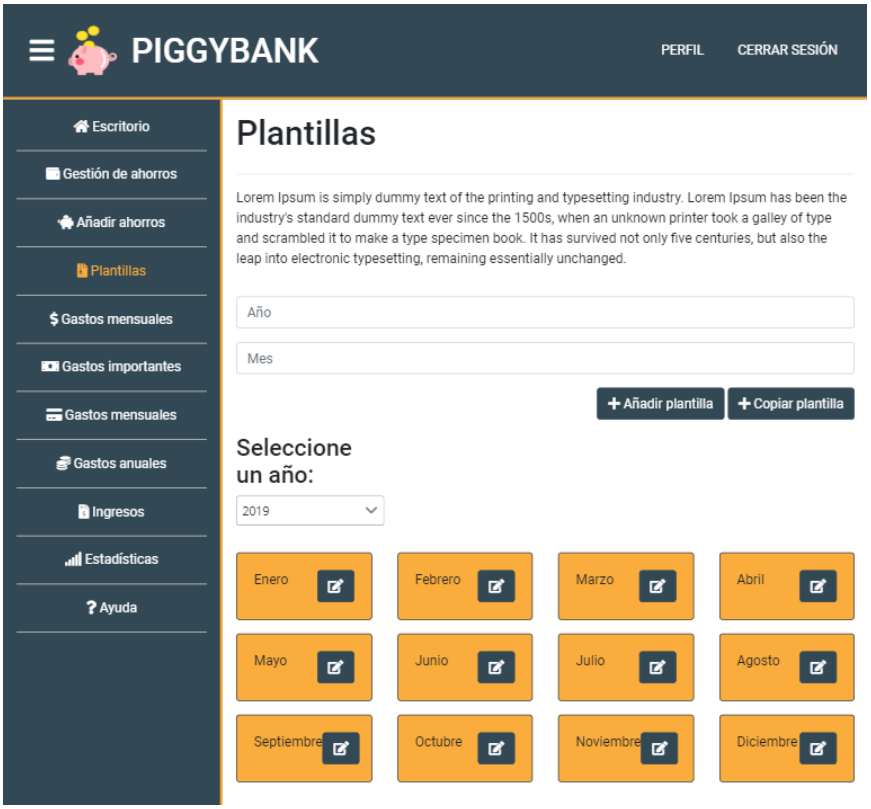


Ilustración 34: Pantalla de plantillas

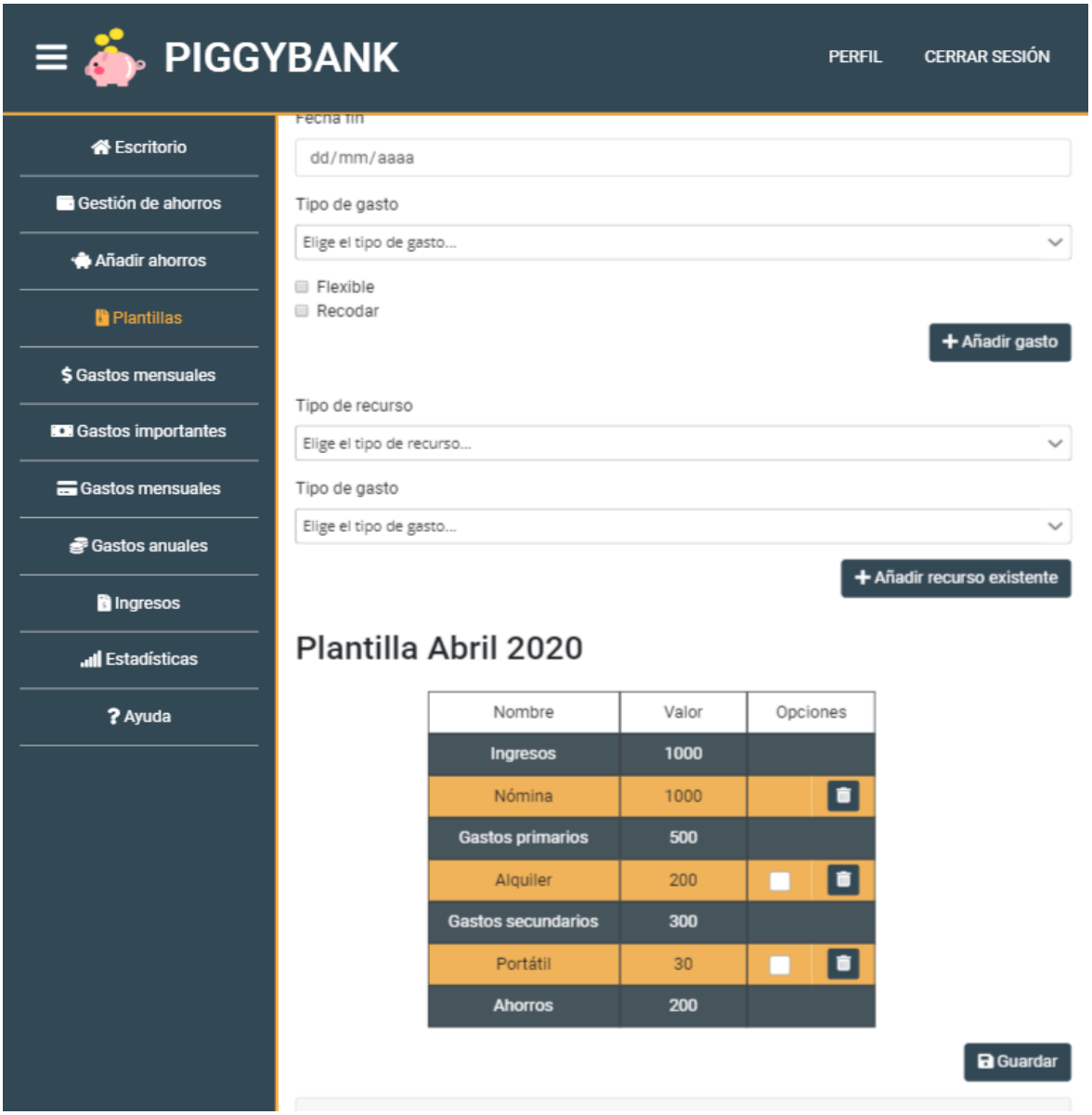


Ilustración 35: Pantalla de crear plantilla

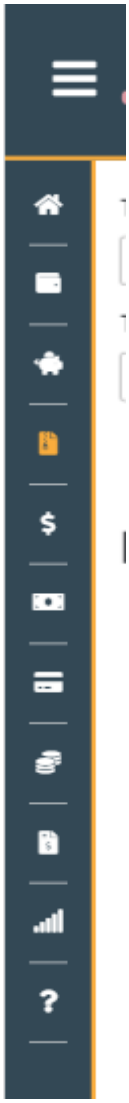


Ilustración 36: Menú colapsado

12. Perfiles de usuario

El estudio lo vamos a enfocar en los últimos años con el fin de obtener los datos más actualizados posibles. Dichos datos, se han obtenido del Instituto Nacional de Estadística y de otros estudios sobre la gestión de los ahorros.

Según la edad de los usuarios tenemos que determinar por una parte que el rango promedio para trabajar es desde los 16 hasta los 65 años, y por otra parte, debemos tener en cuenta con que edades los usuarios utilizan las TIC:

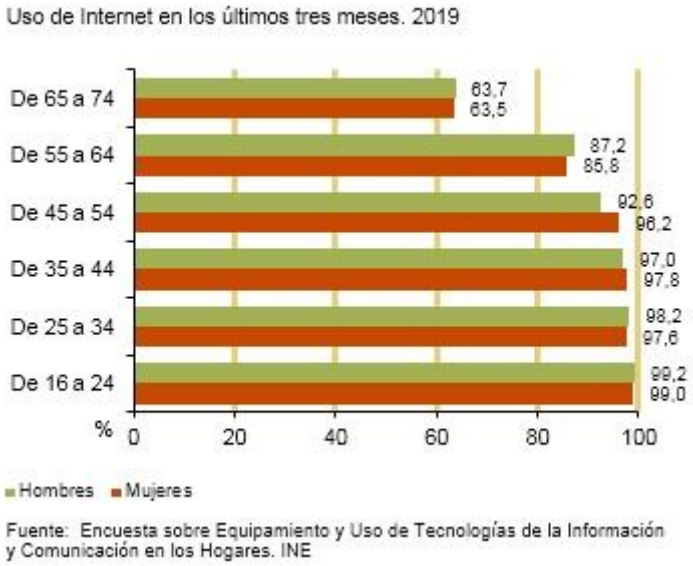


Ilustración 37: Rango de edades que utilizan las TIC

Además, existen estudios que dicen que a medida que aumenta la edad, también aumenta la capacidad de ahorro debido a la disminución de grandes gastos. Hasta los 25 años, los ahorros son limitados debido que son edades que aún se está estudiando o se está comenzado a trabajar, y a partir de los 25 años es cuando comienza a aumentar el ahorro. Por otro lado, desde los 30 a los 34 también aumenta, debido que las personas reservan dinero para afrontar la compra de una vivienda o los gastos de una boda, igual pasa desde los 35 hasta los 40.

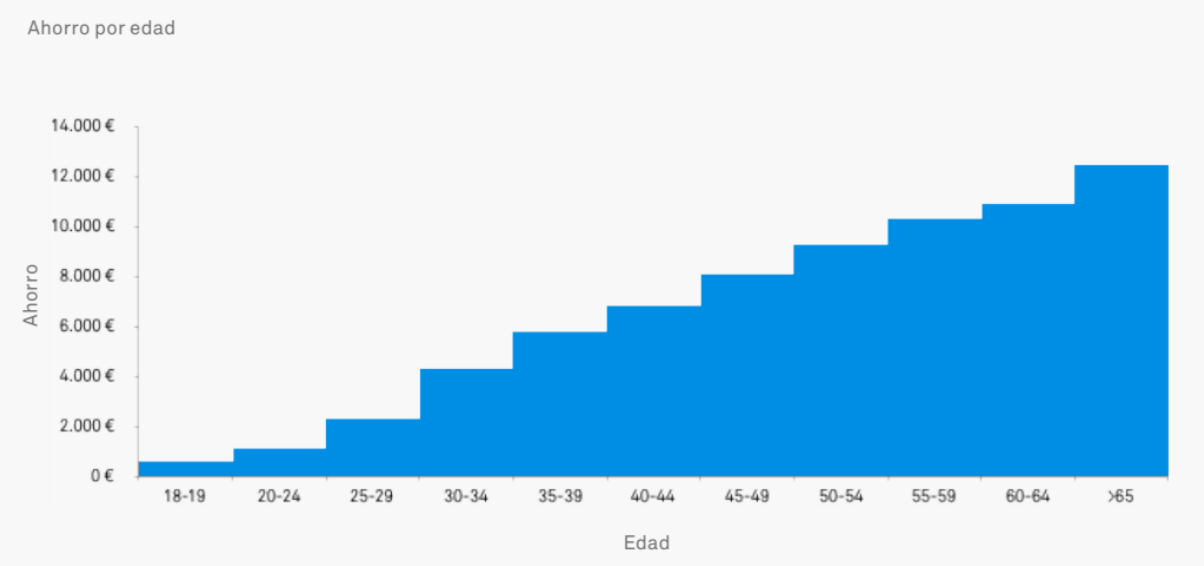


Ilustración 38: Ahorro por edad

De esta forma podemos determinar que nuestro público potencial está conformado por hombres y mujeres con edades comprendidas entre los 18 y 40 años, ya que son las edades donde se hace un mayor uso de las TIC, además de ser las edades donde mejor se necesita gestionar y administrar la propia economía para afrontar gastos importantes como por ser edades donde es más difícil ahorrar.

13. Usabilidad/UX

13.1 Principios de usabilidad

La aplicación cumple los principios de usabilidad:

1. *Visibilidad del estado del sistema: el sistema siempre debería mantener informados a los usuarios de lo que está ocurriendo, a través de retroalimentación apropiada de un tiempo razonable*.
   Este principio se justifica dado que el menú se queda marcado con un estilo diferente con aquella opción que hayamos escogido.

Ilustración 39: Opción del menú resaltada

Además, cuando accedemos a cualquier opción del menú se refleja en las diferentes pantallas el título de la opción donde nos encontramos.

Ilustración 40: Título superior pantalla

2. *Relación entre el sistema y el mundo real: el sistema debería hablar el lenguaje de los usuarios mediante palabras, frase y conceptos que sean familiares al usuario, más que con términos relacionados con el sistema. Seguir las convenciones del mundo real, haciendo que la información aparezca en un orden natural y lógico.*
   El sistema en todo momento que quiere comunicarse con el usuario hace uso de un lenguaje totalmente entendible y que describe con la mayor brevedad posible la acción que haya hecho el usuario.

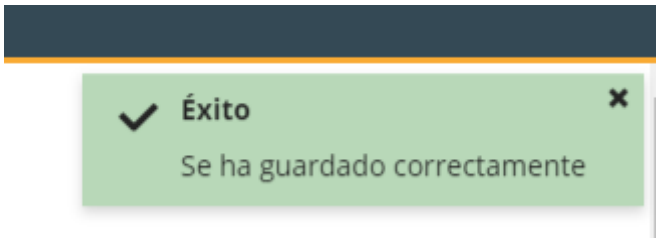


Ilustración 41: Mensaje de éxito

3. *Control y libertad del usuario: hay ocasiones en que los usuarios elegirán las funciones del sistema por error y necesitarán una "salida de emergencia" claramente marcada para dejar el estado no deseado al que accedieron, sin tener que pasar por una serie de pasos. Se deben apoyar las funciones de deshacer y rehacer.*
   A la hora de que el usuario decida ejecutar una acción de borrar cualquier registro de base de datos, se le ofrece la posibilidad de poder confirmar dicha acción en caso de que haya pulsado el botón de "Eliminar" por error.
4. *Consistencia y estándares: los usuarios no deberían cuestionarse si acciones, situaciones o palabras diferentes significan en realidad la misma cosa; siga las convenciones establecidas.*
   En esta aplicación se siguen los estilos establecidos por otras aplicaciones y ejemplo de ello es, el menú de hamburguesa que se utiliza en dispositivos más pequeños. Dicho menú es fácil de entender por cualquier usuario ya que se usa en cualquier otro tipo de aplicación.
5. *Prevención de errores: mucho mejor que un buen diseño de mensajes de error es realizar un diseño cuidadoso que prevenga la ocurrencia de problemas.*
   Dentro de la aplicación se busca prevenir errores, y ejemplo de ello es la doble confirmación de contraseña para que de esa forma, asegurar que el usuario escriba dos veces la contraseña que desea utilizar en la aplicación.

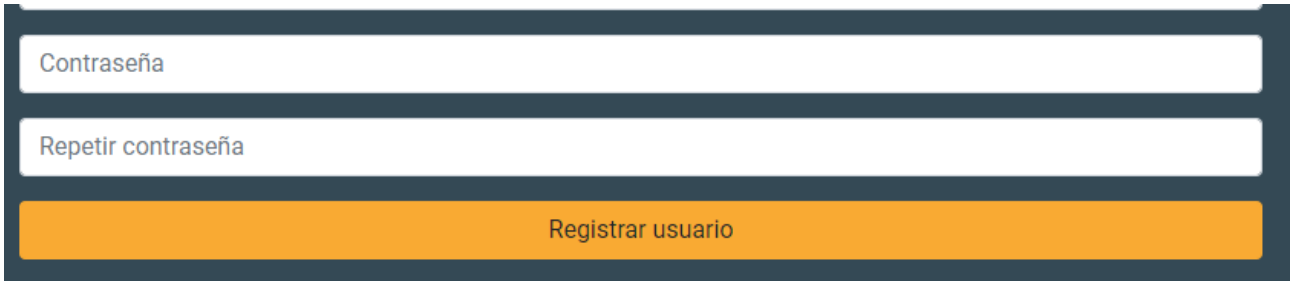


Ilustración 42: Doble confirmación de contraseña

6. *Reconocimiento antes que recuerdo: se deben hacer visibles los objetos, acciones y opciones. El usuario no tendría que recordar la información que se le da en una parte del proceso, para seguir adelante. Las instrucciones para el uso del sistema deben estar a la vista o ser fácimentel recuperables cuando sea necesario.*

   En todo momento el usuario es capaz de acceder a cualquier funcionalidad de la aplicación debido que el menú está presente en todo momento y por ello, no tiene que recurrir a volver a una página inicial sino navegar libremente sin estar recordando cuáles son las funcionalidades claves del sistema.
7. *Flexibilidad y eficiencia de uso: la presencia de aceleradores, que no son vistos por los usuarios novatos, puede ofrecer una interacción más rápida a los usuarios expertos que la que el sistema puede proveer a los usuarios de todo tipo. Se debe permitir que los usuarios adapten el sistema para usos frecuentes.*

   Los usuarios a través del escritorio tienen el atajo de poder acceder directamente a una plantilla en concreto pinchando sobre algunos de los recordatorios existentes con el fin de acceder de una forma más rápida a la plantilla en cuestión.
8. *Estética y diseño minimalista: los diálogos no deben contener información que es irrelevante o poco usada. Cada unidad extra de información en un diálogo, compite con las unidades de información relevante y disminuye su visibilidad relativa.*

   En todo momento se busca que las páginas que conforman la aplicación tengan el mínimo de información posible, ejemplo de ello, son las páginas de inicio y registro donde cuentan con un simple formulario acompañado de un botón y de enlaces a las páginas más importantes.
9. *Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores: los mensajes de error se deben entregar en un lenguaje claro y simple, indicando en forma precisa el problema y sugerir una solución constructiva al problema.*

   En todos los formularios, aquellos campos que sean obligatorios o que deban cumplir con una serie de requisitos, se le comunicará al usuario si los campos los está rellenando de forma correcta, y se les mostrará mensajes con los errores que cometa.
10. *Ayuda y documentación: incluso en los casos que el sistema pueda ser usado sin documentación, podría ser necesario ofrecer ayuda y documentación. Dicha información debería ser fácil de buscar, estar enfocada en las tareas del usuario, con una lista concreta de pasos a desarrollar y no ser demasiado extensa.*

    Mediante el menú los usuarios pueden acceder a una sección de ayuda, donde tienen un manual de usuario que les permitirá conocer mejor el funcionamiento de la aplicación.

Ilustración 43: Sección de ayuda

13.2 Árbol de navegación

En la siguiente imagen se muestra el árbol de navegación de la aplicación donde podemos ver una sección para usuarios con rol administrador y otra para aquellos con rol usuario:

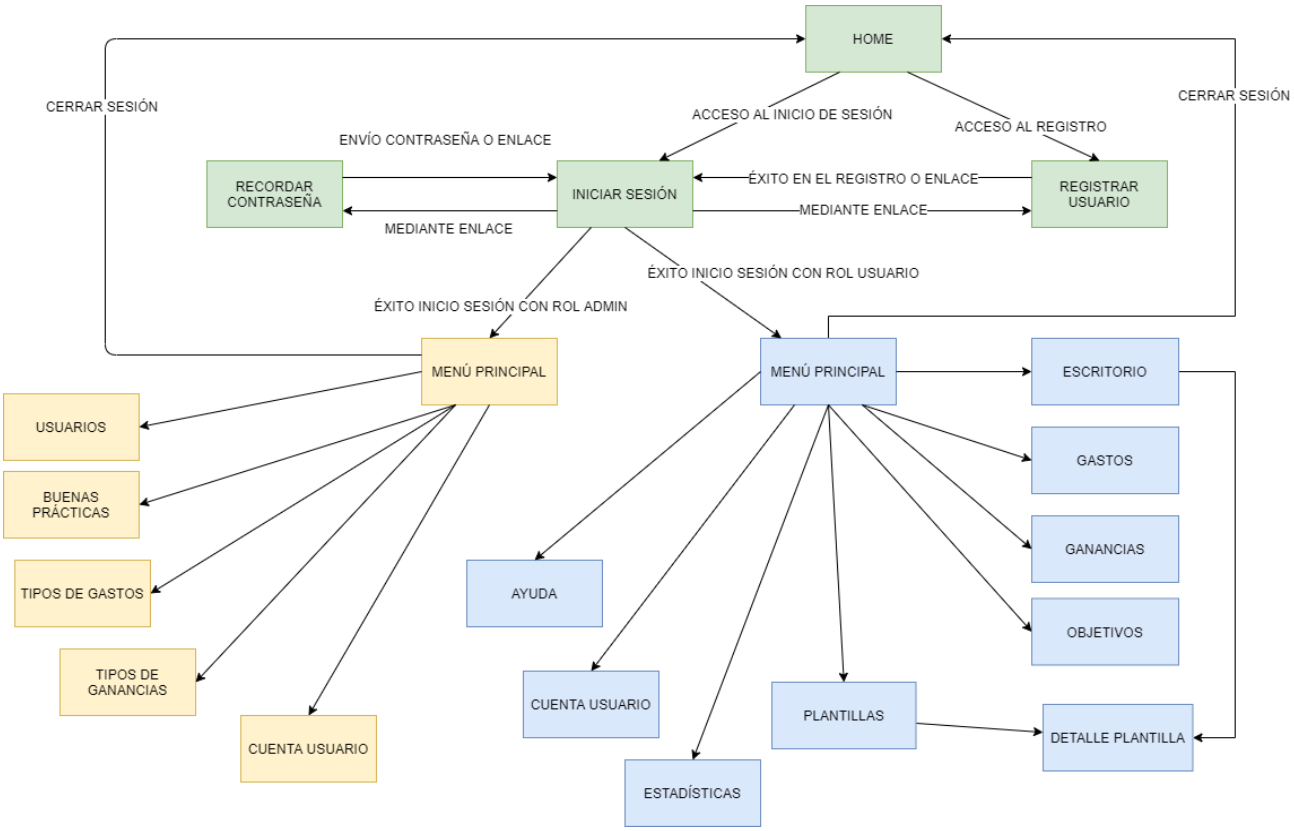


Ilustración 44: Árbol de navegación

Con fin de no aumentar el tamaño del árbol de navegación, es necesario comentar que todas las pantallas una vez nos hayamos autenticado, disponen de una pantalla de detalle excepto las pantallas de: escritorio, ayuda, cuenta usuario y estadísticas.

13.3 Patrones de diseño

Los patrones de diseño utilizados en la aplicación son:

- De personalización.
    - Login: Los usuarios necesitan identificarse contra una base de datos donde previamente se han registrado, y de esta forma debe haber un formulario donde se le pida una combinación de usuario o email junto a una contraseña para poder validarse. De esta forma siempre es necesario decantarse por ofrecer una pantalla donde poder introducir los datos de inicio. Además, es necesario ofrecer dicha pantalla ya que los usuarios necesitan usar la aplicación a través de un perfil que recoja toda la información que se genere durante el uso de la aplicación.

      Ventajas:

      - Espacio donde el usuario poder introducir sus datos de sesión y permitir de esa manera acceder con un perfil a la aplicación.
    - Registro: la pantalla de registro soluciona el problema de pedir al usuario los datos suficientes para poder crearle un perfil en la base de datos con el que poder conectarse a la aplicación. Por ello se ha decidido crear dicha pantalla y mediante un formulario permitir la posibilidad de registrar un usuario en la aplicación.

      Ventajas:

      - Espacio donde el usuario puede introducir sus datos con el fin de poder tener un perfil con el que acceder a la aplicación.
- De dando entrada.
    - Entrada mediante un select: el usuario puede introducir datos en el formulario con la sintaxis correcta ya que los select sólo le permiten elegir aquellos elementos disponibles. Esto le permite al usuario no tener conocimiento exacto de las entradas posibles porque es el formulario quien le aporta dicha información.
    - Form: el usuario al necesitar enviar datos a la aplicación necesita de un espacio que le permita enviar información, y éste es el formulario, por ello se decide de añadir uno que permita obtener los datos de inicio de sesión. Dicho formulario se compone de una etiqueta que especifica el dato que debemos rellenar y un input que recoge dicho dato.

      Ventajas:

      - Permite un medio de comunicación para el usuario donde poder enviar una serie de datos.
- Sobre navegación.
    - Menú hamburguesa: es utilizado en la pantalla móvil y tablet porque con ello aprovechamos mejor el espacio reducido que nos ofrecen estos dispositivos. Dicho menú aparece cuando pulsemos sobre el icono de la hamburguesa.

      Ventajas:

- Pueden contener todo el listado del menú de forma oculta sólo hasta que deseemos mostrar.

- Navegación principal: permite a los usuarios navegar en todo momento a las diferentes secciones de las que se compone la aplicación. En este caso se ha decidido usar un menú lateral de elementos clickables y que se mantenga durante toda la aplicación. Se decide usar un menú para que el usuario tenga disponible la información de los apartados a los que puede navegar y permita conocer de esta forma la estructura de la que se compone el conjunto de la aplicación

  Ventajas:

  - Espacio que siempre se encuentra disponible para poder acceder a otras secciones de la aplicación.
  - Muestra información de la estructura de la aplicación.

- De diseño de búsqueda.
  - Ordenar por columna: cada columna de la tabla se compone de un filtro para poder ordenar de forma ascendente o descendente dicha columna, de forma que se puedan ver ordenados los resultados para esa columna. Dicho orden, se basa también en el tipo del elemento, es decir, puede ser ordenado de forma alfabética o por fechas más antiguas. Además, ya que utilizamos paginación, es necesario filtrar por columnas ya que nos permite no tener que ir navegando por cada página en busca de algún resultado que si podamos conseguir ordenando de esta forma. Por otro lado, al no saber el tamaño que puede llegar alcanzar nuestras tablas, permitimos una forma más de ordenar, permitiendo al usuario más herramientas de búsqueda.

    Ventajas:

    - Organiza y ordena una lista de resultados.
  - Caja de búsqueda: el usuario para poder buscar un elemento o conjunto de elementos necesita de un espacio donde poder escribir su búsqueda, y para ello, están las cajas de búsqueda.

    Ventajas:

    - El usuario sólo filtra los elementos que desee.
    - Permite buscar un elemento en concreto dentro de un listado con muchos resultados.
  - Búsqueda de resultados: en los casos de esta pantalla, es necesario que se muestre un listado de resultados acorde a unos parámetros de búsqueda. Dicha lista de resultados se muestra de forma ordenada y sólo con los resultados filtrados. Los usuarios para realizar la búsqueda usan la caja de búsqueda, la cual es un patrón de diseño que se explica más adelante.

    Ventajas:

    - Se muestra los resultados de forma filtrada, lo que hace más fácil encontrar resultados.

- De interacciones básicas.
    - Botón de acción: se ponen botones de acción ya que se requiere por parte del usuario, la ejecución de acciones importantes dentro de la aplicación.
    - Paginación: debido a que no sabemos el tamaño que va a alcanzar las tablas que se usan en nuestra aplicación, aportamos un paginador que permita navegar entre las diferentes páginas de las que se compone el listado de resultados. Además, la paginación permite a nivel de desarrollo que no se hagan búsquedas muy grandes de resultados, sino que sea el paginador quien determine el número de resultados por página que se va a conseguir.
      Ventajas:
        - Permite un elemento de búsqueda más en caso de listas de resultados muy grandes.
        - No supone carga de elementos muy grandes.
- De layout.
    - Mostly fluid: mediante este patrón se define que a medida que el tamaño de la pantalla se va reduciendo, los elementos se van apilando en una sola columna. De esta manera, según el tamaño, los ítems se van colocando de tal forma que puedan visualizarse. El patrón no implica que se reduzca en una sola columna, sino que se trata de un método donde se va reduciendo el tamaño y por ello los elementos se van apilando verticalmente ocupando el espacio referente al tamaño de la pantalla.
- Comunes.
    - Repetición por las características visuales que comparten. Los elementos no tienen que ser perfectamente idénticos para agruparse de manera repetitiva, deben compartir simplemente un rasgo común. Esto lo hemos podido ver en los listados de resultados, donde los ítems están agrupados debido a que pertenecen a un grupo común.

14. Seguridad

Con el objetivo de aportar mayor seguridad a las llamadas realizadas a la API Rest lo que se ha hecho es securizar dichas llamadas con JWT. Se trata de un estándar abierto basado en JSON donde el servidor genera un token que envía al cliente de forma que ambos puedan verificar que el token es verídico. De esta forma, si se hacen llamadas desde el cliente donde el usuario no haya iniciado sesión o mediante un token inválido, el servidor se encargará de rechazar la petición y con ello aseguramos que sólo se puedan realizar llamadas de usuarios registrados en la aplicación y puedan obtener sus datos a través del propio token.

Dentro de nuestra aplicación hemos aplicado JWT de esta manera:

- Los tokens tienen una validez de una hora de forma que los usuarios sólo puedan permanecer durante este tiempo con la sesión iniciada y después de ello, deben volver a iniciar sesión en la aplicación para obtener un token nuevo.
- Los token son guardados dentro del Local Storage del navegador de forma que cuando el cliente necesite utilizarlos, los pueda recuperar desde ese sitio.
- En el servidor, las rutas están securizadas por los dos roles de la aplicación, de forma que para acceder a un servicio web, se valida que exista un token válido y además que el usuario tenga el rol que se requiere para acceder. El encargado de validar lo comentado es un middleware llamado JwtMiddleware.
- En el cliente, las rutas están securizadas por roles de formas que los guards de Angular se encargan de validar si el token tiene permisos para acceder a una determinada ruta.
- El interceptor de Angular se encarga de añadir a la cabecera de las llamadas http el token necesario para validarnos frente a la API.
- Cuando modificamos nuestra contraseña en la pantalla de cuenta del usuario, el sistema se encarga de invalidar el token que tenemos y otorgarnos uno nuevo.

Además, en cuanto a seguridad, todas las contraseñas se guardan en base de datos con un hash y por otro lado, todas las operaciones de guardar y validar se encargan de validar que los campos obligatorios para guardar los objetos en base de datos sean recibidos en la propia petición.

15. Tests

En el desarrollo únicamente se han realizado pruebas de aceptación. No se han realizado pruebas funcionales ni e2e sino que el producto junto a sus requisitos se iba mostrando al cliente que se encargaba de validar que los requisitos establecidos desde un primer momento se estaban cumpliendo.

En dichas pruebas, los usuarios realizaban una serie de pruebas sobre la aplicación, y tras los resultados obtenidos, pedían nuevos requisitos, mejoras o correcciones. Todo esto se iba a implementar en el código a medida que realizaban las pruebas.

Algunos de los escenarios establecidos en la pruebas fueron:

- Registro e inicio de sesión de usuarios.
- Mostrar cantidad de ahorros acumulados.
- Mostrar cantidad de dinero disponible del mes actual.
- Notificar pagos pendientes.
- Establecer metas de ahorro.
- Crear gastos y ganancias.
- Aplicar método de ahorro a los ingresos y pagos de un mes.
- Copiar plantillas creadas.
- Marcar pagos realizados.
- Estadísticas de la economía.
- Consejos y buenas prácticas sobre las plantillas.

Además, si nos centramos en un escenario en concreto podemos ver el diseño de la prueba:

- **Nombre**: Aplicar método de ahorro a los ingresos y pagos de un mes
- **Condiciones**
  - Gastos primarios disponibles.
  - Gastos secundarios disponibles.
  - Ahorro deseados del mes.
  - Ahorros reales.
- **Pasos**
  - Gastos primarios se obtienen del 50% de los ingresos.
  - Gastos secundarios se obtienen del 30% de los ingresos.
  - Ahorros deseados se obtienen del 20% de los ingresos.
  - Ahorros reales se obtienen del dinero disponible una vez descontados los gastos.
- **Resultado esperado**
  - Automáticamente cada elemento se muestra por pantalla con los cálculos realizados.

16. Requisitos de instalación/implantación/uso

Información detallada acerca de los recursos necesarios para la instalación, implantación y uso:

- Servidor
  - Servidor Apache 2.4.43 o superior.
  - PHP 7.2.30 o superior.
  - phpMyAdmin 5.0.2 o superior.

- Cliente
  - Navegador Google Chrome o Firefox.

- Hardware
  - Alojamiento en el servidor compartido de la UOC.
  - Acceso a la aplicación a través del cliente por cualquier dispositivo: ordenador, móvil o Tablet.

- Formación
  - Experiencia del propio Máster y laboral durante tres años.

17. Instrucciones de instalación/implantación

Para la instalación de la aplicación es necesario:

- En la parte del front-end es necesario la instalación de:
    - Nodejs + npm: en concreto para la aplicación desarrollada se ha utilizado la versión 12.6.1 de Nodejs y la versión 6.4.12 de npm. Estas herramientas nos permiten la gestión de paquetes a través del package.json y de poder tener un entorno de desarrollo donde poder ir construyendo la aplicación. Podemos obtener esta herramienta desde su página web https://nodejs.org/es/ y descargar el ejecutable recomendable para la mayoría.
    - Angular: mediante npm podemos descargar Angular CLI, herramienta necesaria para la creación tanto del proyecto de Angular como de los elementos del propio framework que vayamos creando. Se instala mediante **npm install @angular/cli@latest**. A continuación, creamos el proyecto mediante ng new nombreproyecto y con ello ya tenemos nuestro proyecto listo en Angular 9 para comenzar el desarrollo.

- En la parte del back-end es necesario la instalación de:
    - XAMPP: es un servidor web que nos proporciona todo lo necesario para poder arrancar una aplicación PHP: Servidor Apache, el propio lenguaje PHP, base de datos MySQL y administrador de base de datos phpMyAdmin. Para poder utilizarla sólo es necesario arrancar el servidor Apache y MySQL desde su panel de control, además de tener nuestro proyecto en la carpeta htdocs.
    - Composer: es un gestor de dependencias para PHP que podemos instalar a partir del ejecutable descargable de su página web.
    - Laravel: es un framework PHP que, gracias a la tecnología anteriormente comentada, podemos crear un proyecto en Laravel 7 mediante: **composer create-project laravel/laravel PiggyBank –prefer-dist**

Para la implantación en un servidor es necesario:

- Obtener url del servidor para indicar dicha url dentro del archivo de environment de producción de Angular. De esta forma, las llamadas a la api se realizan a la url del servidor.
- Compilar proyecto de Angular indicando el base href de nuestro usuario dentro del servidor compartido.
- Subir el contenido de la aplicación al servidor. En este paso, no es necesario subir la carpeta del frontend dado que lo único necesario son los archivos que se generan tras la compilación de este.
- Modificar fichero .env con los datos de conexión de la base de datos del servidor.
- Crear base de datos en el servidor.
- Exportar las tablas de la base de datos de local e importarlas en el servidor.

- Añadir fichero .htaccess dentro de la raíz que permite sólo accesos al directorio /public de la aplicación con el fin de no permitir desde el navegador, conocer las carpetas de las que se compone la aplicación junto a sus ficheros.
- Dar permisos sobre la carpeta public de Laravel a todas las peticiones http.

18. Instrucciones de uso

El enlace de la aplicación es: http://eimtcms.eimt.uoc.edu/~oalvarezher/piggy-bank
Para el uso de la aplicación como administrador se aporta la cuenta con email oalvarezher@uoc.edu y contraseña Omar. Si deseamos probar la aplicación como usuario, es necesario registrarse en la aplicación.

Información con pasos detallados acerca de cómo se debe utilizar el servicio/aplicación:

- Registro de usuarios
    1. Desde la pantalla de bienvenida, accedemos al enlace “Registrarse”.
    2. Se rellenan los campos nombre, apellidos, correo electrónico, contraseña y repetir contraseña.
    3. Si se ha realizado con éxito, navegaremos automáticamente a la pantalla de inicio sesión donde aparece un mensaje de éxito con fondo verde y en caso contrario aparece un mensaje de error con fondo rojo.
- Iniciar sesión
    1. Desde la pantalla de bienvenida, accedemos al enlace “Iniciar sesión” o desde la pantalla de registro pulsando sobre el enlace “Si ya tienes una cuenta, inicia sesión”.
    2. Se rellenan los campos correo electrónico y contraseña.
    3. Si se ha realizado con éxito, navegaremos automáticamente dentro de la aplicación y en caso contrario aparece un mensaje de error con fondo rojo.
- Olvidaste tu contraseña
    1. Se rellena el campo correo electrónico.
    2. Se nos envía a nuestro correo un token que debemos utilizar en el formulario que nos aparece de forma automática introduciendo el token comentado y la nueva contraseña.
    3. Si se ha realizado con éxito, navegaremos automáticamente a la pantalla de inicio sesión donde aparece un mensaje de éxito con fondo verde y en caso contrario aparece un mensaje de error con fondo rojo.
- Usuarios (rol administrador)
    1. Búsqueda. Se pueden buscar usuarios filtrando por nombre, apellidos, correo electrónico o rol.
    2. Añadir: pulsamos el botón de “Añadir”, rellenamos el formulario que nos aparece y pulsamos sobre el botón "Guardar”.
    3. Editar: pulsamos sobre el usuario que deseamos editar, modificamos los valores que deseamos y pulsamos sobre el botón “Guardar”.
    4. Eliminar: pulsamos sobre el usuario que deseamos eliminar y apretamos el botón “Eliminar”. Una vez hecho esto, nos aparece un mensaje de confirmación que debemos confirmar en caso de que deseemos borrar el usuario.

- Tipos de gastos/Tipos de ganancias (rol administrador)
    1. Búsqueda. Se pueden buscar tipos filtrando por el campo valor y descripción.
    2. Añadir: pulsamos el botón de “Añadir”, rellenamos el formulario que nos aparece y pulsamos sobre el botón "Guardar”.
    3. Editar: pulsamos sobre el tipo que deseamos editar, modificamos los valores que deseamos y pulsamos sobre el botón “Guardar”.
    4. Eliminar: pulsamos sobre el usuario que deseamos eliminar y apretamos el botón “Eliminar”. Una vez hecho esto, nos aparece un mensaje de confirmación que debemos confirmar en caso de que deseemos borrar el tipo.
- Buenas prácticas (rol administrador)
    1. Búsqueda. Se pueden buscar buenas prácticas filtrando por palabra clave.
    2. Añadir: pulsamos el botón de “Añadir”, rellenamos el formulario que nos aparece y pulsamos sobre el botón "Guardar”.
    3. Editar: pulsamos sobre la buena práctica que deseamos editar, modificamos los valores que deseamos y pulsamos sobre el botón “Guardar”.
    4. Eliminar: pulsamos sobre la buena práctica que deseamos eliminar y apretamos el botón “Eliminar”. Una vez hecho esto, nos aparece un mensaje de confirmación que debemos confirmar en caso de que deseemos borrar la buena práctica.
- Gastos (rol usuario)
    1. Accedemos a la opción del menú Gastos.
    2. Búsqueda. Se pueden buscar gastos filtrando por nombre o tipo de gasto.
    3. Eliminar: pulsamos sobre el botón con icono de papelera del gasto que queremos eliminar. Una vez hecho esto, nos aparece un mensaje de confirmación que debemos confirmar en caso de que deseemos borrar el gasto.
    4. Añadir: pulsamos sobre el botón de “Añadir” y navegamos de forma automática a la pantalla de detalle del gasto. En ella debemos rellenar el formulario que se nos presenta y pulsar una vez finalizado, sobre el botón “Guardar”. En caso de éxito se muestra un mensaje con fondo verde y en caso contrario, un mensaje de error con fondo rojo.
    5. Editar: pulsamos sobre el botón con icono de lápiz del gasto que queremos editar. Una vez hecho esto, navegamos de forma automática a la pantalla de detalle del gasto donde podemos modificar el campo del gasto que deseamos editar. Además, podemos añadir nuevos pagos al gasto en cuestión.
- Ganancias (rol usuario)
    1. Accedemos a la opción del menú Ganancias.
    2. Búsqueda. Se pueden buscar ganancias filtrando por nombre o tipo de ganancia.
    3. Eliminar: pulsamos sobre el botón con icono de papelera de la ganancia que queremos eliminar. Una vez hecho esto, nos aparece un mensaje de confirmación que debemos confirmar en caso de que deseemos borrar la ganancia.

4. Añadir: pulsamos sobre el botón de “Añadir” y navegamos de forma automática a la pantalla de detalle de la ganancia. En ella debemos rellenar el formulario que se nos presenta y pulsar una vez finalizado, sobre el botón “Guardar”. En caso de éxito se muestra un mensaje con fondo verde y en caso contrario, un mensaje de error con fondo rojo.
5. Editar: pulsamos sobre el botón con icono de lápiz de la ganancia que queremos editar. Una vez hecho esto, navegamos de forma automática a la pantalla de detalle de la ganancia donde podemos modificar el campo de la ganancia que deseamos editar. Además, podemos añadir nuevos ingresos a la ganancia en cuestión.

- Pagos (rol usuario)
    1. Accedemos a la opción del menú Gastos.
    2. Navegamos al detalle de un gasto mediante el botón con el icono de un lápiz.
    3. En caso de que el tipo de gasto sea diferente al tipo mensual (se gestionan en las plantillas), podemos dentro de la sección de “Listado de pagos” realizar las siguientes funciones:
        - Eliminar: pulsamos sobre el botón con icono de papelera del pago que queremos eliminar. Una vez hecho esto, nos aparece un mensaje de confirmación que debemos confirmar en caso de que deseemos borrar el pago.
        - Añadir: pulsamos sobre el botón de “Añadir” y navegamos de forma automática a la pantalla de detalle del pago. En ella debemos rellenar el formulario que se nos presenta y pulsar una vez finalizado, sobre el botón “Guardar”. En caso de éxito se muestra un mensaje con fondo verde y en caso contrario, un mensaje de error con fondo rojo.
        - Editar: pulsamos sobre el botón con icono de lápiz del pago que queremos editar. Una vez hecho esto, navegamos de forma automática a la pantalla de detalle del pago donde podemos modificar el campo que deseemos editar.
- Ingresos (rol usuario)
    1. Accedemos a la opción del menú Ganancias.
    2. Navegamos al detalle de una ganancia mediante el botón con el icono de un lápiz.
    3. En caso de que el tipo de ganancia sea diferente al tipo mensual (se gestionan en las plantillas), podemos dentro de la sección de “Listado de ingresos” realizar las siguientes funciones:
        - Eliminar: pulsamos sobre el botón con icono de papelera del ingreso que queremos eliminar. Una vez hecho esto, nos aparece un mensaje de confirmación que debemos confirmar en caso de que deseemos borrar el ingreso.
        - Añadir: pulsamos sobre el botón de “Añadir” y navegamos de forma automática a la pantalla de detalle del ingreso. En ella debemos rellenar el formulario que se nos presenta y pulsar una vez finalizado, sobre el botón “Guardar”. En caso de éxito se muestra un mensaje con fondo verde y en caso contrario, un mensaje de error con fondo rojo.

        - Editar: pulsamos sobre el botón con icono de lápiz del ingreso que queremos editar. Una vez hecho esto, navegamos de forma automática a la pantalla de detalle del ingreso donde podemos modificar el campo que deseemos editar.

- Objetivos (rol usuario)
    1. Accedemos a la opción del menú Objetivos.
    2. Búsqueda. Se pueden buscar objetivos filtrando por año.
    3. Eliminar: pulsamos sobre el botón con icono de papelera del objetivo que queremos eliminar. Una vez hecho esto, nos aparece un mensaje de confirmación que debemos confirmar en caso de que deseemos borrar el objetivo.
    4. Añadir: pulsamos sobre el botón de “Añadir” y navegamos de forma automática a la pantalla de detalle del objetivo. En ella debemos rellenar el formulario que se nos presenta y pulsar una vez finalizado, sobre el botón “Guardar”. En caso de éxito se muestra un mensaje con fondo verde y en caso contrario, un mensaje de error con fondo rojo.
    5. Editar: pulsamos sobre el botón con icono de lápiz del objetivo que queremos editar. Una vez hecho esto, navegamos de forma automática a la pantalla de detalle del objetivo donde podemos modificar el campo que deseamos editar.
- Añadir plantillas (rol usuario)
    1. Accedemos a la opción del menú Plantillas.
    2. Seleccionamos dentro de la sección “Creador de plantillas”, la opción “Añadir plantilla”.
    3. Seleccionamos un año y un mes y pulsamos sobre el botón “Añadir plantilla”.
    4. En caso de éxito se muestra un mensaje con fondo verde y en caso contrario, un mensaje de error con fondo rojo.
- Copiar plantillas (rol usuario)
    1. Accedemos a la opción del menú Plantillas.
    2. Seleccionamos dentro de la sección “Creador de plantillas” la opción “Copiar plantilla”.
    3. Seleccionamos un año y un mes de una plantilla que ya exista, además del año y mes de la nueva plantilla. Una vez hecho esto, pulsamos sobre el botón “Copiar plantilla”.
    4. En caso de éxito se muestra un mensaje con fondo verde y en caso contrario, un mensaje de error con fondo rojo.
- Editar plantillas (rol usuario)
    1. Accedemos a la opción del menú Plantillas.
    2. Eliminar plantilla: pulsamos sobre el botón con el icono de una papelera de aquella plantilla que deseemos eliminar. A continuación, nos aparece un mensaje de confirmación que debemos aceptar si queremos eliminar la plantilla.
    3. Editar plantilla: pulsamos sobre el botón con el icono de un lápiz de aquella plantilla que deseamos editar. Las plantillas se organizan por años y mediante el combo “Seleccione un año” podemos elegir la plantilla que deseemos. A continuación, navegamos de forma automática a la pantalla de la plantilla donde podemos:

- Añadir recursos a la plantilla: seleccionamos aquellas ganancias y gastos que hayamos creado dentro de la plantilla en la que nos encontremos. Una vez añadido los recursos deseados y rellenado el valor de cada uno de ellos, pulsamos sobre el botón “Guardar”. Los recursos de las plantillas podemos además marcarlos como pagados y eliminarlos de la propia plantilla pulsando sobre el botón con icono de papelera.

19. Proyección a futuro

Información, predicciones y sugerencias acerca de ampliaciones a futuro del trabajo, y/o lista de mejoras a realizar en hipotéticas futuras versiones del servicio/aplicación.

- Establecer un sistema de logros basado en insignias y trofeos que incentive a los usuarios a cumplir objetivos y retos de ahorros.
- Permitir a los usuarios crear diferentes fines de ahorros donde puedan ir destinando los ahorros que van generando. Ejemplo de ello puede ser crear un fin de ahorro para un viaje donde el usuario va destinando parte de sus ahorros.
- Introducir diferentes métodos de ahorros dentro del panel de administradores que sean seleccionables en las plantillas de los usuarios para que puedan elegir el método de ahorro que mejor se ajuste a sus necesidades.
- Una nueva sección dentro del panel de usuarios donde este introduzca un nuevo gasto y el sistema se encargue en base a los ingresos que recibe, recomendar o no afrontar ese nuevo gasto.
- Soporte para multitud de idiomas.
- Aunque los usuarios móviles puedan acceder a la aplicación mediante la web y verla de forma adecuada al dispositivo en que se encuentran, la idea es traspasar la aplicación web a una aplicación móvil más accesible para los usuarios.
- Asistente a modo de tutorial que ayude a los nuevos usuarios a entender el funcionamiento de la aplicación.

20. Presupuesto

Se presentan las siguientes tablas donde se definen el presupuesto estimado de la aplicación. Para el presupuesto del equipo humano se ha estimado que se ha trabajado durante 4 horas cada día por un precio de 35 €/hora.

EQUIPO HUMANO			
Tarea	**Tiempo**	**Horas**	**Importe**
Requisitos	10 días	40	1400
Diseño	21 días	84	2940
Implementación	28 días	112	3920
Verificación	20 días	80	2800
Total	79 días	316	11060

Tabla 4: Presupuesto equipo humano

Para el equipamiento técnico se ha estimado el coste necesario para poder mantener nuestra aplicación en un servidor disponible para cualquier usuario a lo largo de un año y del equipo utilizado para desarrollar la aplicación.

EQUIPO TÉCNICO	
Tarea	**Importe**
Dominio (.es y .com)	29.9
Hosting	60
Ordenador	1200
Total	1289.9

Tabla 5: Presupuesto equipo técnico

El presupuesto total del proyecto es 12349.9 euros.

21. Análisis de mercado

En el mercado existen diferentes aplicaciones destinadas a la administración de los ahorros, ejemplo de ellos son las siguientes aplicaciones:

- Fintonic: centraliza dentro de la aplicación todos los bancos y tarjetas del usuario y su función es permitir controlar los ingresos y gastos y, por otro lado, te genera un score financiero para poder saber a qué productos puede acceder el usuario.
- Coinscrap: tiene un método único y diferente de redondeos que se destinan a los ahorros además de crear objetivos de ahorro como pueden ser un viaje o un plan de jubilación.
- Desafío 52 semanas: propone un reto de ahorrar dinero semanalmente de forma progresiva durante un año.
- Acorns: tiene un funcionamiento similar al de Coinscrap, pero los ahorros van destinados a la cartera bursátil que elija el usuario.
- Betterment: es una aplicación que te guía en el proceso de inversión en función de tus intereses y del tipo de cartera que el usuario quiera probar suerte.
- Yudonpay: está más enfocada en tarjetas de fidelización.

Todas estas aplicaciones tienen como objetivo que el usuario incremente sus ahorros y beneficios, por ello se convierten en opciones viables que un usuario puede escoger el día de mañana para llevar a cabo la gestión de su economía, pero lo que se busca con la aplicación que desarrollamos en este proyecto es permitir al usuario sin la necesidad del uso de sus cuentas bancarias o tarjetas que llevan a muchos usuarios a la desconfianza de introducir datos importantes en cuentas de terceros, que tenga la posibilidad de llevar a cabo una gestión manual de sus propios gastos y ahorros, poniéndose las metas que desee y además, proponiéndole un método de ahorro diferente al que hacen otras aplicaciones, basado en la regla 50/30/20, una de las técnicas de ahorro más conocidas y aconsejadas, cuyo objetivo es administrar el dinero en gastos primarios, secundarios y ahorros, que ayuda al usuario ahorrar de una forma muy sencilla y que le permite conocer sus gastos en todo momento.

22. Conclusión/-es

El trabajo final de máster me ha permitido afianzar todos los conocimientos recopilados en el máster, ya que he podido trabajar elementos de cada una de las asignaturas de este. Estoy muy satisfecho con poder haber realizado un trabajo de estas características, porque creo que cada capítulo de la presente memoria permite conocer las dimensiones que tiene desarrollar una aplicación web.

En cuanto a los conocimientos adquiridos, Angular es una herramienta con la que ya he trabajado durante tres años y que gracias a este trabajo he podido seguir afianzando conceptos y aprendiendo nuevas cosas. Con respecto a Laravel, he podido aprender muchas cosas debido que desconocía más este framework, dado que sólo he podido utilizarlo en una ocasión y ha sido dentro del propio máster.

El resultado obtenido me ha dejado muy satisfecho, ya que he podido realizar todas las funcionalidades que desde un principio he tenido en mente, además de poder acompañarlo de una memoria donde he podido tocar otros puntos como prototipos, planificación o presupuestos entre otros, para poder hacer en global un proyecto completo y avanzando para tratarse de una versión inicial con idea de ser escalable y fiable.

Finalmente hay que comentar que la planificación del proyecto se ha podido cumplir con los objetivos fijados en los tiempos marcados, gracias a la metodología en cascada que se ha utilizado y que me ha dejado bastante satisfecho.

Anexo 1. Entregables del proyecto

Lista de archivos entregados y su descripción:

- Documentación
  - PAC_FINAL_mem_ AlvarezHernandez_OmarAdolfo: esta memoria.
  - PAC_FINAL_prs_ AlvarezHernandez_OmarAdolfo: presentación escrita-visual.
  - PAC_FINAL_informe_ AlvarezHernandez_OmarAdolfo: informe de autoevaluación.
- Proyecto
  - PAC_FINAL_prj_AlvarezHernandez_OmarAdolfo: proyecto de Laravel. Dentro de la carpeta resources/frontend podemos encontrar el proyecto de Angular.
- PAC_FINAL_vid_AlvarezHernandez_OmarAdolfo: presentación en vídeo del proyecto entregado en la herramienta @Presenta de la Universidad.

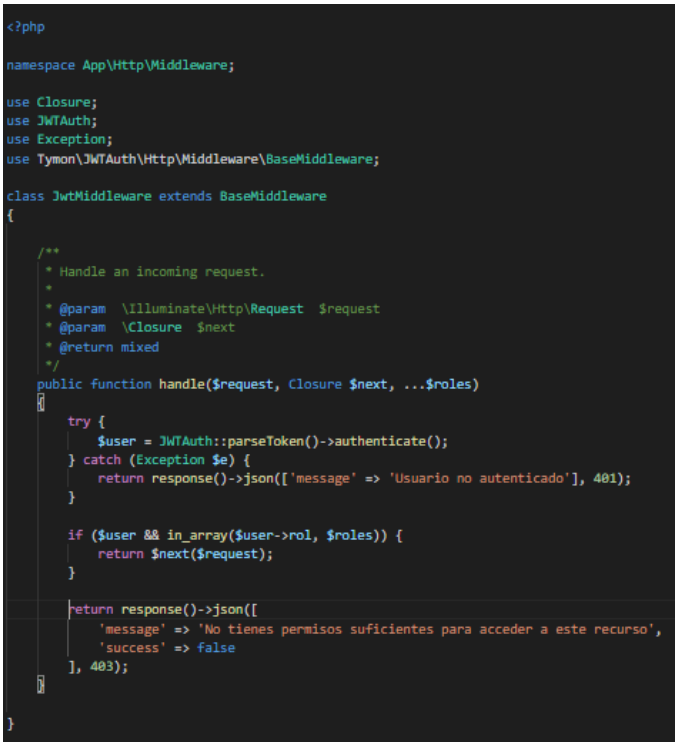
Anexo 2. Código fuente (extractos)

Fichero de rutas utilizado por Laravel donde podemos ver que las rutas están securizadas por rol a través de un middleware que comprueba el rol y la validez del token.

```php
Route::post('/api/register', 'AuthController@register');
Route::post('/api/login', 'AuthController@login');
Route::get('/api/get-token-email', 'AuthController@getTokenByEmail');
Route::post('/api/reset-password', 'AuthController@resetPassword');

Route::group(['middleware' => 'jwt.verify:ADMIN,USER'], function() {
    Route::get('/api/user', 'AuthController@getAuthenticatedUser');
    Route::get('api/tipos-gastos/all', 'TipoGastoController@findAll');
    Route::get('api/tipos-ganancias/all', 'TipoGananciaController@findAll');
    Route::get('api/buenas-practicas/all', 'BuenaPracticaController@findAll');
    Route::post('api/change-password', 'AuthController@changePassword');
});

Route::group(['middleware' => 'jwt.verify:USER'], function() {
    Route::resource('api/gastos', 'GastoController');
    Route::get('api/gastos-primarios', 'GastoController@findAllPrimaryMonthlyExpenses');
    Route::get('api/gastos-secundarios', 'GastoController@findAllSecondaryMonthlyExpenses');
    Route::resource('api/pagos', 'PagoController');
    Route::resource('api/meta-ahorros', 'MetaAhorroController');
    Route::resource('api/ganancias', 'GananciaController');
    Route::get('api/ganancias-mensuales', 'GananciaController@findAllProfits');
    Route::resource('api/ingresos', 'IngresoController');
    Route::get('api/ahorros-anuales', 'AhorroController@savingsAndExpensesByYear');
    Route::get('api/ahorros/cuenta-ahorro', 'AhorroController@cuentaAhorro');
    Route::get('api/ahorros/recordatorios-anuales', 'AhorroController@recordatoriosAnuales');
    Route::resource('api/plantillas', 'PlantillaController');
    Route::get('api/plantilla-actual', 'PlantillaController@getTemplateMonthActual');
    Route::post('api/plantilla-clone', 'PlantillaController@clone');
    Route::get('api/estadisticas', 'EstadisticasController@getStats');
});

Route::group(['middleware' => 'jwt.verify:ADMIN'], function() {
    Route::resource('api/tipos-gastos', 'TipoGastoController');
    Route::resource('api/tipos-ganancias', 'TipoGananciaController');
    Route::resource('api/usuarios', 'UserController');
    Route::resource('api/buenas-practicas', 'BuenaPracticaController');
});
```

Ilustración 45: Fichero web.php

Middleware que se encarga de comprobar si un usuario puede acceder a una determinada ruta.



```php
<?php

namespace App\Http\Middleware;

use Closure;
use JWTAuth;
use Exception;
use Tymon\JWTAuth\Http\Middleware\BaseMiddleware;

class JwtMiddleware extends BaseMiddleware
{

    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next, ...$roles)
    {
        try {
            $user = JWTAuth::parseToken()->authenticate();
        } catch (Exception $e) {
            return response()->json(['message' => 'Usuario no autenticado'], 401);
        }

        if ($user && in_array($user->rol, $roles)) {
            return $next($request);
        }

        return response()->json([
            'message' => 'No tienes permisos suficientes para acceder a este recurso',
            'success' => false
        ], 403);
    }

}
```

Ilustración 46: Middleware JWT

Ejemplo de servicio en Angular donde se realizan llamadas a la API

```typescript
export class ExpenseService {

  public url: string;

  constructor(
    private http: HttpClient
  ) {
    this.url = environment.url + 'gastos';
  }

  get(form: any, sortable?: string, sortOrder?: number, page?: number): Observable<Expense[]> {
    let params: string[] = [];
    let query = '';
    if (form.nombre) {
      params.push('nombre=' + form.nombre);
    }
    if (form.tipo_gasto) {
      params.push('tipo_gasto=' + form.tipo_gasto.id);
    }
    if (sortable) {
      params.push('sortable=' + sortable);
      if (sortOrder) {
        params.push(sortOrder === 1 ? 'orderBy=asc' : 'orderBy=desc');
      }
    }
    if (page) {
      params.push('page=' + page);
    }
    if (params.length > 0) {
      query = '?' + params.join('&');
    }
    return this.http.get<Expense[]>(this.url + query);
  }

  findAllPrimaryMonthlyExpenses(): Observable<any> {
    return this.http.get<Expense[]>(this.url + '-primarios');
  }

  findAllSecondaryMonthlyExpenses(): Observable<any> {
    return this.http.get<Expense[]>(this.url + '-secundarios');
  }

  find(id: number) {
    return this.http.get<Response>(this.url + '/' + id);
  }

  add(expense: Expense): Observable<Response> {
    return this.http.post<Response>(this.url, expense);
  }
```

Ilustración 47: Servicio en Angular

Ejemplo de controlador en Laravel de la API de la aplicación

```php
use Illuminate\Http\Request;
use App\Gasto;
use Validator;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Str;
use JWTAuth;
use DB;

class GastoController extends Controller
{

    public function findAllPrimaryMonthlyExpenses() {
        $gastos_primarios = Gasto::select('gastos.*')
        ->join('tipos_gastos', 'gastos.id_tipo_gasto', '=', 'tipos_gastos.id')
        ->join('usuarios', 'gastos.id_usuario', '=', 'usuarios.id')
        ->where('usuarios.id', '=', JWTAuth::user()->id)
        ->where('tipos_gastos.valor', '=', 'Mensuales Primarios')
        ->where(static function ($query) {
            $now = date('Y-m-d');
            $query->where('gastos.fecha_fin', '>=', $now)
            ->orWhere('gastos.fecha_fin', '=', null);
        })
        ->get();
        return response()->json(array(
            'data' => $gastos_primarios
        ), 200);
    }

    public function findAllSecondaryMonthlyExpenses() {
        $gastos_secundarios = Gasto::select('gastos.*')
        ->join('tipos_gastos', 'gastos.id_tipo_gasto', '=', 'tipos_gastos.id')
        ->join('usuarios', 'gastos.id_usuario', '=', 'usuarios.id')
        ->where('usuarios.id', '=', JWTAuth::user()->id)
        ->where('tipos_gastos.valor', '=', 'Mensuales Secundarios')
        ->where(static function ($query) {
            $now = date('Y-m-d');
            $query->where('gastos.fecha_fin', '>=', $now)
            ->orWhere('gastos.fecha_fin', '=', null);
        })
        ->get();
        return response()->json(array(
            'data' => $gastos_secundarios
        ), 200);
    }
```

Ilustración 48: Ejemplo controlador Laravel

```php
public function index(Request $request)
{
    $orderBy = !is_null($request['orderBy']) ? $request['orderBy'] : 'asc';
    $sortable = !is_null($request['sortable']) ? $request['sortable'] : 'id';

    $gastos = new Gasto;

    if ($request->has('nombre')) {
        $gastos = $gastos
            ->where('nombre', $request['nombre'])
            ->orWhere('nombre', 'like', '%' . $request['nombre'] . '%');
    }

    if ($request->has('tipo_gasto')) {
        $gastos = $gastos
            ->where('id_tipo_gasto', $request['tipo_gasto'])
            ->orWhere('id_tipo_gasto', 'like', '%' . $request['tipo_gasto'] . '%');
    }

    $gastos = $gastos
            ->orderBy($sortable, $orderBy)
            ->where('id_usuario', '=', JWTAuth::user()->id)
            ->paginate(10);

    return response()->json(array(
        'data' => $gastos
    ), 200);
}

public function show($id) {
    $gasto = Gasto::find($id);
    if (!is_null($gasto) && $gasto->id_usuario === JWTAuth::user()->id) {
        return response()->json(array(
            'data' => $gasto->load('pagos')
        ), 200);
    } else {
        return response()->json([
            'message' => 'El gasto no existe'
        ], 500);
    }
}
```

Ilustración 49: Ejemplo controlador Laravel

```php
public function store(Request $request)
{
    $validator = Validator::make($request->all(), [
        'nombre' => 'required',
        'id_tipo_gasto' => 'required',
    ]);
    if ($validator->fails()) {
        return response()->json(['message' => 'Validaciones erróneas'], 500);
    }
    $postArray = $request->all();
    $gastos_bd = Gasto::where('nombre', '=', $postArray['nombre'])->where('id_usuario', '=', JWTAuth::user()->id)->first();
    if (is_null($gastos_bd)) {
        $gasto = new Gasto();
        $gasto->nombre = $postArray['nombre'];
        $gasto->id_tipo_gasto = $postArray['id_tipo_gasto']['id'];
        $gasto->id_usuario = JWTAuth::user()->id;
        if (!is_null($postArray['fecha_fin'])) {
            $gasto->fecha_fin = date('Y-m-d h:i:s', strtotime($postArray['fecha_fin']));
        } else {
            $gasto->fecha_fin = null;
        }
        if ($request->has('flexible')) {
            $gasto->flexible = $postArray['flexible'];
        } else {
            $gasto->flexible = 0;
        }
        $gasto->save();
        return response()->json([
            'message' => 'Se ha creado un nuevo gasto',
            'data' => $gasto->load('id_tipo_gasto')
        ], 200);
    } else {
        return response()->json([
            'message' => 'El gasto con ese nombre ya existe'
        ], 500);
    }
}
```

Ilustración 50: Ejemplo 3 controlador Laravel

Anexo 3. Librerías/Código externo utilizado

Las bibliotecas externas utilizadas han sido:

- Bootstrap: es una biblioteca multiplataforma para el diseño de aplicaciones web. Esta librería aporta muchos componentes y utilidades que aceleran el proceso de diseñar las aplicaciones web. Se ha utilizado dentro de la aplicación sobre todo para utilizar el sistema de rejillas que aporta para llevar a cabo la responsividad del sitio web. El comando para instalar la librería es: **npm install –save bootstrap jquery**
- Fontawesome: es una librería que proporciona multitud de iconos que han sido utilizados dentro de nuestra aplicación en las opciones del menú superior y lateral como en los botones. El comando para instalar la librería es: **npm install @fortawesome/fontawesome-free**
- Google Fonts: es un repositorio de fuentes de web de donde se ha obtenido la fuente Roboto como font-family de nuestra aplicación.
- Primeng: es una librería de componentes de Angular muy popular de donde se ha obtenido componentes muy potentes y con mucha capacidad de personalizar a nuestro gusto. De esta librería se ha utilizado: tablas, mensajes, calendarios, elementos de formularios y tooltips. Además, se ha utilizado para la sección de estadísticas acompañado de la descarga de la librería chart.js.
- Librerías JWT: desde el lado de Laravel se ha utilizado tymon/jwt-auth para llevar a cabo toda la generación de tokens, middlewares de autenticación y control de rutas por roles. Y para la parte de Angular, se ha utilizado jwt-decode para poder decodificar los tokens y el rol del usuario autenticado y, por otro lado, auth0/angular-jwt para poder obtener la validez del token.
- ng2-pdf-viewer: esta librería permite visualizar archivos pdf dentro de Angular. Se ha utilizado para añadir las guías de usuario a los apartados de “Ayuda”.
- Angular material: se ha utilizado la librería de componentes de Angular para hacer uso del componente menú en aquellos dispositivos más pequeños con el fin de aprovechar mejor la pantalla y tener un menú desplegable de hamburguesa más accesible.

Anexo 4. Guía de usuario

Las guías de usuario la dividiremos en rol de administrador y rol de usuario. Cada una de las guías, son accesibles desde la opción del menú “Ayuda”, una vez hayamos iniciado sesión.

Antes de comenzar con cada guía, comentaremos el proceso común para ambos roles, que consiste en el proceso donde tenemos que iniciar sesión.

Guía común

La pantalla principal de la aplicación es la página de bienvenida y desde ella puede acceder al inicio de sesión o al registro.

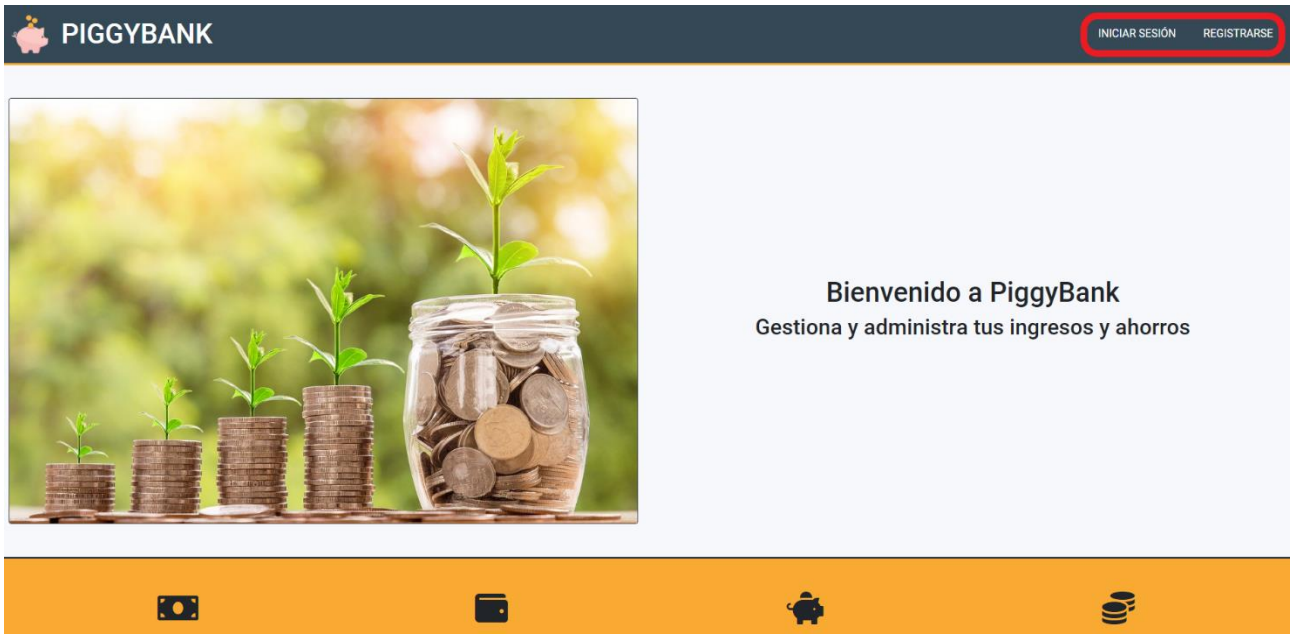


Ilustración 51: Página de bienvenida

En la pantalla de inicio de sesión, debe introducir su correo electrónico y contraseña.

Ilustración 52: Pantalla de inicio de sesión

En caso de no poder iniciar sesión en la aplicación, tienes dos opciones:

- Si no tienes un usuario registrado en la aplicación, pinche sobre “Crea una cuenta” y navegará a la pantalla de registro donde debe rellenar el formulario y utiliza un correo que aún no haya sido dado de alta en la aplicación. Una vez creado el usuario con rol de usuario, navegará de forma automática a la pantalla de inicio de sesión.

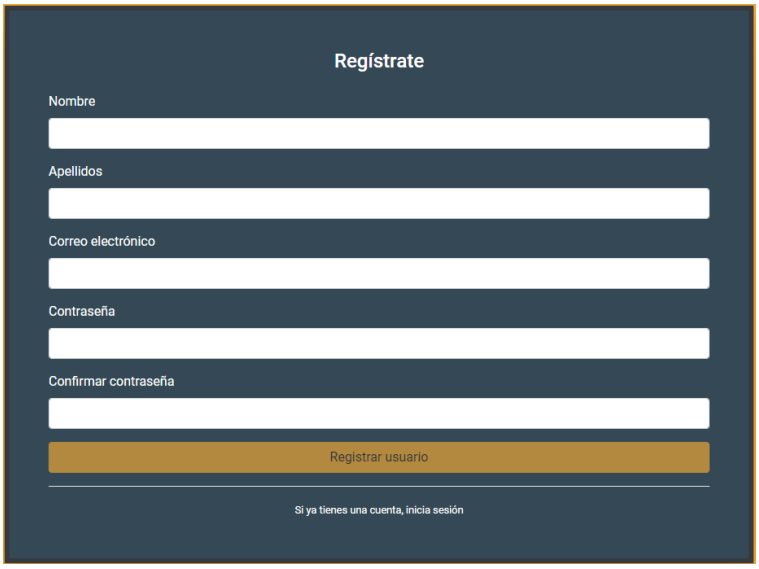


Ilustración 53: Pantalla de registro

- Si no recuerda la contraseña, pinche sobre “¿Olvidaste tu contraseña?” y navegará a la pantalla de recuperar contraseña donde debe introducir el correo electrónico que tiene registrado en la aplicación, donde recibirá un token que debe introducir en el formulario junto a la nueva contraseña.

Ilustración 54: Pantalla para recuperar contraseña

Una vez iniciada sesión, puede acceder a su perfil de usuario, donde puede ver sus datos además de poder cambiar su contraseña.

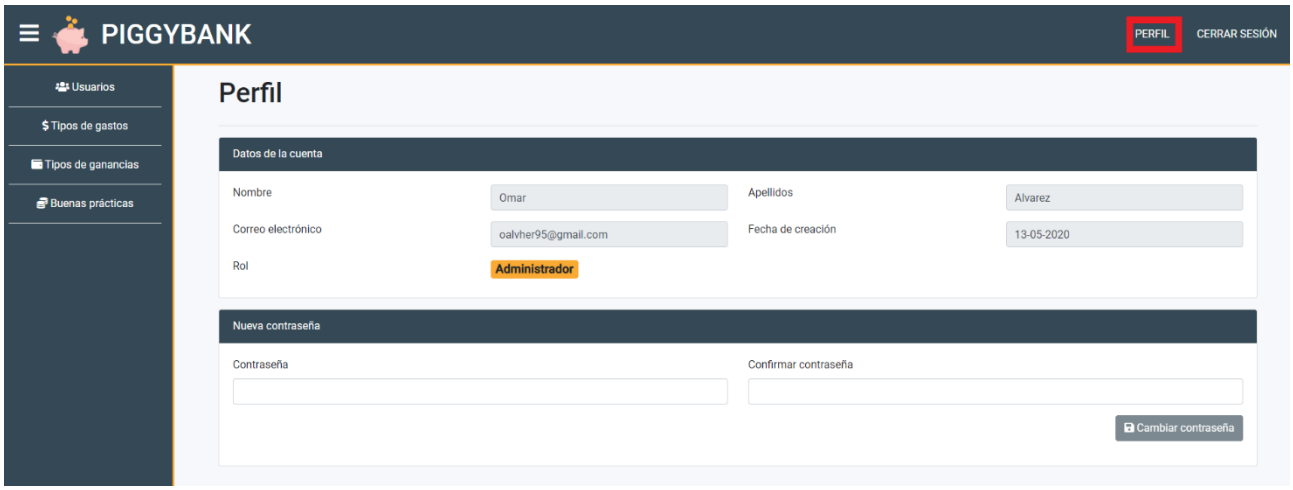


Ilustración 55: Perfil de usuario

Por último, en cuanto a elementos comunes a ambos roles, si aprieta sobre cerrar sesión puede cerrar su sesión y saldrá de la aplicación.

Guía con el rol Administrador

Las pantallas a las que puede acceder siendo administrador son:

- Usuarios: puede visualizar los usuarios registrados en la aplicación. Las acciones disponibles son:
  - Búsqueda. Se pueden buscar usuarios filtrando por nombre, apellidos, correo electrónico o rol.
  - Añadir: pulse el botón de “Añadir”, rellene el formulario que aparece y pulse sobre el botón "Guardar”. Los usuarios creados recibirán un correo con su contraseña.

- Editar: pulse sobre el botón con el icono de un lápiz del usuario que quiera editar. Una vez hecho esto, navegará de forma automática a la pantalla de detalle del usuario donde puede modificar sus campos. Cuando se hayan modificado los campos, pulse sobre el botón “Guardar”.
- Eliminar: pulse sobre botón con el icono de una papelera del usuario que quiera eliminar. Una vez hecho esto, le aparece un mensaje de confirmación que debe confirmar en caso de que quiera borrar el usuario.

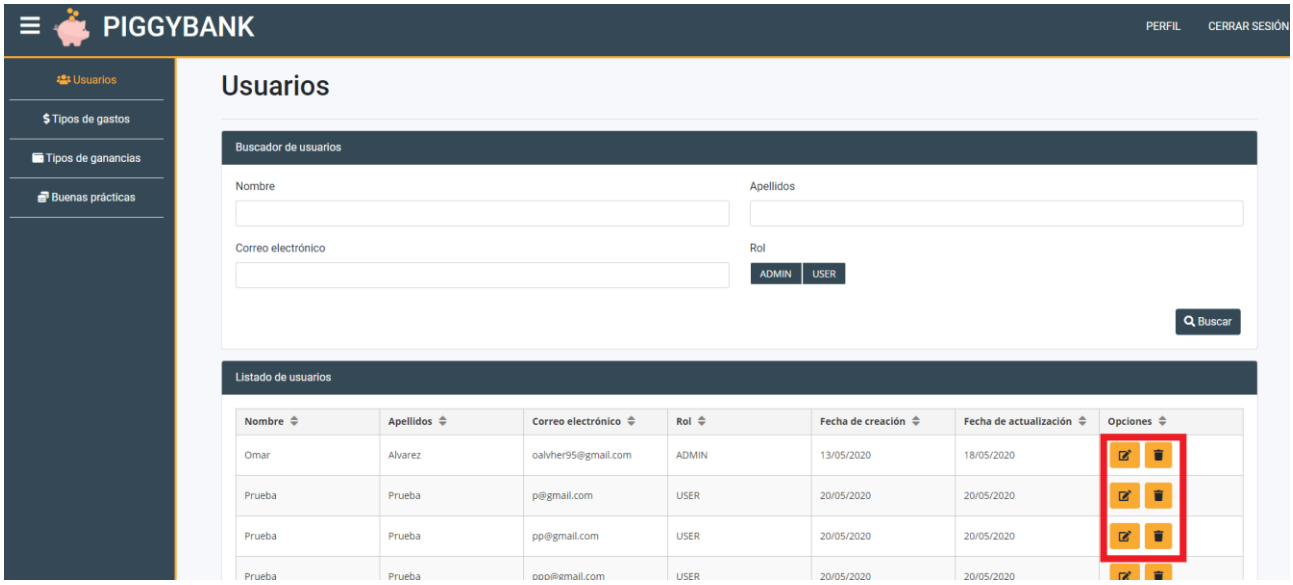


Ilustración 56: Pantalla de usuarios

- Tipos de gastos: puede visualizar todos los tipos de gastos a los que los usuarios tendrá disposición a la hora de crear un gasto en concreto. Aquellos tipos de gastos creados por un usuario sólo pueden ser editados y eliminados por el mismo usuario. Las acciones disponibles son:
  - Búsqueda. Se pueden buscar tipos filtrando por el campo valor y descripción.
  - Añadir: pulse el botón de “Añadir”, rellene el formulario que aparece y pulse sobre el botón "Guardar”.
  - Editar: pulse sobre el botón con el icono de un lápiz del tipo de gasto que quiera editar. Una vez hecho esto, navegará de forma automática a la pantalla de detalle del tipo de gasto donde puede modificar sus campos. Cuando se hayan modificado los campos, pulse sobre el botón “Guardar”.
  - Eliminar: pulse sobre botón con el icono de una papelera del tipo de gasto que quiera eliminar. Una vez hecho esto, le aparece un mensaje de confirmación que debe confirmar en caso de que quiera borrar el tipo de gasto.

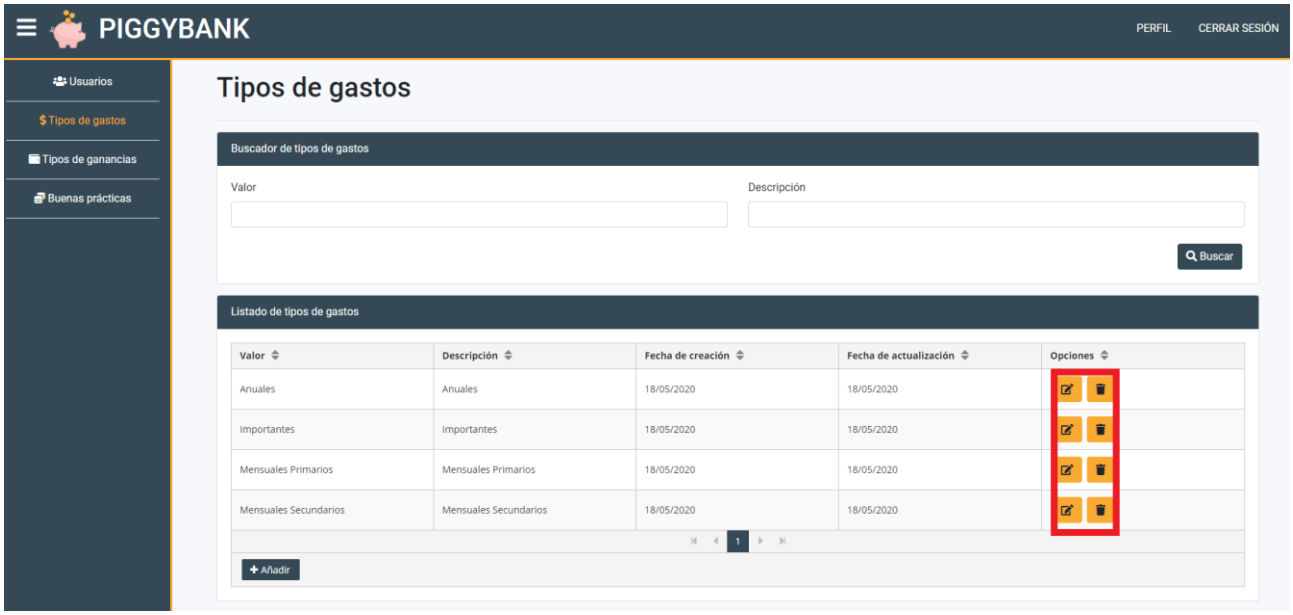


Ilustración 57: Pantalla de tipos de gastos

- Tipos de gastos: puede visualizar todos los tipos de ganancias a los que los usuarios tendrá disposición a la hora de crear una ganancia en concreto. Aquellos tipos de ganancias creados por un usuario sólo pueden ser editados y eliminados por el mismo usuario. Las acciones disponibles son:
    - Búsqueda. Se pueden buscar tipos filtrando por el campo valor y descripción.
    - Añadir: pulse el botón de “Añadir”, rellene el formulario que aparece y pulse sobre el botón "Guardar”.
    - Editar: pulse sobre el botón con el icono de un lápiz del tipo de ganancia que quiera editar. Una vez hecho esto, navegará de forma automática a la pantalla de detalle del tipo de ganancia donde puede modificar sus campos. Cuando se hayan modificado los campos, pulse sobre el botón “Guardar”.
    - Eliminar: pulse sobre botón con el icono de una papelera del tipo de ganancia que quiera eliminar. Una vez hecho esto, le aparece un mensaje de confirmación que debe confirmar en caso de que quiera borrar el tipo de ganancia.

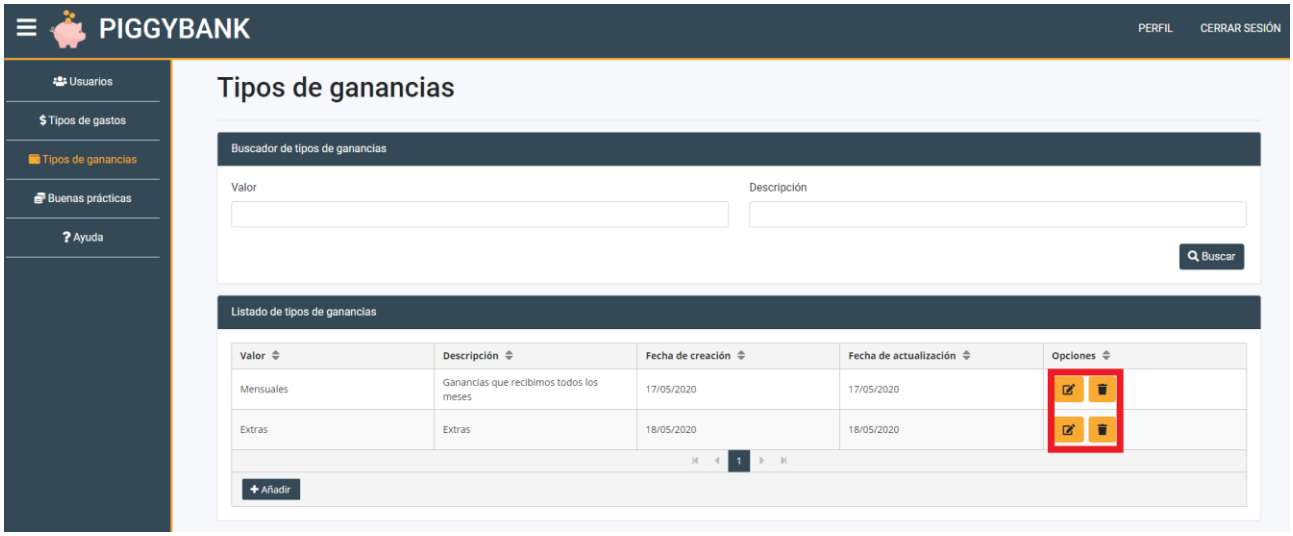


Ilustración 58: Pantalla de tipos de ganancias

- Buenas prácticas: puede visualizar todas las buenas prácticas que serán aplicadas dentro de la gestión mensual de cada usuario. Dichos elementos permiten saber al usuario si su gestión de sus gastos está siendo correcta. Las acciones disponibles son:
    - Búsqueda. Se pueden buscar buenas prácticas filtrando por palabra clave
    - Añadir: pulsamos el botón de “Añadir”, rellenamos el formulario que nos aparece y pulsamos sobre el botón "Guardar”.
    - Editar: pulse sobre el botón con el icono de un lápiz de la buena práctica que quiera editar. Una vez hecho esto, navegará de forma automática a la pantalla de detalle de la buena práctica donde puede modificar sus campos. Cuando se hayan modificado los campos, pulse sobre el botón “Guardar”.
    - Eliminar: pulse sobre botón con el icono de una papelera de la buena práctica que quiera eliminar. Una vez hecho esto, le aparece un mensaje de confirmación que debe confirmar en caso de que quiera borrar la buena práctica.

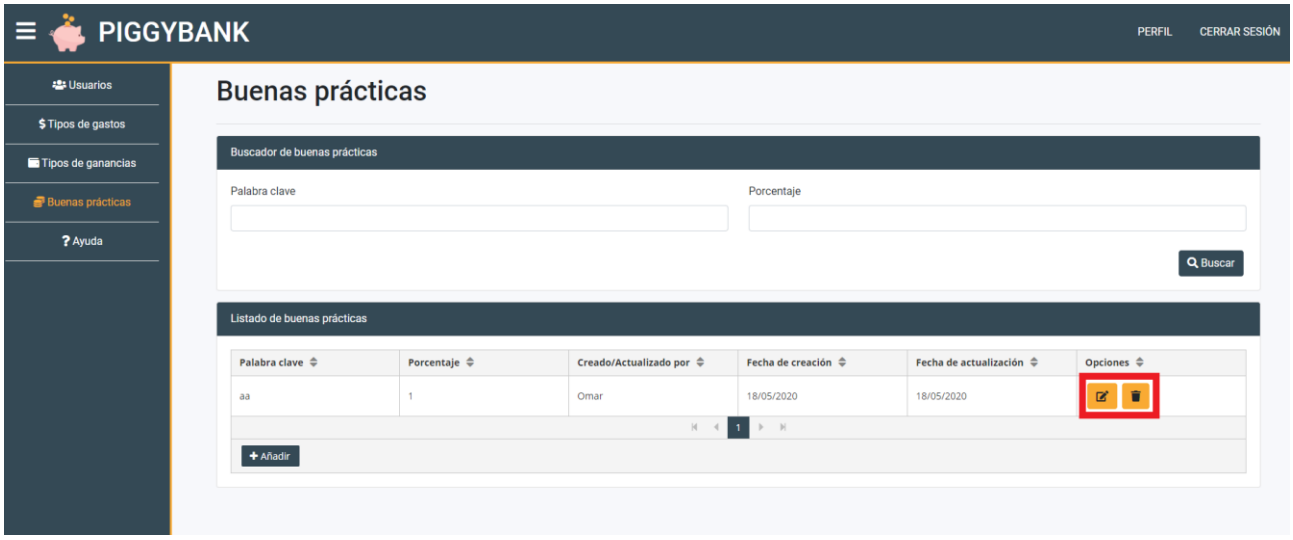


Ilustración 59: Pantalla de buenas prácticas

Guía con el rol Usuario

Las pantallas a las que puede acceder siendo usuario son:

- Escritorio. En esta pantalla puede visualizar:
    1. Cuenta de gastos: dinero disponible del mes actual.
    2. Cuenta ahorros: total de ahorros acumulados tras realizar la diferencia entre ingresos y pagos sin contar los del mes actual.
    3. Total de gastos pagados: porcentaje de los gastos mensuales que ya ha pagado frente a todos los gastos mensuales que tiene.
    4. Recordatorios mensuales: aparecen todos los gastos mensuales que aún le quedan por pagar. Aquellos gastos marcados como flexibles, no aparece en esta sección.
    5. Recordatorios anuales: aparece gastos de tipo anual que con un período previo a un año para recordar gastos que se pueden realizar en el mes en cuestión o el mes siguiente con el fin de recordarle próximos pagos.

Ilustración 60: Página de escritorio

- Gastos: en esta pantalla puede añadir todos los gastos que tienes como usuario. Dichos gastos, los puede marcar del tipo que desee teniendo en cuenta que aquellos de tipo mensual, deben ser gestionados en la pantalla de plantillas. Por otro lado, aquellos marcado como anuales podrán serle recordado dentro del escritorio. Por último, los importantes no serán incluidos dentro de sus metas de ahorro. Las acciones disponibles son:
    1. Búsqueda. Se pueden buscar gastos filtrando por nombre o tipo de gasto.
    2. Eliminar: pulse sobre el botón con icono de papelera del gasto que quiera eliminar. Una vez hecho esto, aparece un mensaje de confirmación que debe confirmar en caso de que quiera borrar el gasto.
    3. Añadir: pulse sobre el botón de “Añadir” y navegará de forma automática a la pantalla de detalle del gasto. En ella debe rellenar el formulario que se presenta y pulsar una vez finalizado, sobre el botón “Guardar”. En caso de éxito se muestra un mensaje con fondo verde y en caso contrario, un mensaje de error con fondo rojo. El campo fecha fin permite indicar cuando el gasto va a finalizar, de tal modo que, no será incluido en plantillas con fecha posterior a su finalización. Por otro lado, el campo flexible permite evitar que un gasto sea recordado dentro del escritorio. Ambos campos aparecen con gastos de tipo mensual.
    4. Editar: pulse sobre el botón con icono de lápiz del gasto que quiera editar. Una vez hecho esto, navegará de forma automática a la pantalla de detalle del gasto donde puede modificar el campo del gasto que desea editar. Además, puede añadir nuevos pagos al gasto en cuestión.

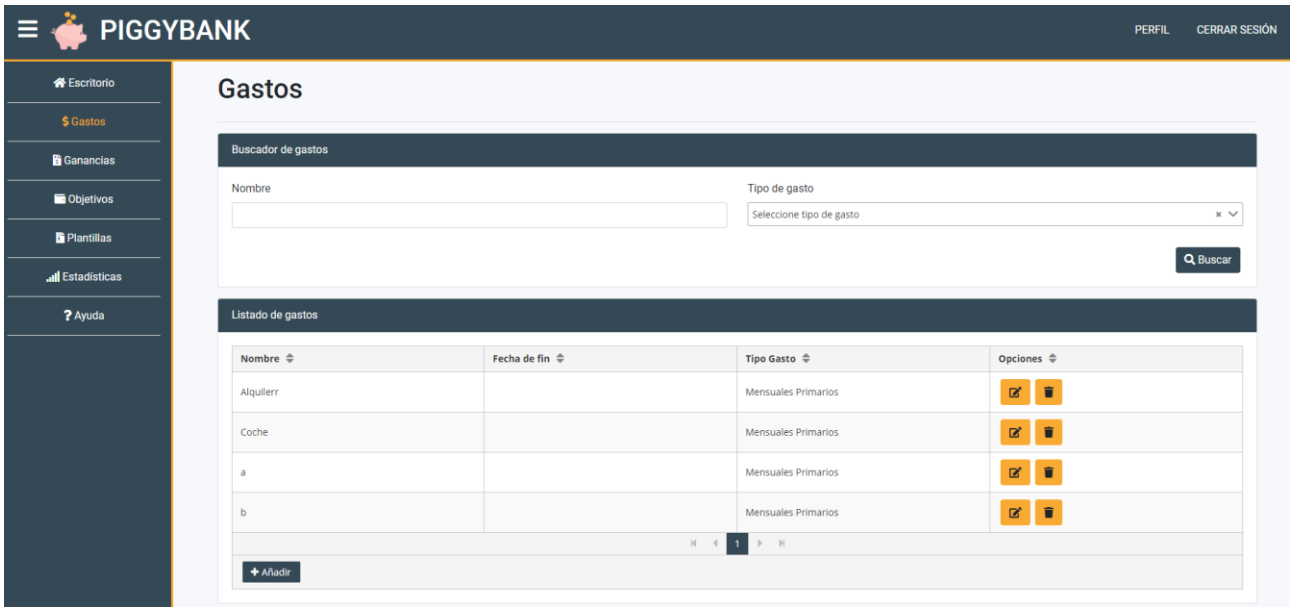


Ilustración 61: Pantalla de gastos

- Ganancias: en esta pantalla puede añadir todas las ganancias que recibe como usuario. Las acciones disponibles son:
    1. Búsqueda. Puede buscar ganancias filtrando por nombre o tipo de ganancia.
    2. Eliminar: pulsamos sobre el botón con icono de papelera de la ganancia que queremos eliminar. Una vez hecho esto, nos aparece un mensaje de confirmación que debemos confirmar en caso de que deseemos borrar la ganancia.
    3. Añadir: pulse sobre el botón de “Añadir” y navegará de forma automática a la pantalla de detalle de la ganancia. En ella debe rellenar el formulario que se presenta y pulsar una vez finalizado, sobre el botón “Guardar”. En caso de éxito se muestra un mensaje con fondo verde y en caso contrario, un mensaje de error con fondo rojo.
    4. Editar: pulse sobre el botón con icono de lápiz de la ganancia que quiera editar. Una vez hecho esto, navegará de forma automática a la pantalla de detalle de la ganancia donde puede modificar el campo de la ganancia que desea editar. Además, puede añadir nuevos ingresos a la ganancia en cuestión.

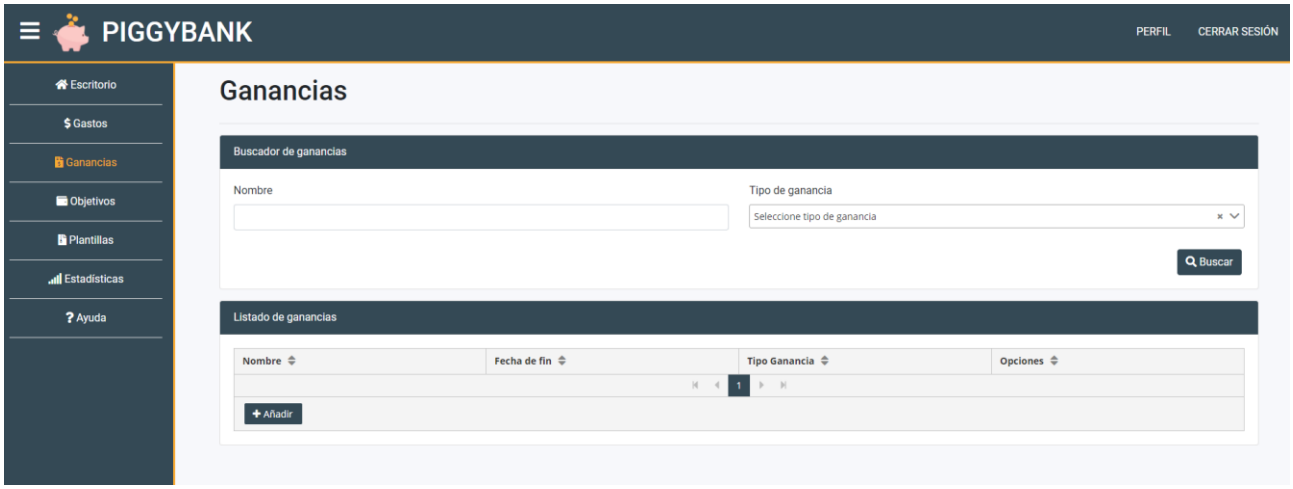


Ilustración 62: Pantalla de ganancias

- Pagos: puede añadir los pagos correspondientes dentro de la pantalla de detalle de cada gasto. Debe tener en cuenta que aquellos gastos de tipo mensual sólo pueden gestionarse a través de la pantalla de “Plantillas”. Las acciones disponibles son:
    1. Eliminar: pulse sobre el botón con icono de papelera del pago que quiera eliminar. Una vez hecho esto, nos aparece un mensaje de confirmación que debe confirmar en caso de que quiera borrar el pago.
    2. Añadir: pulse sobre el botón de “Añadir” y navegará de forma automática a la pantalla de detalle del pago. En ella debe rellenar el formulario que se nos presenta y pulsar una vez finalizado, sobre el botón “Guardar”. En caso de éxito se muestra un mensaje con fondo verde y en caso contrario, un mensaje de error con fondo rojo.

3. Editar: pulsamos sobre el botón con icono de lápiz del pago que quiera editar. Una vez hecho esto, navegará de forma automática a la pantalla de detalle del pago donde puede modificar el campo que deseemos editar.

- Ingresos: puede añadir los ingresos correspondientes dentro de la pantalla de detalle de cada ganancia. Debe tener en cuenta que aquellas ganancias de tipo mensual sólo pueden gestionarse a través de la pantalla de “Plantillas”. Las acciones disponibles son:
    1. Eliminar: pulse sobre el botón con icono de papelera del ingreso que quiera eliminar. Una vez hecho esto, aparece un mensaje de confirmación que debe confirmar en caso de que quiera borrar el ingreso.
    2. Añadir: pulse sobre el botón de “Añadir” y navegará de forma automática a la pantalla de detalle del ingreso. En ella debe rellenar el formulario que se presenta y pulsar una vez finalizado, sobre el botón “Guardar”. En caso de éxito se muestra un mensaje con fondo verde y en caso contrario, un mensaje de error con fondo rojo.
    3. Editar: pulse sobre el botón con icono de lápiz del ingreso que quiera editar. Una vez hecho esto, navegará de forma automática a la pantalla de detalle del ingreso donde puede modificar el campo que quiera editar.
- Objetivos: en esta pantalla puede añadir metas u objetivos de ahorros para cada año con el fin de observar si es capaz de alcanzar dichos objetivos. Cuando ingrese la cantidad para un respectivo año, la aplicación se encarga de indicarle si es superado o no.
    1. Búsqueda. Puede buscar objetivos filtrando por año.
    2. Eliminar: pulse sobre el botón con icono de papelera del objetivo que quiera eliminar. Una vez hecho esto, aparece un mensaje de confirmación que debe confirmar en caso de que quiera borrar el objetivo.
    3. Añadir: pulsamos sobre el botón de “Añadir” y navegará de forma automática a la pantalla de detalle del objetivo. En ella debe rellenar el formulario que se presenta y pulsar una vez finalizado, sobre el botón “Guardar”. En caso de éxito se muestra un mensaje con fondo verde y en caso contrario, un mensaje de error con fondo rojo.
    4. Editar: pulse sobre el botón con icono de lápiz del objetivo que quiera editar. Una vez hecho esto, navegará de forma automática a la pantalla de detalle del objetivo donde puede modificar el campo que quiera editar.

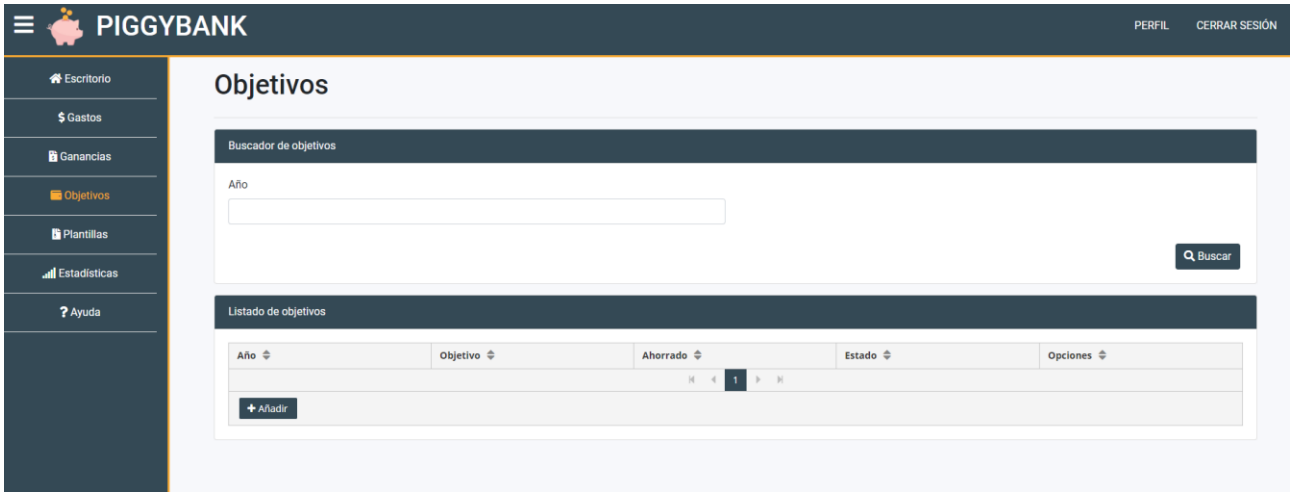


Ilustración 63: Pantalla de objetivos

- Plantilla: en esta pantalla puede gestionar las plantillas que corresponden a los pagos e ingresos de un mes en cuestión. Las acciones disponibles son:
    1. Añadir plantillas
        - Seleccione dentro de la sección “Creador de plantillas” la opción “Añadir plantilla”.
        - Seleccione un año y un mes y pulse sobre el botón “Añadir plantilla”
        - En caso de éxito se muestra un mensaje con fondo verde y en caso contrario, un mensaje de error con fondo rojo.
    2. Copiar plantillas
        - Seleccione dentro de la sección “Creador de plantillas” la opción “Copiar plantilla”.
        - Seleccione un año y un mes de una plantilla que ya exista, además del año y mes de la nueva plantilla. Una vez hecho esto, pulse sobre el botón “Copiar plantilla”.
        - En caso de éxito se muestra un mensaje con fondo verde y en caso contrario, un mensaje de error con fondo rojo.

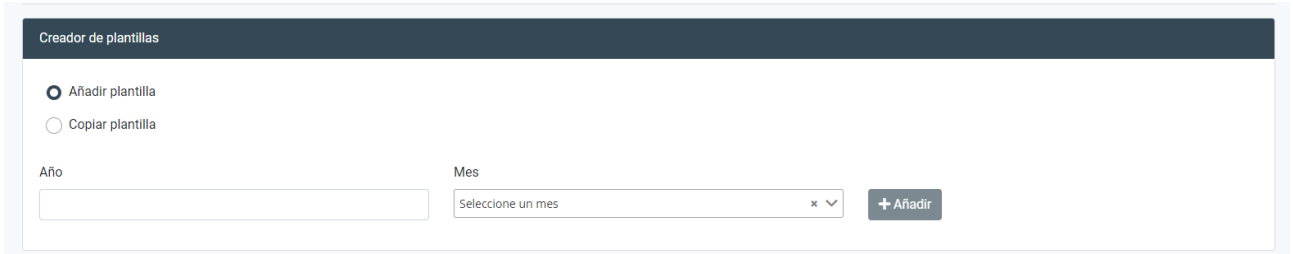


Ilustración 64: Creador de plantillas

    3. Editar plantillas (rol usuario)
        - Eliminar plantilla: pulse sobre el botón con el icono de una papelera de aquella plantilla que quiera eliminar. A continuación, aparece un mensaje de confirmación que debe aceptar si quiere eliminar la plantilla.

- Editar plantilla: pulse sobre el botón con el icono de un lápiz de aquella plantilla que quiera editar. Las plantillas se organizan por años y mediante el combo “Seleccione un año” puede elegir la plantilla que quiera. A continuación, navegará de forma automática a la pantalla de la plantilla donde podemos:
    - Añadir recursos a la plantilla: seleccione aquellas ganancias y gastos que hayamos creado dentro de la plantilla en la que nos encontremos.

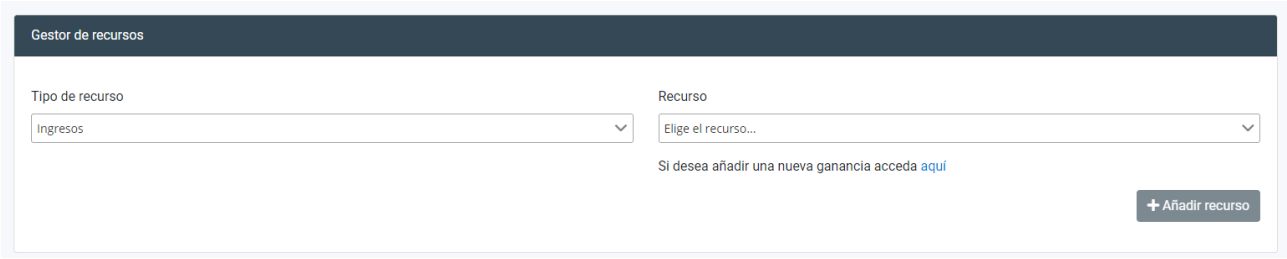


Ilustración 65: Gestor de recursos

- Editar recursos: Una vez añadido los recursos deseados y rellenado el valor de cada uno de ellos, pulsamos sobre el botón “Guardar”. Los recursos de las plantillas podemos además marcarlos como pagados y eliminarlos de la propia plantilla pulsando sobre el botón con icono de papelera.

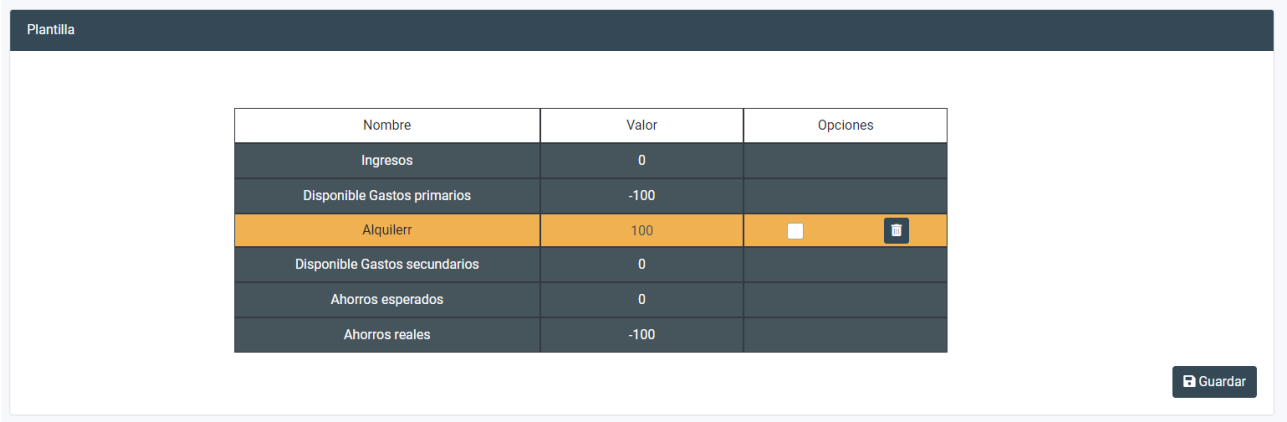


Ilustración 66: Plantilla

- Recibir consejos y buenas prácticas: puede observar que a medida que edite la plantilla recibirá consejos y buenas prácticas de cómo está gestionando la economía de ese mes.

- Estadísticas: en esta pantalla recibe estadísticas sobre:
    1. Ingresos y pagos por mes del año actual
    2. Total de ganancias y gastos
    3. Gastos donde invierte mayor cantidad de dinero.

Anexo 5. Libro de estilo

Libro de estilo que define la línea gráfica del trabajo:

- Logotipos y anagramas.

- Paleta de colores.

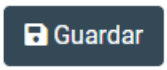
  Color principal #344955

  Color secundario #F9AA33

- Paleta tipográfica y tamaño de fuentes.

  Roboto de Google Fonts

- Botones.



Anexo 6. Bibliografía

Documentación de Angular. (s. f.). Recuperado 1 de abril de 2020, de https://angular.io/

Documentación de Laravel. (s. f.). Recuperado 1 de abril de 2020, de https://laravel.com/

Build fast, responsive sites with Bootstrap. (s. f.). Recuperado 15 de marzo de 2020, de https://getbootstrap.com/

PrimeNG. (s. f.). Recuperado 1 de abril de 2020, de https://primefaces.org/primeng/showcase/#/

Afolayan, F. (2018, 25 junio). Laravel and JWT. Recuperado 5 de abril de 2020, de https://blog.pusher.com/laravel-jwt/

Tymon, S. (2015, 22 abril). jwt-auth. Recuperado 1 de abril de 2020, de https://github.com/tymondesigns/jwt-auth/wiki/Creating-Tokens

Laracasts. (s. f.). Recuperado 1 de abril de 2020, de https://laracasts.com/

En Laravel: ¿Cómo enviar correos? (Configura el envío en 3 simples pasos). (s. f.). Recuperado 15 de abril de 2020, de https://programacionymas.com/blog/como-enviar-mails-correos-desde-laravel

Afolayan, F. (2018, 25 junio). Laravel and JWT. Recuperado 5 de abril de 2020, de https://blog.pusher.com/laravel-jwt/

Pattern library. (s. f.). Recuperado 15 de marzo de 2020, de http://www.welie.com/patterns/index.php

Design patterns. (s. f.). Recuperado 15 de marzo de 2020, de https://ui-patterns.com/patterns

Ferraris, J. C. (2017, 16 febrero). Patrones básicos de Navegación en Apps Móviles. Recuperado 15 de marzo de 2020, de https://medium.com/@juancaferraris/patrones-b%C3%A1sicos-de-navegaci%C3%B3n-en-apps-m%C3%B3viles-5b0b160ed1bb

Radic, Z. (2018, 10 junio). Loader Bar on Every HTTP Request in Angular 6. Recuperado 1 de mayo de 2020, de https://medium.com/@zeljkoradic/loader-bar-on-every-http-request-in-angular-6-60d8572a21a9

Cobos, J. (2017, 17 julio). Laravel: error stream_socket_enable_crypto(): al tratar de usar certificados autofirmados. Recuperado 20 de mayo de 2020, de https://javiercobossanz.com/2017/07/17/laravel-error-stream_socket_enable_crypto-al-tratar-de-usar-certificados-autofirmados/

Juan Carlos. (2018, 15 marzo). Angular inside Laravel? Recuperado 20 de mayo de 2020, de https://medium.com/@juancarlosjc/angular-inside-laravel-b155736ea84b

Anexo 7. Vita

Omar Adolfo Álvarez Hernández es natural de Santa Cruz de Tenerife. Es graduado en Ingeniería Informática por la Universidad de La Laguna en el año 2017. Comienza a trabajar en la misma empresa donde desarrolla las Prácticas Externas de la carrera, llamada Arte Consultores.

Después de un año de trabajo, decide comenzar el Máster en Desarrollo de Sitios y Aplicaciones Web al que pertenece esta memoria.

Durante el desarrollo de su trabajo, se siente apasionado y decide seguir formándose en el desarrollo web. Su idea es una vez finalizado este máster, seguir aprendiendo y estudiando nuevas tecnologías para poder seguir avanzando en su carrera y cada día convertirse tanto en mejor programador como persona.