DESARROLLO DE UNA HERRAMIENTA DE GESTIÓN DE CARTERAS DE INVERSIÓN

DEVELOPMENT OF AN INVESTMENT PORTFOLIO MANAGEMENT TOOL

TRABAJO FIN DE GRADO
CURSO 2020–2021

AUTOR
EDUARDO MARTÍNEZ MARTÍN

DIRECTOR
ANTONIO SARASA CABEZUELO

TRABAJO DE FIN DE GRADO EN INGENIERÍA DEL SOFTWARE

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

DESARROLLO DE UNA HERRAMIENTA DE GESTIÓN DE CARTERAS DE INVERSIÓN

DEVELOPMENT OF AN INVESTMENT PORTFOLIO MANAGEMENT TOOL

TRABAJO DE FIN DE GRADO EN INGENIERÍA DEL SOFTWARE

AUTOR
EDUARDO MARTÍNEZ MARTÍN

DIRECTOR
ANTONIO SARASA CABEZUELO

CONVOCATORIA: SEPTIEMBRE 2021

TRABAJO DE FIN DE GRADO EN INGENIERÍA DEL SOFTWARE

FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

6 DE SEPTIEMBRE DE 2021

Autorización de difusión

Los abajo firmantes, alumno y tutor del Trabajo Fin de Grado (TFG) en Ingeniería del Software de la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor, el Trabajo Fin de Grado (TFG) cuyos datos se detallan a continuación. Así mismo autorizan a la Universidad Complutense de Madrid a que sea depositado en acceso abierto en el repositorio institucional con el objeto de incrementar la difusión, uso e impacto del TFG en Internet y garantizar su preservación y acceso a largo plazo.

Título del TFG: **Desarrollo de una herramienta de gestión de carteras de inversión**

Curso académico: 2020–2021

Nombre del Alumno: **Eduardo Martínez Martín**

Director del TFG: **Antonio Sarasa Cabezuelo**, (Sistemas Informáticos y Computación).

Tutor del TFG: **Antonio Sarasa Cabezuelo**, (Sistemas Informáticos y Computación).

Firma de los alumnos Firma del director

La alegría está en la lucha, en el esfuerzo, en el sufrimiento que supone la lucha, y no en la victoria misma. Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa

Mahatma Gandhi

Agradecimientos

A mi padre, por apoyarme a lo largo de toda la carrera y recordarme que más vale tarde que nunca cuando quería tirar la toalla. Sin su apoyo e insistencia no hubiera llegado a la meta. Te quiero mucho.

Resumen

El presente trabajo consiste en una aplicación web que permite al usuario gestionar y analizar conjuntos de fondos de inversión(carteras de inversión).

Para facilitar la creación de carteras de inversión la aplicación dispone de un buscador de fondos y una lista de deseos que ayudan al usuario a buscar y guardar los fondos que le interesan.

Las carteras se almacenan en una lista y se pueden ordenar por parámetros tales como la rentabilidad, la volatilidad y los gastos, lo que ayuda a realizar su seguimiento y gestión.

De cada cartera podemos ver en detalle su distribución geográfica, sectorial y diez mayores posiciones o acciones, para así poder crear carteras en base a diferentes criterios.

La aplicación también funciona como una red social de inversores en la que los usuarios pueden ver y opinar acerca de las carteras de otros usuarios.

Palabras clave

Fondos, Carteras, Gestión, Análisis, Seguimiento, Inversores, Red social, Flask

Abstract

The present work consists in an app that allows a user to manage and analyze a set of funds(investment portfolio).

To simplify portfolio's creation the app has a fund's search engine and a wish list that helps the user to search and save funds.

Portfolios are stored in a portfolios list where you can order them by performance, volatility and expenses which helps to track and manage them.

For each portfolio we can see in detail its geographical distribution, sectorial distribution and top ten holding stocks to create a portfolio within a certain criteria.

The application also works as an investor social network where users can see and comment on other user's portfolio.

Palabras clave

Funds, Portfolios, Manage, Analysis, Tracking, Investors, Social media, Flask

Índice

Índice de Figuras

Índice de Tablas

1 Introducción

En este primer capítulo se explicará cual es la motivación que ha llevado a realizar el trabajo, los objetivos a alcanzar y el plan de trabajo.

1.1 Motivación

Desde pequeño veía a mi padre invertir en el mercado de valores, por lo que pronto me entro la curiosidad de saber en qué empresas estaría invirtiendo y cuáles eran las razones por las que invertía en ellas. A raíz de ello empecé a investigar sobre diferentes productos de inversión y acabé interesándome por los fondos.

Un fondo de inversión acumula el dinero de varios inversores para invertirlo de forma agregada en acciones y bonos, entre otros activos financieros. En este trabajo nos centraremos exclusivamente en los fondos de renta variable, que se componen de un conjunto de acciones que actúan como vehículo de inversión.

Un fondo puede ser de gestión pasiva porque replica a un índice de valores o de gestión activa donde un equipo gestor selecciona valores para intentar superarlo.

Una de las ventajas de los fondos de inversión es su exposición a diferentes países, sectores y empresas, lo que permite diversificar una inversión de forma sencilla. Diversificar consiste en repartir el ahorro entre diferentes activos para reducir el riesgo de perder nuestro dinero.

De cualquier fondo de inversión se puede consultar en páginas web como morningstar [1] su composición al detalle, pero ¿cómo podemos obtener dicha información si repartimos nuestras inversiones entre varios fondos?

Para resolver este problema se ha planteado en este trabajo el desarrollo de una aplicación web que permitirá conocer al detalle la información de un conjuntos de fondos(cartera) de manera agregada.

1.2 Objetivos

El objetivo principal del trabajo es el desarrollo de una aplicación web que permita a los usuarios analizar y gestionar sus carteras.

Para facilitar la creación de las carteras la aplicación dispondrá de un buscador de fondos y una lista de deseos donde se guardarán los fondos que interesen a los usuarios.

La aplicación también funcionará como una red social de inversores que permitirá a los usuarios opinar y compartir carteras.

Del objetivo principal podemos extraer varios objetivos específicos:

- La aplicación debe permitir buscar y guardar en una lista de deseos los fondos que interesen a los usuarios para posteriormente poder crear carteras de manera intuitiva y sencilla.

- La aplicación debe almacenar en una lista las carteras que cada usuario haya creado, permitiendo editar y eliminar cada una de ellas.

- La aplicación debe permitir ver en detalle la distribución geográfica, sectorial y diez mayores posiciones(acciones) de las carteras, entre otros parámetros.

- La aplicación debe permitir a los usuarios ordenar las carteras por parámetros tales como la rentabilidad, la volatilidad y los gastos.

- La aplicación debe permitir hacer públicas las carteras para que el resto de usuarios puedan opinar sobre ellas.

- La aplicación debe tener un buscador de carteras públicas.

1.3 Plan de trabajo

- Aprender las tecnologías en las que se va a desarrollar la aplicación

- Desarrollar funciones de extracción de datos mediante técnicas de web scrapping [2].

- Implementar las vistas de la aplicación de forma que la web sea responsive.

- Diseñar la base de datos con sus respectivas claves primarias y relaciones.

- Implementar la lógica de negocio de la aplicación

- Implementar funciones en JavaScript [3] para mejorar la experiencia de usuario.

1 Introduction

This chapter explains what the motivation is to make this work, the objectives to achieve and the work plan.

1.1 Motivation

Since I was little, I saw my father investing in the stock market, so I soon became curious to know in which companies he would be investing and what were the reasons why he was investing in them. As a result, I started researching different investment products and ended up being interested in funds.

An investment fund accumulates the money of several investors to invest it in an aggregate way in stocks and bonds, among other financial assets. In this work we will just focus exclusively on equity funds. An equity investment fund is an investment vehicle that owns a set of stocks.

A fund can be passively managed because it replicates an index of stocks or actively managed where a management team selects stocks to try to outperform it.

One of the advantages of investment funds is their exposure to different countries, sectors, and companies, which allows diversifying an investment in a simple way. Diversifying consists of dividing savings among different assets to reduce the risk of losing our money.

The composition of any investment fund can be consulted on web pages such as morningstar [1] in detail, but how can we obtain this information if we distribute our investments among several funds?

To solve this problem, the development of a web application has been proposed in this work that will allow to know in detail the information of a group of funds (portfolio) in an aggregated way.

1.2 Objectives

The main objective of this work is the development of a web application that allows users to analyze and manage portfolios.

To facilitate portfolios creation the app will have a funds search engine and a wish list that will save funds that interest users

The app also will work like an investors social network that will allow users to share and comment portfolios.

From the main objective we can extract several specific objectives:

- The application must allow users to search and save the funds that interest them in a wish list to be able to create portfolios later in an intuitive and simple way.

- The application must store in a list the portfolios that each user has created, allowing each one of them to be edited and deleted.

- The application must show a detailed view of portfolios geographical and sector distribution along with the ten largest positions(stocks), among other parameters.

- The application should allow users to sort portfolios by parameters such as performance, volatility, and expenses.

- The application must allow portfolios to be made public so that other users can comment them.

- The application must have a public portfolio search engine

1.3 Work plan

- Learn about the technologies in which the application will be developed

- Develop data extraction functions with web scrapping techniques [2].

- Implement responsive views of the application

- Design the database with their respective primary keys and relationships.

- Implement the business logic of the application

- Implement functions with JavaScript [3] to improve user experience

2 Estado del arte

En este capítulo se van a describir aplicaciones con características similares al proyecto realizado.

Existen pocas aplicaciones que ofrezcan servicios de comparación y análisis de carteras, la mayoría de ellas se centran exclusivamente en los fondos de inversión y no en carteras de inversión(conjunto de fondos)

A continuación se describen las aplicaciones más populares sobre fondos de inversión:

- Morningstar [1]:
    - Morningstar es una herramienta para la búsqueda de fondos de inversión y dispone de un servicio que genera informes detallados de carteras. Es la web de referencia para todos los inversores y es donde se encuentra la base de datos más completa y actualizada.

- Portfolio Performance [4]:
    - En una herramienta de código abierto muy útil que te permite gestionar y analizar tu cartera. Es una aplicación de escritorio y se centra en una variedad de instrumentos financieros, por lo que aumenta la complejidad para el inversor que solo quiera invertir en fondos. La herramienta no es del todo intuitiva y hay que dedicarle mucho tiempo para entender su uso.

- Finect [5]:
    - Finect es una aplicación web para comparar y encontrar los mejores productos financieros. Permite buscar fondos por temáticas, muestra los más populares y los más rentables. Tiene una sección en la que analizan el contenido de tu cartera si conectas tu banco a su aplicación.

- Allfunds [6]:
    - Allfunds es una gestora de fondos. Permite acceder a más de 200 mil fondos de inversión a través de su buscador, comparar fondos y monitorizar carteras de fondos. Solo permite el acceso a su aplicación a profesionales que trabajen para compañías financieras.

3 Especificación de requisitos

En este capítulo se realizará la especificación y el análisis de requisitos del sistema.

3.1 Actores del sistema

A continuación se describen los actores que podrían utilizar la aplicación:

- **Usuario no registrado:** representa a los usuarios que no están registrados en la aplicación. Un usuario no registrado solo podrá acceder al formulario de registro y al de iniciar sesión, por lo tanto es necesario estar registrado para usar la aplicación.

- **Usuario registrado:** representa a los usuarios registrados y tendrá acceso a todas las funcionalidades que ofrece la aplicación, tales como crear carteras, buscar fondos y demás servicios.

- **Administrador:** representa a la persona encargada de monitorizar los datos de la aplicación, revisándolos y actualizándolos cuando sea necesario.

3.2 Módulos

Las funcionalidades de la aplicación se han agrupado en tres módulos funcionales:

- Modulo Usuario
- Modulo Carteras
- Modulo Fondos

El actor administrador tendrá acceso a todas las funciones del sistema, con lo cual, para añadir más claridad a los diagramas, no aparecerá en ellos.

3.2.1 Módulo Usuario

Un usuario registrado podrá iniciar sesión, cerrar sesión, ver su perfil y el de otros usuarios, modificar su perfil y borrar su cuenta. Un usuario no registrado solo podrá registrarse en la aplicación.

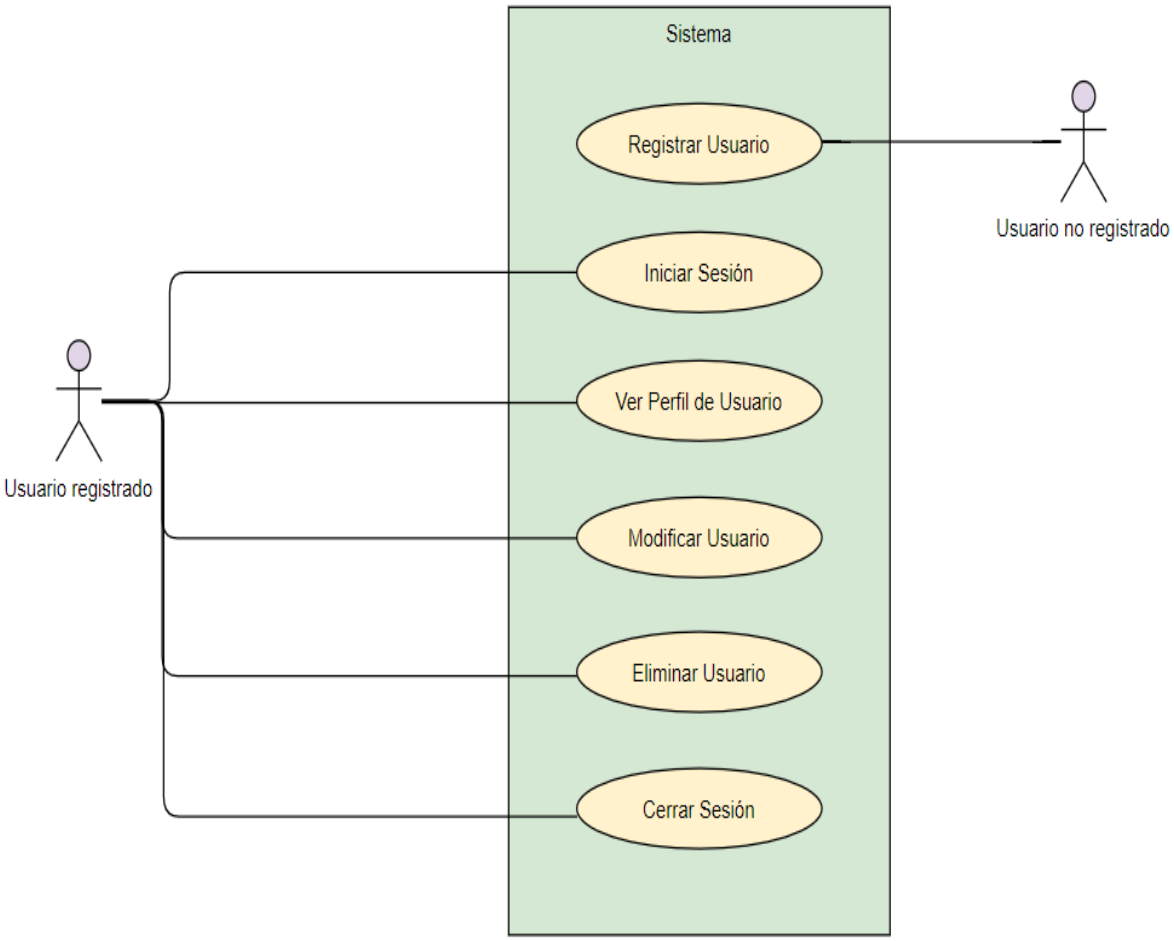

Figura 1 - Módulo usuario

3.2.2 Módulo Fondos

Un usuario registrado podrá buscar un fondo, añadirlo a su lista de deseos, eliminarlo de su lista de deseos y guardar su lista de deseos. Desde la lista de deseos se podrán añadir fondos directamente a las carteras. También podrá consultar los fondos más populares de la aplicación.

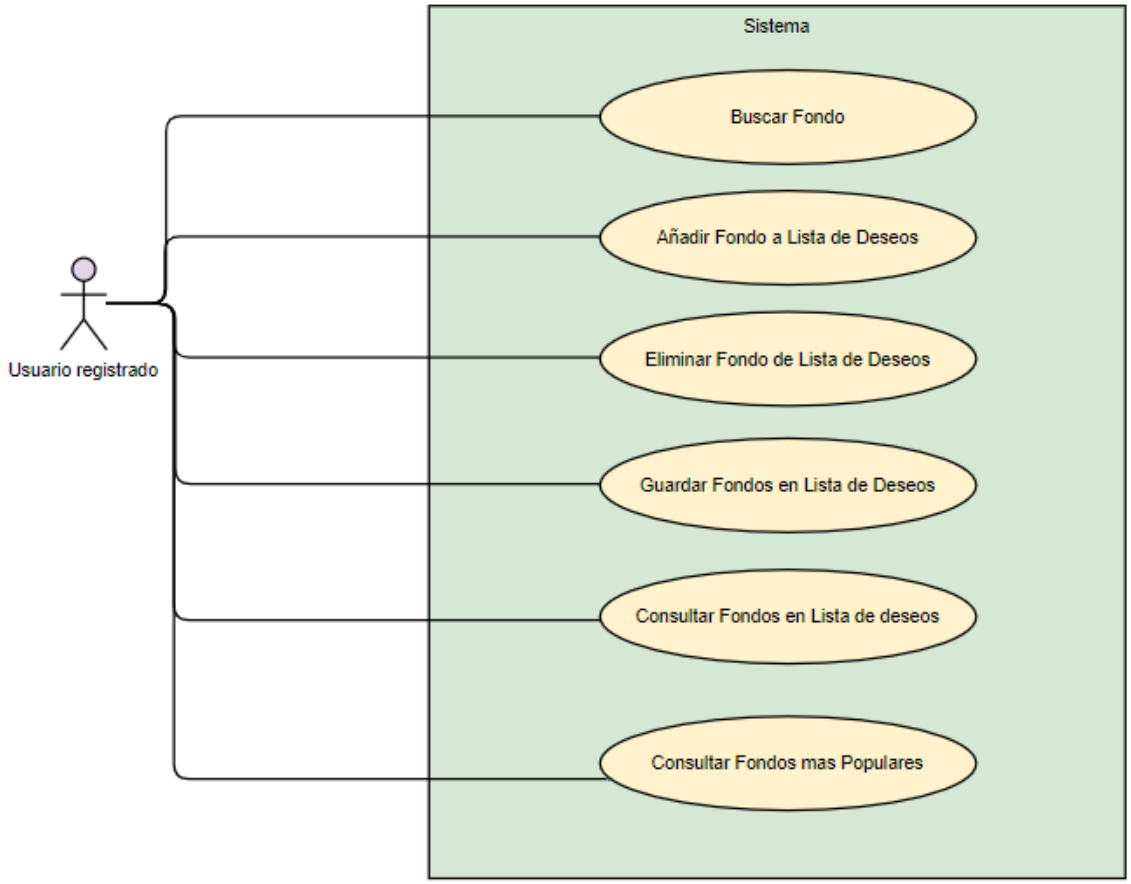

Figura 2 - Módulo Fondos

3.2.3 Módulo Carteras

Un usuario registrado podrá crear una cartera, editarla, borrarla, visualizarla en detalle y ver un listado donde se encuentran todas sus carteras privadas. Todas las carteras que cree serán privadas a no ser que se indique lo contrario en su creación o edición. Las carteras se pueden ordenar por parámetros como la rentabilidad o volatilidad, entre otros. Un usuario registrado también podrá ver un listado de carteras públicas, podrá comentar carteras públicas y podrá buscarlas por su nombre.

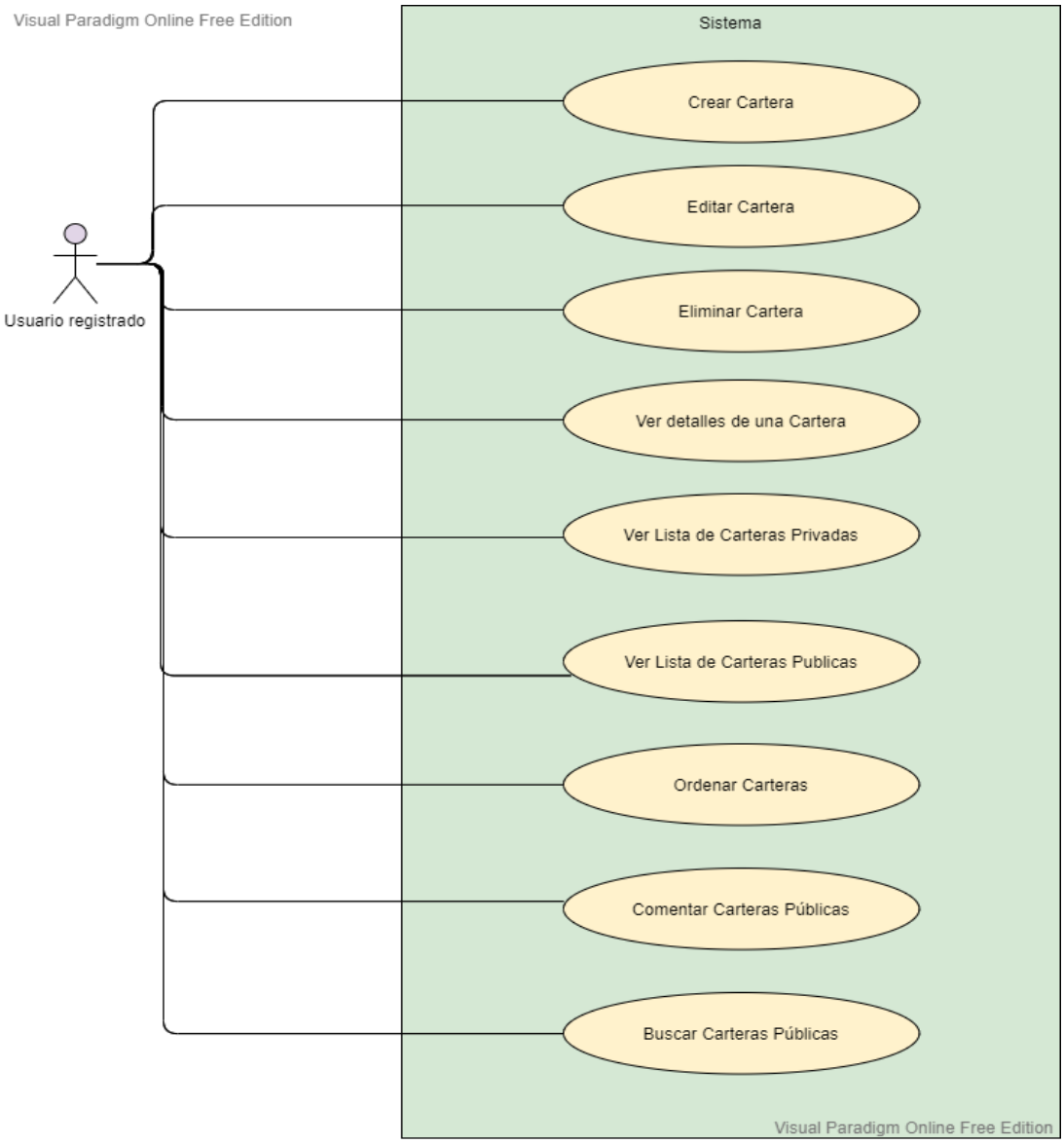

Figura 3 - Módulo Carteras

3.3 Análisis de requisitos

En este apartado se realizará un análisis de los requisitos del sistema que se ha desarrollado, diferenciando entre requisitos funcionales y no funcionales.

3.3.1 Requisitos funcionales

Un requisito funcional describe lo que debe hacer el sistema, a continuación se enumeran todos los requisitos funcionales de la aplicación:

- Un usuario no registrado podrá registrarse en la aplicación
- Un usuario podrá iniciar sesión
- Un usuario podrá cerrar sesión
- Un usuario podrá consultar su perfil
- Un usuario podrá consultar el perfil de otro usuario
- Un usuario podrá editar su perfil
- Un usuario podrá borrar su cuenta
- Un usuario podrá buscar un fondo de inversión
- Un usuario podrá añadir un fondo a su lista de deseos
- Un usuario podrá borrar un fondo de su lista de deseos
- Un usuario podrá guardar los fondos añadidos a su lista de deseos
- Un usuario podrá consultar su lista de deseos
- Un usuario podrá consultar los fondos más populares de la aplicación
- Un usuario podrá añadir un fondo desde su lista de deseos a una cartera
- Un usuario podrá crear una cartera
- Un usuario podrá editar una cartera
- Un usuario podrá eliminar una cartera
- Un usuario podrá ver en detalle una cartera
- Un usuario podrá consultar su lista de carteras privadas
- Un usuario podrá ordenar su carteras por rentabilidad, volatilidad, gastos y PER [7]
- Un usuario podrá ver las carteras públicas de otros usuarios
- Un usuario podrá comentar las carteras públicas de otros usuarios
- Un usuario podrá buscar por nombre las carteras públicas de otros usuarios

3.3.2 Requisitos no funcionales

Un requisito no funcional es aquel que impone algún tipo de restricción en el diseño o la implementación de una aplicación. A continuación se enumeran los requisitos no funcionales:

- La aplicación será una aplicación web
- La aplicación será responsive y las vistas serán maquetadas con Bootstrap [8]
- Se usará Python [9] como lenguaje en el servidor
- Se usará Flask [10] como framework para facilitar la creación de la aplicación web
- Se usará el motor de plantillas Jinja2 [11] para las vistas
- Se usará SQLAlchemy [12] como ORM para facilitar la comunicación con la Base de datos
- Se usará una base de datos relacional basada en SQL
- Se deberán encriptar por seguridad las contraseñas de los usuarios de la aplicación
- Se usará Selenium [13] para extraer los datos mediante técnicas de web scrapping
- Se utilizará el patrón de diseño MVC(Modelo-vista-controlador) [14] para que la aplicación sea más simple y mantenible.

4 Tecnología empleada

En este capítulo se van a comentar las tecnologías utilizadas para el desarrollo del proyecto.

4.1 Tecnologías para el desarrollo del Frontend

4.1.1 Bootstrap [8]

Es un framework CSS que sirve para desarrollar aplicaciones web responsive. Que sea responsive quiere decir que el frontend de la aplicación tiene la capacidad de adaptarse a la pantalla del dispositivo en el que se está ejecutando. Incluye una amplia gama de elementos ya prediseñados por lo que es muy útil para desarrollar páginas web bonitas sin necesidad de ser un experto en CSS.

4.1.2 jQuery [15]

Es una herramienta que permite agregar interactividad a una página web de forma más sencilla facilitando el uso de JavaScript en la parte del cliente. jQuery ayuda a reducir el número de líneas de código que se tendrían que escribir en JavaScript.

4.1.3 Chart.js [16]

Es una librería de JavaScript fácil de usar que sirve para crear diferentes tipos de gráficos para la visualización de datos. Entre ellos, gráficos de barras y circulares.

4.1.4 Jinja2 [11]

Es un motor de plantillas que ayuda a simplificar el desarrollo de las vistas de una aplicación. Se encuentra configurado en el Framework Flask de forma automática, permitiendo separar las vistas de la lógica de negocio del sistema.

4.2 Tecnologías para el desarrollo del Backend

4.2.1 Python [9]

Es un lenguaje de programación sencillo, versátil y fácil de leer. Tiene multitud de librerías y una comunidad muy extensa lo que facilita el desarrollo de cualquier aplicación.

4.2.2 Flask [10]

Flask es un framework escrito en Python que proporciona herramientas para construir de forma sencilla una aplicación web. Se define como un microframework ya que no tiene dependencias con librerías externas, por lo que es más flexible y tiene una curva de aprendizaje menor que otros frameworks desarrollados en Python como Django [17].

4.2.3 Sqlalchemy [12]

Es un conjunto de herramientas SQL y un ORM(Object Relational Mapper) [18] escrito en Python. Un ORM permite mapear clases con las tablas de una Base de datos y realizar todo tipo de operaciones sobre estas sin necesidad de conocer el lenguaje SQL.

4.3 Tecnologías para la extracción de los datos

4.3.1 Selenium [13]

Es una herramienta que sirve para automatizar una serie de acciones realizadas sobre un navegador web. Selenium permite identificar los elementos que nos interesan en un navegador y por lo tanto extraer información de estos.

4.4 Tecnologías para la gestión de los datos

4.4.1 XAMPP [19]

Es un paquete de software que incluye el servidor web Apache [20] y el sistema de gestión de bases de datos MySQL [21]. Permite desplegar una aplicación web en tu propio ordenador. También incluye el programa de administración de base de datos phpMyAdmin [22].

4.4.2 MySQL [21]

Es un sistema de base de datos relacional con modelo Cliente-servidor que sirve para crear y administrar bases de datos. Un modelo de base de datos relacional aporta sencillez e integridad a los datos.

4.5 Tecnologías de soporte al desarrollo

4.5.1 Visual Studio Code [23]

Es un editor de código compatible con varios lenguajes de programación. Permite agregar extensiones que facilitan enormemente el desarrollo.

4.5.2 Git [24]

Es un software de control de versiones y proporciona un listado de las diferentes versiones por las que ha pasado una aplicación a lo largo de todo su desarrollo. Permite deshacer cambios y volver a versiones anteriores con facilidad.

4.5.3 Venv [25]

Es un módulo que proporciona entornos con sus propios directorios aislados de los directorios del sistema. Es de gran utilidad para tener un conjunto independiente de paquetes de Python instalados, evitando de esta manera posibles conflictos con paquetes ya instalados en el sistema.

5 Modelo de datos

En este capítulo se va a describir el modelo de datos. Se hará distinción entre los que se almacenan en una base de datos relacional y los que se cargan dinámicamente.

5.1 Modelo entidad relación [26]

En la Figura 4 se muestra el modelo entidad relación, en el cual aparecen las entidades del sistema y sus relaciones. Posteriormente se implementa dicho modelo en la Figura 5, donde ya se pueden observar las tablas y sus columnas. El modelo ha sido implementado mediante una base de datos relacional basada en SQL.

Las columnas de los usuarios son: su nombre de usuario, su email, su contraseña y su imagen de perfil la cual es opcional.

Las columnas de las carteras son: el usuario al que pertenece dicha cartera, el nombre de la cartera, la descripción de la cartera y la fecha de creación de la cartera. Las carteras pueden ser públicas o privadas, por defecto son privadas.

Las columnas de los fondos son: la cartera a la que pertenecen, su nombre, dos códigos, uno le identifica unívocamente en cualquier buscador de fondos, y otro se corresponde al que le identifica en la página web de morningstar(necesario para extraer el resto de información mediante web scraping), el peso que supone el fondo en la cartera, y los gastos de gestión de dicho fondo.

Un usuario puede tener deseos(fondos que quiere añadir a sus carteras), por cada deseo se almacena el código del fondo y el nombre del fondo. Un cartera puede ser comentada por varios usuarios y varios usuarios pueden comentar una cartera, formando una tabla intermedia que contiene: el id de la cartera, el id del usuario que la comenta, el comentario y la fecha en la cual se escribió el comentario.

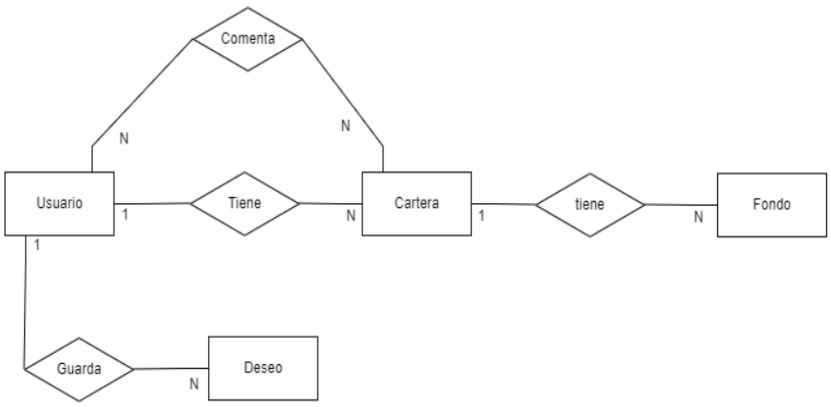

Figura 4 - Modelo entidad-relación

5.2 Implementación base de datos

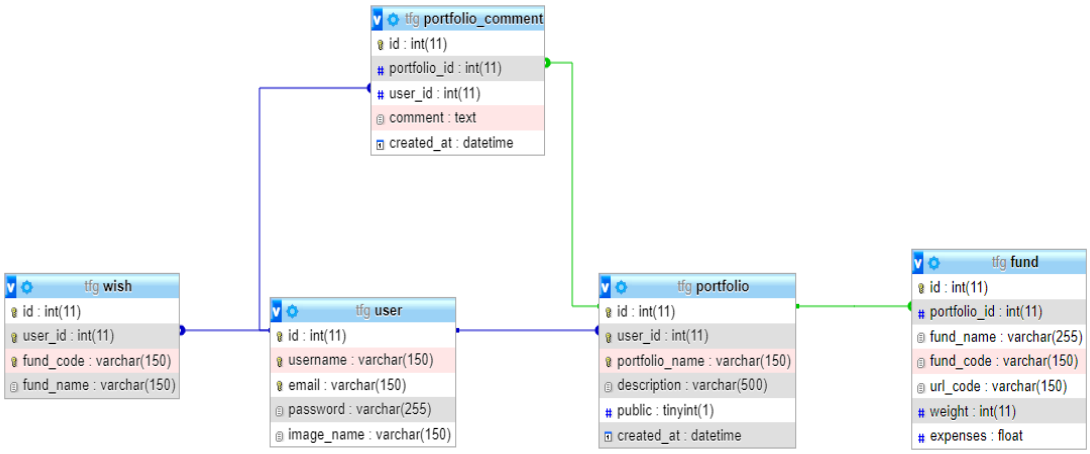

Figura 5 - Implementación de la base de datos

A continuación se van a describir cada una de las tablas de la base de datos:

USER

id	Número que identifica unívocamente a un usuario, se genera de forma automática y es autoincremental. Es la clave primaria.
username	Es el nombre de un usuario. Es único, por lo tanto no puede haber dos usuarios con distinto id y mismo nombre de usuario.
email	Es el email de un usuario.
password	Es la contraseña de un usuario.
image_name	Es el nombre de la imagen que usa como avatar el usuario.

Tabla 1 - Tabla de Usuarios

PORTFOLIO

id	Número que identifica unívocamente a una cartera, se genera de forma automática y es autoincremental. Es la clave primaria.
user_id	Es el id que identifica a que usuario pertenece la cartera. Es único.
portfolio_name	Es el nombre de la cartera. Es único.
description	Es la descripción de la cartera.
public	Representa si una cartera es pública o privada, es decir, si su valor es 1, será pública, en caso contrario será privada.
created_at	Es la fecha en la que se creó la cartera.

Tabla 2 - Tabla de Carteras de Inversión

FUND

id	Número que identifica unívocamente a un fondo, se genera de forma automática y es autoincremental. Es la clave primaria.
portfolio_id	Es el id que identifica a una cartera. Es único
fund_name	Es el nombre del fondo de inversión.
fund_code	Es el ISIN del fondo de inversión, código que identifica unívocamente a un fondo
url_code	Es el código que identifica a un fondo de inversión solo en la página web de morningstar
weight	Es el peso que representa el fondo en la cartera a la que pertenece.
expenses	Son los gastos de gestión del fondo.

Tabla 3 - Tabla de Fondos de Inversión

PORTFOLIO_COMMENT

id	Número que identifica unívocamente a un comentario, se genera de forma automática y es autoincremental. Es la clave primaria.
portfolio_id	Es el id que identifica a una cartera. Es único
user_id	Es el id que identifica a que usuario. Es único.
comment	Es el comentario que ha realizado el usuario(user_id) a la cartera(portfolio_id)
created_at	Es la fecha en la que se hizo el comentario.

Tabla 4 - Tabla de Comentarios

WISH

id	Número que identifica unívocamente a un fondo que se encuentra en una lista de deseos, se genera de forma automática y es autoincremental. Es la clave primaria.
user_id	Es el id que identifica de que usuario es la lista de deseos. La lista de deseos representa los fondos que un usuario tiene guardados porque le interesa añadir a su cartera. Es único.
fund_code	Es el ISIN del fondo de inversión, código que identifica unívocamente a un fondo.
fund_name	Es el nombre del fondo de inversión

Tabla 5 - Tabla de Deseos

5.3 Datos dinámicos

Los datos dinámicos han sido extraídos mediante técnicas de web scrapping [2]. Son datos, que de no obtenerlos dinámicamente habría que actualizarlos de manera continua, de otro modo, no se mantendría la coherencia de estos.

Como se observa en la Figura 6, por cada cartera de inversión o conjunto de fondos se pueden observar los siguientes datos:

- **Rentabilidades acumuladas** [27]
    - Rentabilidades a 3 meses, 6 meses y 1 año de una cartera.
    - Rentabilidad YTD(year to date) [28], que representa la rentabilidad desde el comienzo del año hasta la fecha actual.

- **Rentabilidades anualizadas**
    - Rentabilidades a 3 años y 5 años anualizadas

- **Volatilidad** [29]**:** Es lo que varía la rentabilidad de un activo respecto a su media en un periodo determinado. Nos ayuda a medir el riesgo.
    - Volatilidad a 3 años
    - Volatilidad a 5 años

- **Comisión:** Representa la comisión total del conjunto de fondos de la cartera, calculada teniendo en cuenta los pesos de los fondos en una cartera y la comisión propia de cada uno de ellos.

- **PER** [7]**:** Es el ratio que compara el precio de una acción por el beneficio por acción de una empresa. El PER de un fondo es la suma de los PER de todos sus valores, teniendo en cuenta el peso de cada uno de ellos. Por lo tanto, el PER de una carretera es la suma de los PER de todos sus fondos teniendo en cuanto el peso de cada uno de ellos sobre la cartera que hayamos creado.

Rentabilidades			*Datos*	
3 meses	**6 meses**	**1 año**	**Comisión**	**PER**
10,74 %	9,60 %	41,91 %	1.87 %	40,10
YTD	**3 años/an**	**5 años/an**	**Volatilidad 3a**	**Volatilidad 5a**
14,82 %	29,00 %	22,65 %	20,63 %	16,64 %

Figura 6 - Datos Dinámicos

- **Distribución geográfica**
    - Representa en que zonas geográficas tienen mayor peso las inversiones de una cartera.
- **Distribución sectorial**
    - Representa en que sectores tienen mayor peso las inversiones de una carretera
- **Diez primeras acciones**
    - Representan las acciones(empresas) de mayor peso en la cartera

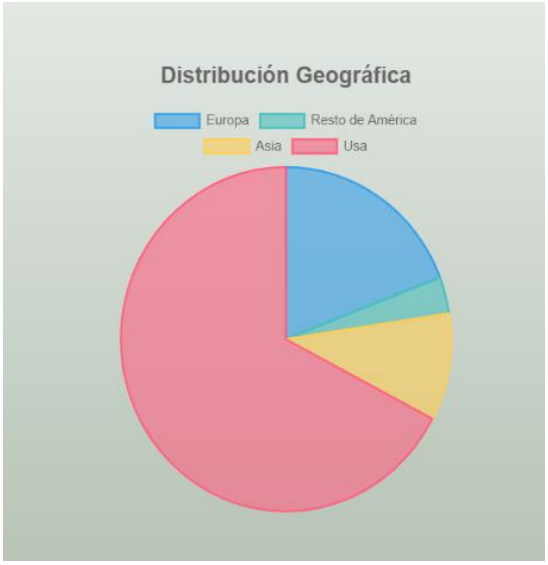

Figura 7 - Datos de Distribución geográfica

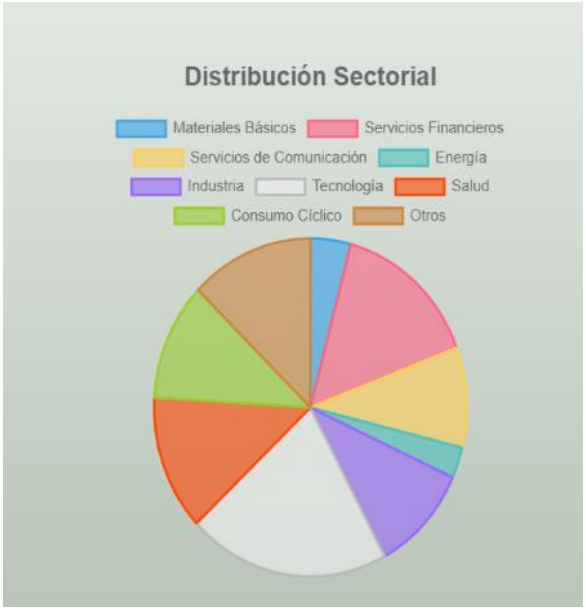

Figura 8 - Datos de Distribución sectorial

6 Arquitectura

En este capítulo se describe la arquitectura del sistema y los patrones arquitectónicos y de diseño aplicados en el desarrollo de la aplicación.

6.1 Arquitectura del sistema

Las aplicaciones web están basadas en una arquitectura cliente/servidor [30], por un lado está el navegador que representa al cliente, y por otro, el servidor web, que responde a las peticiones del cliente. Se usa la arquitectura típica cliente/servidor con el servidor web alojado en local.

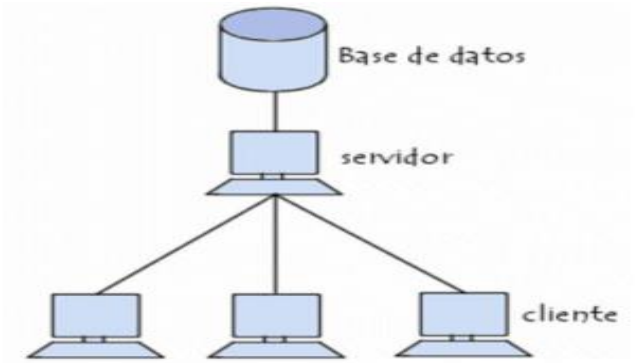

Figura 9 - Arquitectura cliente/servidor

6.2 Patrones arquitectónicos y de diseño

6.2.1 Modelo Vista Controlador [14]

El **patrón MVC** divide la aplicación en tres capas, cada una de ellas con una tarea bien definida:

- **Modelo:** contiene la representación de los datos del sistema y se encarga de su gestión.
- **Vista:** representa la interfaz de la aplicación, compuesta por la información que se le envía al usuario.
- **Controlador:** actúa como intermediario entre el Modelo y la Vista, comunicándose con ambos para dar la respuesta adecuada a la petición de un usuario.

En la aplicación desarrollada los paquetes **models, templates y routes** corresponden respectivamente a las capas del modelo, la vista y el controlador.

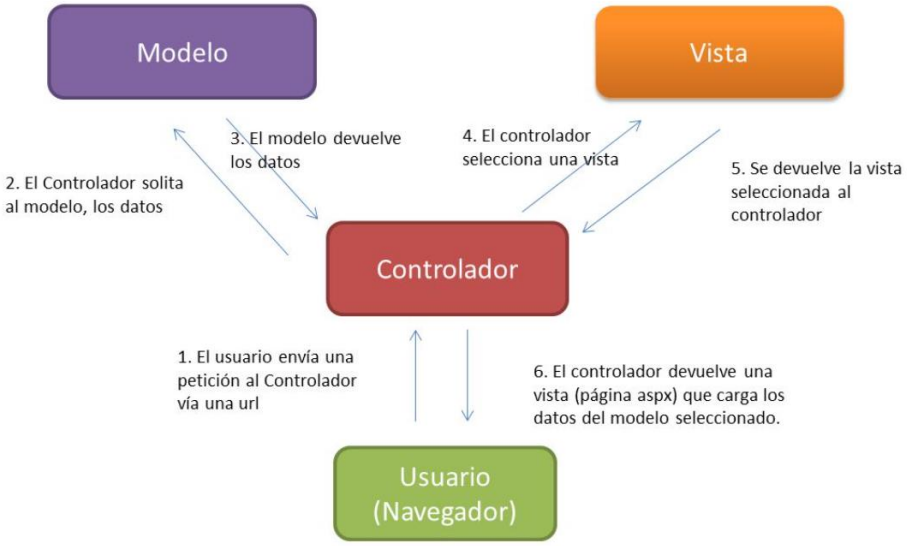

Figura 10 - Modelo Vista Controlador

6.2.2 Patrón Decorador [31]

Es un patrón de diseño estructural que permite añadir funcionalidades en tiempo de ejecución a objetos. Por ejemplo, si tienes una serie de funcionalidades en una aplicación a las que solo quieres que tengan acceso los usuarios registrados, se podría añadir un decorador que compruebe si un usuario ha iniciado sesión.

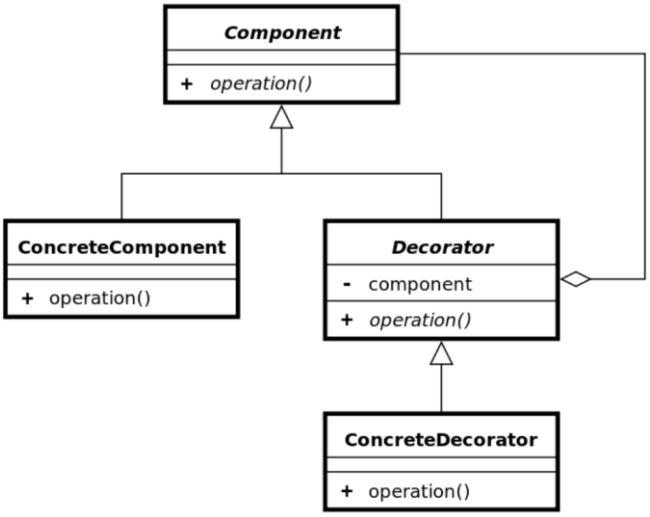

Figura 11 - Patrón Decorador

En la aplicación desarrollada se ha implementado un decorador que redirige al usuario a la página de inicio de sesión en el caso de que no la haya iniciado, por lo tanto, es necesario estar registrado e iniciar sesión en la aplicación para poder acceder al resto de funcionalidades.

7 Implementación

En este capítulo se va a describir el diseño y las principales funcionalidades de la aplicación.

7.1 Diseño

Partiendo de la base de que se quería desarrollar una aplicación web responsive que se adaptara a cualquier tipo de pantalla, el frontend se desarrolló con Bootstrap.

Respecto al **diseño** de los demás elementos visuales de la aplicación como la barra de navegación o los botones, bootstrap dispone de una colección de elementos prediseñados por lo que no es necesario apenas agregarles estilo. También se utilizaron iconos de Google fonts [32] en varias vistas de la aplicación. En las Figuras 12 y 13 se puede observar como la aplicación es responsive.

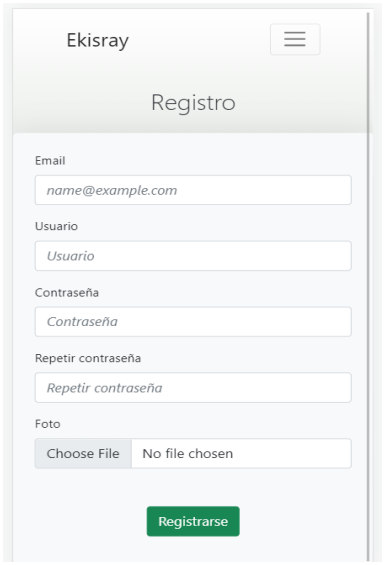

Figura 12 – Vista de registro en móvil

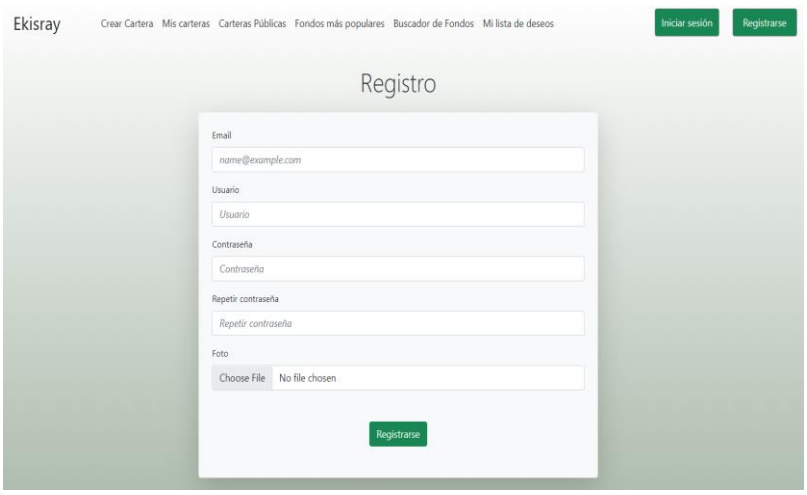

Figura 13 – Vista de registro en web

Para los **gráficos** de las distribuciones geográfica y sectorial de las carteras se optó por dos diagramas de sectores implementados con ChartJS.

Respecto a la **usabilidad** de la aplicación era necesario desarrollar varias funcionalidades en JavaScript. A continuación, se enumeran las más importantes:

- Un sumador automático de los pesos de los fondos(deben sumar 100).
- Un agregador de filas para añadir cuantas filas quieras al formulario de crear cartera.
- Un pop-up para la lista de deseos que permita consultarla en cualquier vista de la aplicación.
- Mensajes de información que ayuden al usuario a realizar diferentes acciones.
- Pulsar el nombre de un fondo en el buscador y que se añada a la lista de deseos.
- Un botón que permita agregar un fondo al formulario de crear cartera desde la lista de deseos

7.2 Funcionalidad de la aplicación

En este apartado se describen las funcionalidades implementadas, así como las vistas más importantes de la aplicación.

7.2.1 Registro e inicio de sesión

En primer lugar, para hacer uso de la aplicación es necesario que el usuario haya iniciado sesión, en caso contrario será redirigido a la página de "login". Un usuario que no haya iniciado sesión solo tendrá acceso a los formularios de registro y de inicio de sesión. Para ello, se ha implementado un decorador, en la Figura 14 se puede observar su implementación, y en la Figura 15 como se añade dicho decorador a una ruta en particular.

```python
# Decorator to use the app only if you are registered
def login_required(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if 'user' in session:
            return f(*args, **kwargs)
        else:
            flash("Necesitas iniciar sesión para usar la aplicación", "error")
            return redirect(url_for('login'))
    return wrap
```

Figura 14 - Implementación del Decorador

```
@app.route('/create_portfolio', methods=['POST', 'GET'])
@login_required
def create_portfolio():
```

Figura 15 - Decorador añadido a función

El manejo de las sesiones en Flask se lleva a cabo a través del objeto "sessión" [33], cuyo objetivo es mantener la información del usuario en las cookies del navegador. Cuando un usuario inicia sesión se guarda su "id" y su "username", como podemos observar al final de la implementacion del inicio de sesión en la Figura 16. El objeto "session" permite implementar el decorador que se ha mencionado anteriormente.

```
@app.route('/login', methods=['POST', 'GET'])
def login():
    if request.method == 'POST':
        user = User.query.filter_by(email=request.form['email']).first()
        if user is None:
            flash("El email introducido no es correcto!!", "error")
            return render_template('login.html')
        if not bcrypt.check_password_hash(user.password, request.form['password']):
            flash("La contraseña introducida es incorrecta!!", "error")
            return render_template('login.html')

        session['user'] = user.id
        session['username'] = user.username
```

Figura 16 - Implementación login

Para mantener la seguridad de las contraseñas, se guardan encriptadas(Figura 18), para ello se utiliza la librería Flask-Bcrypt [34], como podemos observar en la Figura 17.

```
pw_hash = bcrypt.generate_password_hash(
    request.form['password']).decode('utf-8')
```

Figura 17 - Uso de librería Flask-Bcrypt

id	username	email	password
4	edu	edu@gmail.com	$2b$12$8CY.yM59u7ux4bkh9SeMTeG1iqyVi22Ut4ShrPQ4RTO...

ara los elementos que están marcados: Editar Copiar Borrar Exportar

Figura 18 - Contraseña encriptada en base de datos

La clase “User” en el paquete “models” permite registrar usuarios con facilidad con la ayuda del ORM SQLAlchemy, lo que facilita enormemente al manejo de los datos de la aplicación. En la Figura 19 se muestra la implementación de la lógica de negocio del registro, y se resalta la función del ORM.

```python
@app.route('/register', methods=['POST', 'GET'])
def register():
    if request.method == 'POST':
        # checkings
        if User.query.filter_by(email=request.form['email']).scalar():
            flash('Email ya existente!!', "error")
            return render_template('register.html')
        if User.query.filter_by(username=request.form['user']).scalar():
            flash('Nombre de usuario ya existente!!', "error")
            return render_template('register.html')
        if request.form['password'] != request.form['repeat']:
            flash('Las contraseñas no coinciden!!', "error")
            return render_template('register.html')


        filename = request.files['photo'].filename if request.files['photo'] else 'no_avatar.png'
        pw_hash = bcrypt.generate_password_hash(
            request.form['password']).decode('utf-8')

        # create and insert user
        user = User(username=request.form['user'],
                    email=request.form['email'],
                    password=pw_hash,
                    image_name=filename)

        db.session.add(user)
        db.session.commit()

        # save photo
        if filename != 'no_avatar.png':
            path = os.path.join(
                app.config['UPLOAD_FOLDER'], request.files['photo'].filename)
            request.files['photo'].save(path)

        flash("Se ha registrado con éxito!!", "exito")
        return redirect(url_for('login'))
    return render_template('register.html')
```

Figura 19 - Implementación del registro

7.2.2 Buscador de fondos

La función principal del buscador de fondos es la de ayudar al usuario a encontrar fondos de diferentes temáticas o estilos de inversión. Esto supone una tarea sencilla si se introducen palabras clave como “tech” o “energy”, que mostraran resultados de fondos de temática tecnológica y energética respectivamente. También existe la posibilidad de buscar un fondo por su código identificativo o su nombre.

Por cada resultado de la búsqueda se muestra: el nombre del fondo, su código identificativo, y la rentabilidad acumulada desde el inicio del año, como se observa en la Figura 20.

Figura 20 - Buscador de fondos

El buscador de fondos se ha **implementado** con la ayuda de Selenium, usando técnicas de web scraping para la extracción de los datos, como se observa en la Figuras 21 y 22.

```
def extract_funds(driver):
    # fondos
    elements = WebDriverWait(driver, wait_time).until(EC.presence_of_all_elements_located(
        (By.CSS_SELECTOR, '#funds-browser-holder > div.broker-product-advanced-search__content \
                          > div > div > div > div > section > div > div')))

    funds = []
    for elem in elements:
        fund = {}
        elem = elem.text.splitlines()
        fund["name"] = elem[2]
        fund["isin"] = elem[3]
        fund["performance"] = elem[len(elem) - 2]
        funds.append(fund)
    return funds
```

Figura 21 - Implementación del web scraper del buscador 1

```
def start_driver_openbank():
    chrome_options = Options()
    chrome_options.add_argument("--headless")
    chrome_options.add_argument('--log-level=3')
    driver = webdriver.Chrome(options=chrome_options)
    driver.get(
        'https://www.openbank.es/en/broker-online/investment-funds-finder')

    # close the pop up
    try:
        close_pop_up(driver)
    except:
        print("No pop-up")

    # Choose equity option
    options = WebDriverWait(driver, wait_time).until(EC.element_to_be_clickable(
        (By.XPATH, '//*[@id="react-select-2--value"]/div[1]')))

    options.click()
    x = int(options.location['x'])
    y = int(options.location['y'])
    action = webdriver.common.action_chains.ActionChains(driver)
    action.move_by_offset(x + 0, y + 170)
    action.click()
    action.perform()

    return driver
```

Figura 22 - Implementación del web scraper del buscador 2

7.2.3 Lista de deseos

La lista de deseos(Figura 23) sirve para mantener en memoria los fondos que el usuario quiera añadir a futuras carteras. Los fondos se añaden a la lista de deseos desde el buscador de la aplicación, como se puede observar en la Figura 24.

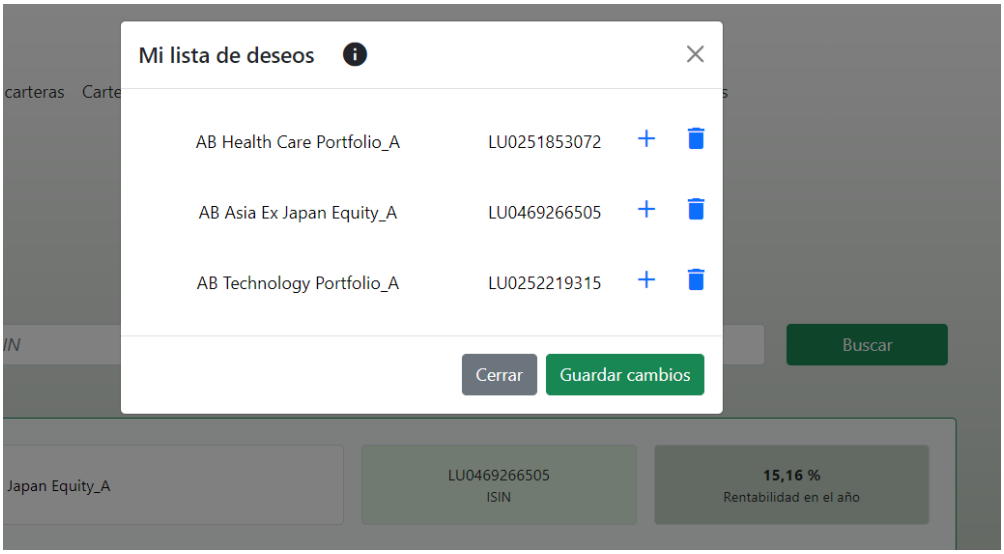

Figura 23 - Lista de deseos

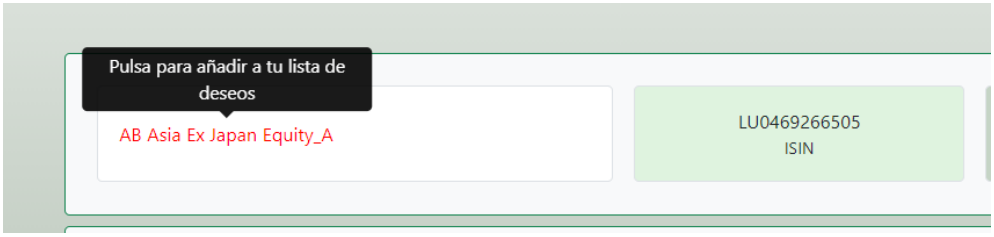

Figura 24 - Añadir a lista de deseos

La lista se guarda en la base de datos del sistema cuando el usuario pulsa el botón de “Guardar cambios”(Figura 23). El poder añadir y eliminar fondos de la lista se ha **implementado** en JavaScript con ayuda de jQuery. Un ejemplo de cómo se añade un fondo a la lista de deseos se muestra en la Figura 25.

```
// add fund row to wish list
//-------------------------------------------------------------------------------
$(".wish-list").click(function () {
  $(".modal-body").empty();
  for (var i = 0; i < localStorage.length; i++) {
    var key = localStorage.key(i);
    var value = localStorage.getItem(key);

    let fundEl = `<div class="fund-row-${key} row row-cols-4"></div>`;
    let name = `<input type="text" value="${value}" class="col-7 my-3 text-center" style="border: 0;" name="fund-name">`;
    let code = `<input type="text" value="${key}" class="col-3 my-3 text-center" style="border: 0;" name="fund-code">`;
    let icon_add = `<a class="col-1 my-3"><span class="material-icons add-icon" id=${key}>add</span></a>`;
    let icon_delete = `<a class="col-1 my-3"><span class="material-icons delete-icon" id=${key}>delete</span></a>`;

    $(".modal-body").append(fundEl);
    $(`.fund-row-${key}`).append(name, code, icon_add, icon_delete);
  }
});
```

Figura 25 - Implementación Añadir Fondo a Lista de Deseos

7.2.4 Creación de carteras

El formulario para la creación de las carteras se puede observar en la Figura 26.

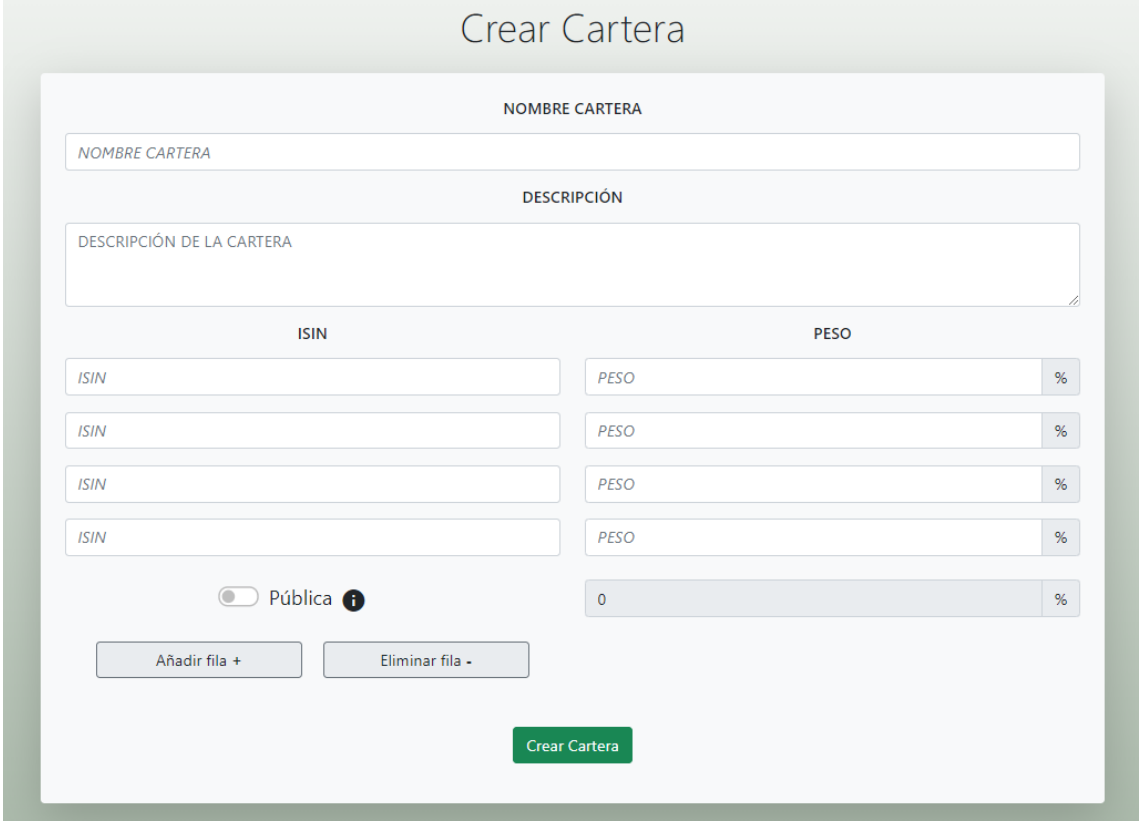

Figura 26 - Formulario Creación Cartera

Para poder crear una cartera, es necesario conocer los códigos ISIN [36] de los fondos que queremos incluir en ella, y además, los pesos que cada fondo va a representar en el conjunto de esta.

Introducir el código de un fondo se puede realizar de dos formas:

- Buscando el código ISIN que lo identifica en la web
- Añadiéndolo directamente desde la lista de deseos

Dependiendo del número de fondos del que se componga una cartera, existen dos botones en el formulario que nos permiten añadir y eliminar filas.

La suma de los pesos del conjunto de fondos de una cartera tiene que ser siempre 100, en caso contrario, el sistema no deja crearla. Se ha implementado un campo que hace la función de sumador automático según se vayan asignando los pesos.

Una cartera puede ser pública o privada. Una cartera por defecto es privada. Si se marca una cartera como publica todos los usuarios de la aplicación tendrán acceso a ella.

Para **implementar** la creación de carteras se necesita recopilar información de morningstar(web que contiene información detallada acerca de fondos de inversión). En concreto, de cada fondo se extrae: el nombre, los gastos de gestión, y el código que lo identifica en morningstar. Los códigos que identifican a los fondos en morningstar son necesarios para generar los informes de las carteras. En la Figura 27 se observa la implementación de la creación de una cartera, y se señala la parte del código en donde se scrapea la información. En la Figura 28 se muestra como la URL [37] correspondiente a un fondo contiene su código identificativo.

```python
@app.route('/create_portfolio', methods=['POST', 'GET'])
@login_required
def create_portfolio():
    if request.method == 'POST':
        # check portfolio name
        if Portfolio.query.filter(Portfolio.portfolio_name == request.form['portfolio_name']).\
                filter(Portfolio.user_id == session['user']).scalar():
            flash('Nombre de cartera ya existente!!', "error")
            return render_template('create_portfolio.html')

        # get fund ids and weights from user
        fund_ids = request.form.getlist('isin')
        weights = request.form.getlist('peso')
        fund_ids, weights = format_ids_weights(fund_ids, weights)

        # check repeated fund codes
        for elem in fund_ids:
            if fund_ids.count(elem) > 1:
                flash("Fondo repetido", "error")
                return render_template('create_portfolio.html')

        # check weight sum
        if sum(weights) != 100:
            flash("La suma de los pesos del conjunto de fondos no es del 100%", "error")
            return render_template('create_portfolio.html')

        # get funds info from morningstar (url_id, name, expenses)
        try:
            funds = web_scraper.get_funds_info_from_morningstar(fund_ids)
        except:
            flash("Error del servicio web, por favor inténtelo de nuevo","error")
            return redirect(url_for('create_portfolio'))
```

Figura 27 - Implementación Creación de Cartera

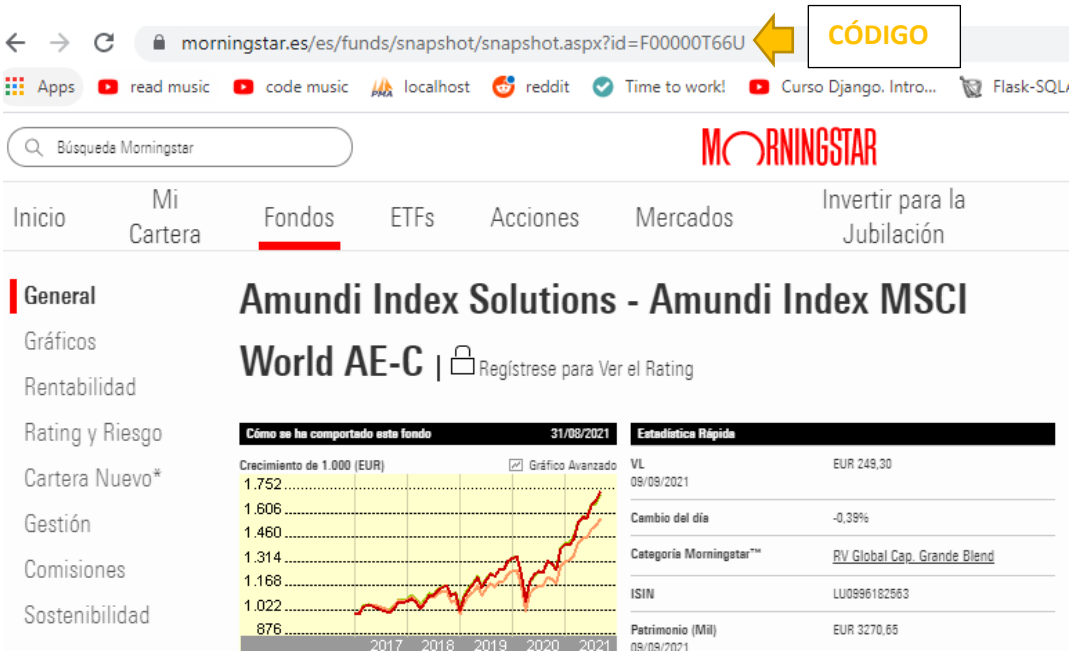

Figura 28 - Código de Fondo en Morningstar

7.2.5 Listar carteras

El usuario puede acceder a un listado de las carteras que ha creado en el apartado “Mis carteras”. En la figura 29 se puede observar el listado de carteras de un usuario

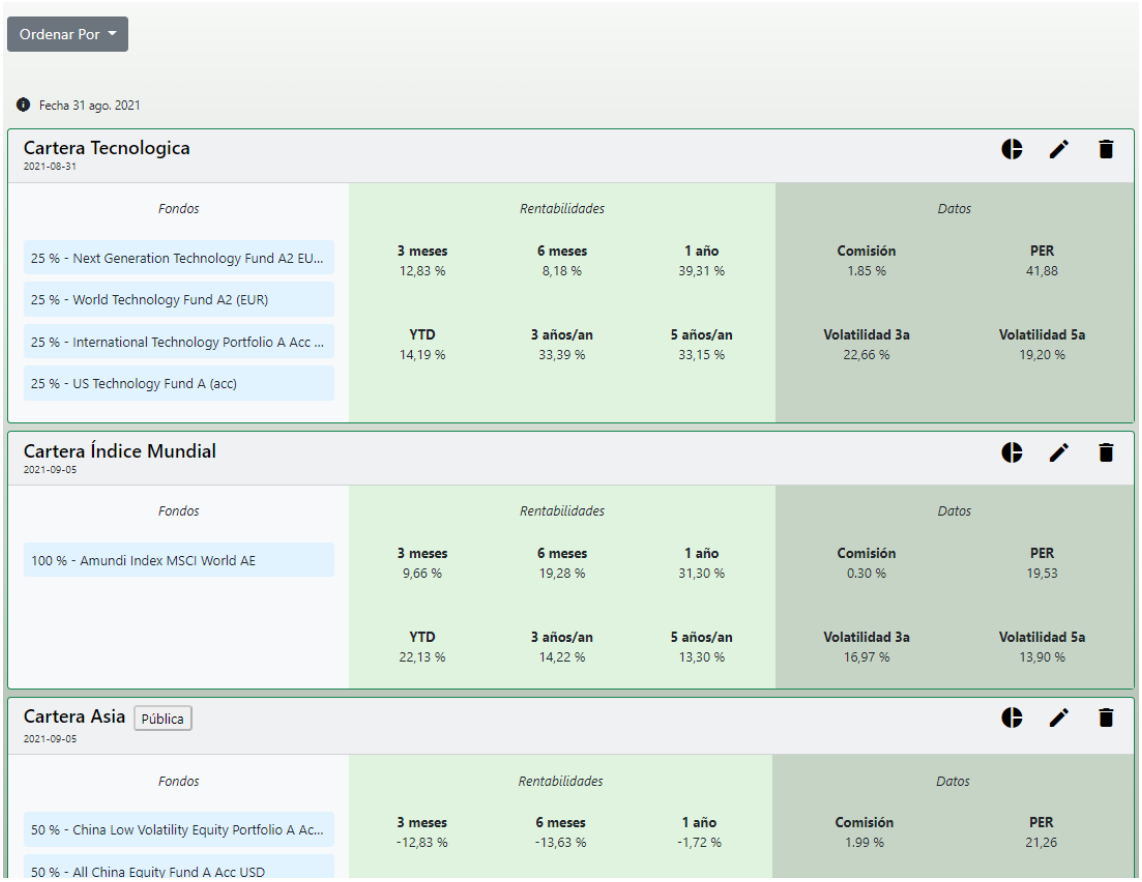

Figura 29 - Listado de Carteras de un Usuario

Como se puede observar en la Figura 30 cada cartera dispone de la siguiente información: nombre de la cartera, fecha en la que fue creada, etiqueta que indica si es publica, y 3 iconos. Pulsando el primer icono se accede a los detalles de una cartera, el segundo permite editarla, y el tercero borrarla. En el primer cuadrante se muestran los 4 fondos de mayor peso, en el segundo las rentabilidades, y en el tercero, la volatilidad, las comisiones y el PER [7].

Figura 30 - Información de una Cartera en el Listado de Carteras

Para implementar el listado de carteras de un usuario se extraen de la base de datos todas sus carteras, como podemos observar en la Figura 31. En la base de datos solo se encuentra cierta información relativa a las carteras que nos ayuda a extraer el resto de información mediante técnicas de web scraping. En la Figura 32 se puede observar cómo se integra la información obtenida de la base de datos con la extraída de la web, a partir del comentario "Get data from service".

```python
# get all private portfolios from user
@app.route('/portfolios')
@login_required
def portfolios():
    rows = Portfolio.query.filter_by(user_id=session['user']).join(Fund).all()
    portfolios = formatter.format_portfolios(rows)
    session['portfolios'] = portfolios
    return render_template('portfolios.html', portfolios=portfolios)
```

Figura 31 – Implementación del Listado de Carteras de un Usuario

```python
# Extract all the portfolios from the user
def format_portfolios(rows):
    portfolios = []

    for row in rows:
        total_expenses = 0
        portfolio = {}
        portfolio['id'] = int(row.id)
        portfolio['name'] = row.portfolio_name
        portfolio['public'] = row.public
        portfolio['created_at'] = str(row.created_at).split()[0]

        for f in row.funds:
            fund = {}
            fund['code'] = f.url_code
            fund['weight'] = f.weight
            fund['name'] = compress_fundname(f)
            fund['expenses'] = f.expenses
            total_expenses += f.expenses * f.weight

            if "funds" in portfolio:
                portfolio["funds"].append(fund)
            else:
                portfolio["funds"] = [fund]

        # sort funds
        portfolio["funds"].sort(key=lambda k: k['weight'], reverse=True)

        # calculate expenses
        expenses = total_expenses / 100
        expenses = round(expenses, 2)
        expenses = format(expenses, '.2f')
        portfolio['expenses'] = expenses

        # Get data from service
        codes = [fund['code'] for fund in portfolio["funds"]]
        weights = [fund['weight'] for fund in portfolio["funds"]]
        pdf_data = get_pdf_data(codes, weights)

        # add data
        performance = get_performance(pdf_data)
        portfolio['performance'] = performance
        pe = get_pe(pdf_data)
        portfolio['pe'] = pe
        vol_3, vol_5 = get_volatility(pdf_data)
        portfolio['volatility'] = (vol_3, vol_5)
        date = get_date(pdf_data)
        portfolio['date'] = date

        # add portfolio
        portfolios.append(portfolio)

    return portfolios
```

Figura 32 - Integración de información del Listado de Carteras

7.2.6 Extracción de datos

La rentabilidad, la volatilidad, las comisiones y el PER de una cartera, entre otros datos, se obtienen llamando a una URL que genera un informe con la información detallada de una cartera o conjunto de fondos. Para llamar a la URL se necesita el código que identifica a cada fondo en morningstar(extraído de su web en la creación de la cartera) y el peso que supone dicho fondo en una cartera. Una vez generado el informe se extrae la información necesaria y se elimina.

En las figuras 33, 34 y 35 se puede observar parte de la **implementación** de lo anteriormente descrito.

```
    # Get data from service
    codes = [fund['code'] for fund in portfolio["funds"]]
    weights = [fund['weight'] for fund in portfolio["funds"]]
    pdf_data = get_pdf_data(codes, weights)

    # add data
    performance = get_performance(pdf_data)
    portfolio['performance'] = performance
    pe = get_pe(pdf_data)
    portfolio['pe'] = pe
    vol_3, vol_5 = get_volatility(pdf_data)
    portfolio['volatility'] = (vol_3, vol_5)
    date = get_date(pdf_data)
    portfolio['date'] = date

    # add portfolio
    portfolios.append(portfolio)
```

Figura 33 - Llamada a la función generadora del informe

```
def get_pdf_data(codes, weights):
    report_url = web_scraper.call_pdf_url(codes, weights)
    pdf = save_pdf(report_url)
    data = pdf_to_text(pdf)
    delete_pdf_file()
    return data
```

Figura 34 - Función que extrae los datos del Informe y elimina el Informe

```
def call_pdf_url(funds_ids, percents):
    funds = combined(funds_ids)
    percents = combined(percents)
    strs = ["FO" for x in range(len(funds_ids))]
    strs = combined(strs)

    getVars = {'securityIds': funds,
               'marketValues': percents, 'typeids': strs}
    url = 'https://lt.morningstar.com/j2uwuwirpv/xraypdf/default.aspx?LanguageId=es-ES&CurrencyId=EUR&'

    url = url + urllib.parse.urlencode(getVars)
    return url
```

Figura 35 - Función generadora del Informe

7.2.7 Ordenación de carteras

Respecto a la lista de carteras, es posible ordenarla por diferentes parámetros, lo que es de gran utilidad para compararlas. El parámetro por el que se ordena la lista aparece resaltado en rojo como se puede observar en la Figura 36.

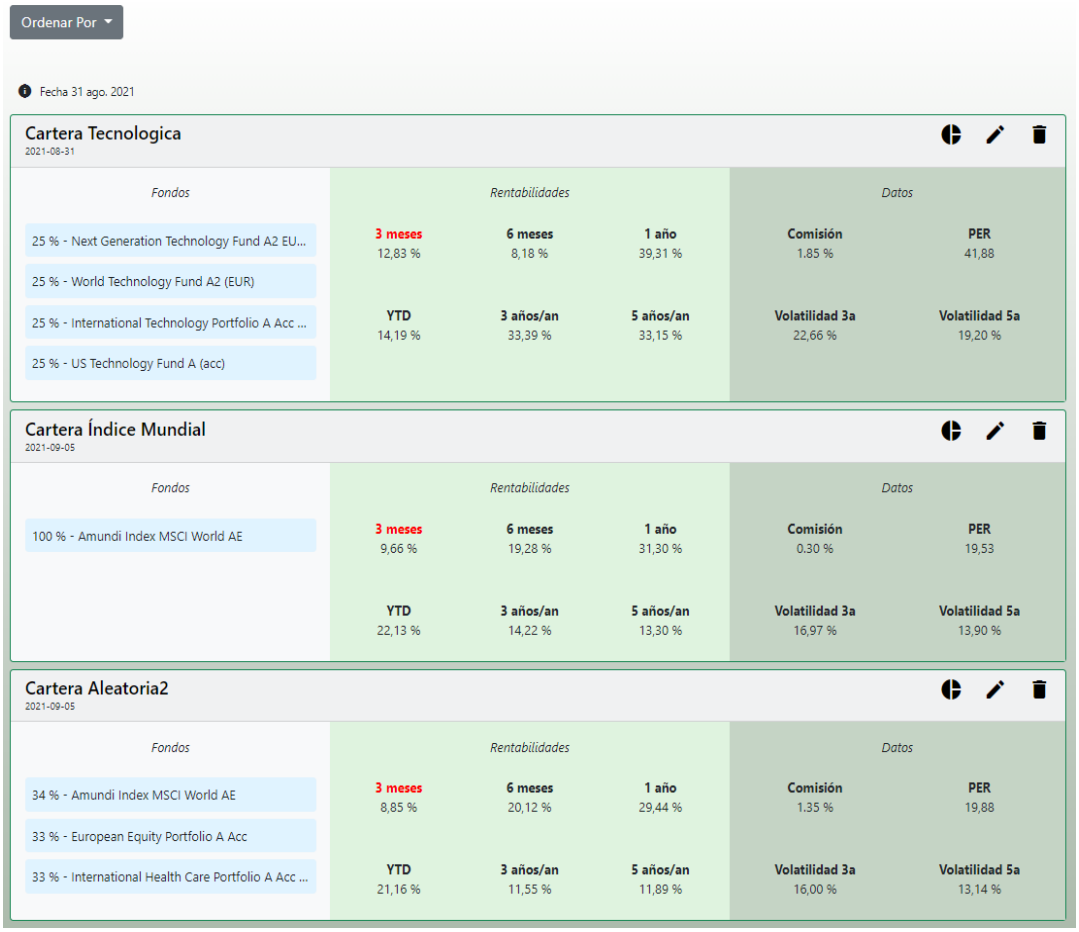

Figura 36 - Ordenación de Carteras

La implementación de la ordenación de la lista de carteras es sencilla, cada cartera es un diccionario dentro de una lista de diccionarios, con lo cual solo hay que indicarle a la función "sorted" por qué clave se quiere ordenar la lista, como se puede observar en la Figura 37.

```
# Order private portfolios
# ------------------------------------------------------------------------------
@app.route('/portfolios/orderby/expenses')
@login_required
def porfolios_by_expenses():
    portfolios = sorted(session['portfolios'], key=itemgetter('expenses'))
    return render_template('portfolios.html', portfolios=portfolios)
```

Figura 37 - Implementación de Ordenar Carteras por comisiones

7.2.8 Ver detalles de una cartera

El usuario puede acceder a los detalles de cada cartera de su lista. De los detalles de una cartera se puede visualizar la distribución geográfica y sectorial en dos diagramas de sectores, las 10 primeras acciones(empresas) de mayor peso en la cartera y la lista de fondos que componen la cartera. Los gráficos están implementados con ChartJS. En la Figura 38 se puede observar la vista de los detalles de una cartera.

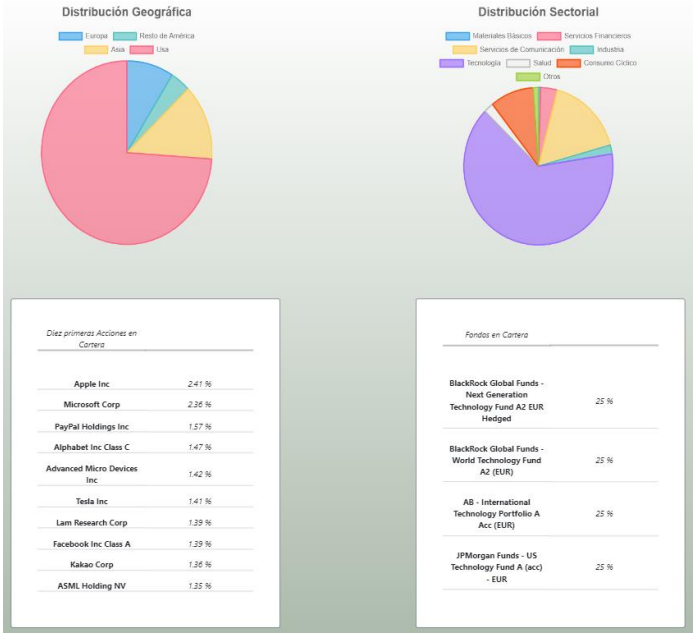

Figura 38 - Detalles de composición de una cartera

Los detalles de una cartera se obtienen del informe anteriormente mencionado en el apartado extraccion de datos. En la Figura 39 se muestra la implementacion de los detalles de una cartera.

```
@app.route('/portfolio/<id>')
@login_required
def get_portfolio(id):
    # get portfolio info from db
    portfolio = Portfolio.query.get(id)

    # need codes and weights of funds to call the web service and build the pdf
    codes = []
    weights = []
    funds = {}
    for fund in portfolio.funds:
        codes.append(fund.url_code)
        weights.append(fund.weight)
        funds[fund.fund_name] = fund.weight

    # get pdf data from web service
    pdf_data = pdf_scraper.get_pdf_data(codes, weights)

    # get portfolio info from pdf
    continents = pdf_scraper.get_continent_distribution(pdf_data)
    sectors = pdf_scraper.get_sectors(pdf_data)
    sectors_name = list(sectors.keys())
    sectors_weight = list(sectors.values())
    stocks = pdf_scraper.get_stocks(pdf_data)

    return render_template('portfolio.html', continents=continents, sectors_name=sectors_name,
                           sectors_weight=sectors_weight, stocks=stocks, funds=funds, portfolio=portfolio)
```

Figura 39 - Implementación detalles de una cartera

7.2.9 Carteras públicas y red social de Inversores

Una de las funcionalidades más útiles de la aplicación es la posibilidad de hacer públicas las carteras de los usuarios. Si un usuario publica su cartera, el resto de usuarios tendrán acceso a ella en la lista de carteras públicas.

La lista de carteras públicas(Figura 40) se puede ordenar por diferentes parámetros, del mismo modo que la lista propia de un usuario. Además, el sistema permite buscar por nombre carteras públicas. Esta funcionalidad podría ayudar al usuario a filtrar carteras de una tematica en particular, ya que si el usuario publica una cartera que contiene fondos tecnológicos, podría ponerle un nombre como “Cartera Tech” o “Fondos Tech”.

De cada cartera publica, además de todos los detalles de su composición, se especifica el usuario que la creo(resaltado en Figura 40), pudiendo acceder al perfil de este. El perfil de un usuario contiene su correo, su nombre de usuario y su lista de carteras publicadas.

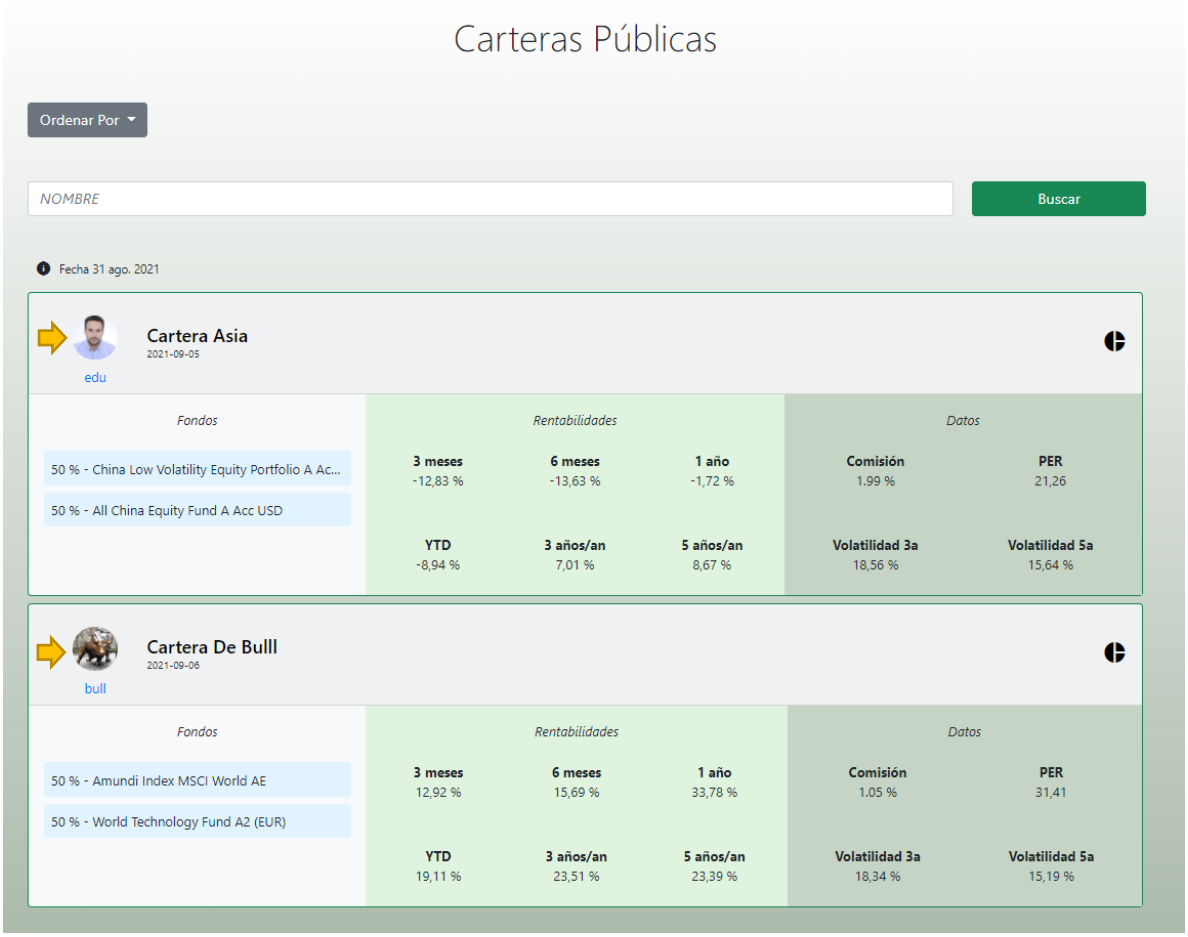

Figura 40 - Lista de carteras públicas

En la Figura 41 se observa cómo se extraen todas las carteras públicas de la base de datos filtrándolas por su atributo “public”.

```python
# get all public portfolios
@app.route('/public/portfolios')
@login_required
def public_portfolios():
    rows = Portfolio.query.filter_by(public=True).join(Fund).all()
    portfolios = formatter.format_public_portfolios(rows)
    session['public_portfolios'] = portfolios
    return render_template('public_portfolios.html', portfolios=portfolios)
```

Figura 41 - Implementación de Listar Carteras públicas

Todas las carteras publicas pueden recibir comentarios de los usuarios de la aplicación. Los comentarios pueden ayudar al creador de la cartera a replantearse la combinación de fondos de su cartera, por recomendación de uno o varios usuarios. El formulario para escribir un comentario y varios comentarios en una cartera publica se puede observar en la Figura 42.

Figura 42 - Formulario Comentarios y comentarios realizados

La implementación de comentarios en carteras públicas se puede observar en la Figura 43 y se realiza de la siguiente forma:

- Se recupera la cartera a la que accede el usuario de la Base de datos
- Se recupera el usuario que está usando la aplicación de la Base de datos
- Se crea el objeto “Portfolio_comment” que relaciona la cartera con el usuario.
- Se añade el comentario a la cartera
- Se añade el objeto “Portfolio_comment” a la base de datos

```python
@app.route('/comment/portfolio/<portfolio_id>', methods=['POST'])
@login_required
def comment(portfolio_id):
    if request.method == 'POST':
        # get comment and user_id
        comment = request.form.get('description')
        user_id = session['user']

        # link portfolio, user and comment
        portfolio = Portfolio.query.get(portfolio_id)
        user = User.query.get(user_id)
        comment = Portfolio_comment(
            comment=comment, user=user, portfolio=portfolio)
        portfolio.user_comments.append(comment)

        # add it to the database
        db.session.add(comment)
        db.session.commit()
        return redirect(url_for('get_portfolio', id=portfolio_id))
```

Figura 43 - Implementación de comentarios en Carteras Públicas

7.3 Acceso al repositorio de GitHub del proyecto

En este apartado se proporciona el enlace al repositorio de GitHub del proyecto:

https://github.com/edwardmartins/ekisray

8 Conclusiones y trabajo futuro

8.1 Conclusiones

En este proyecto se ha desarrollado una aplicación web que permite a los usuarios crear carteras de inversión(conjunto de fondos de renta variable) con el objetivo de conocer su exposición a diferentes países, sectores, y empresas, además de sus parámetros más significativos, como su rentabilidad, su volatilidad y sus comisiones. Es una herramienta que sirve para que los usuarios puedan realizar el seguimiento de sus ideas de inversión o de su propia cartera.

La aplicación ayuda a los usuarios en la creación de sus carteras, ya que dispone de un buscador de fondos y de una lista de deseos. La lista de deseos permite a los usuarios guardar fondos que les interesan para después incluirlos en sus carteras.

Además, la aplicación funciona como una red social de inversores y permite a los usuarios compartir y recibir opiniones de sus carteras.

Para finalizar se considera que el sistema desarrollado es de gran utilidad para gestionar y analizar carteras, aunque diversas acciones son lentas debido a la dificultad en la extracción de los datos, y la implementación de la aplicación como red social de inversores es una idea interesante pero desarrollada en local.

8.2 Trabajo futuro

A continuación se expondrán diferentes ideas para mejorar la aplicación:

- **Encontrar una API** [37] **para la extracción de los datos**
    - Los datos se extraen de varios portales web, por lo que la carga de datos es lenta y propensa a fallos. El objetivo sería encontrar una API que nos proporcionara la información de forma rápida y segura.
- **Publicar la aplicación en un servidor web publico**
    - La aplicación esta alojada en un servidor local, la idea sería alojarla en un servidor web público para conocer su popularidad.
- **Mejorar la aplicación como red social**
    - La aplicación solo permite compartir carteras y comentarlas, por lo que sería interesante poder enviar mensajes privados entre usuarios y dar “me gusta” a comentarios y carteras.
- **Mostrar estadísticas relevantes**
    - Se podrían obtener estadísticas relevantes, como la cartera más popular, la más diversificada o el usuario que ha creado las carteras más rentables, entre otras.

8 Conclusions and future work

8.1 Conclusions

A web application has been developed in this project that allows users to create portfolios(set of equity funds) with the objective to know their exposure to different countries, sectors and companies, in addition to their most important parameters like their performance, volatility and expenses. It is a tool that helps users to track their investments ideas or their own portfolio.

The app helps users to create their portfolios through a fund search engine and a wish list. The wish list allows users to save funds that interest them to include in their portfolios later.

Moreover, the app works like a investors social network that allows users to share and receive opinions about their portfolios.

Finally, it is considered that the developed system is a great tool to manage and analyze portfolios, although several actions are slow caused by data difficulty extraction, and the app implementation like a social network is a great idea but only developed locally.

8.2 Future work

The application can be improved in the following way:

- **Find a data extraction API**
    - Data is extracted from several websites causing slow and prone to fail data loading. The goal would be to find an API that provides us a faster and safer way to obtain data.
- **Publish the app in a public web server**
    - The app is hosted in a local server, the idea would be to publish it in a public web server to measure his success.
- **Improve the app as a social network**
    - The app only allows users to share and comment other user's portfolios, so it would be interesting to be able to send private messages between users and like user comments and portfolios.
- **Show relevant statistics**
    - The app could show users relevant statistics like the most popular portfolio, the most diversified portfolio or the user that has created the most profitable portfolios, among others.

Bibliografía

[1] Morningstar, [En línea]. Available: https://morningstar.es/es/. [Último acceso: 10 9 2021].

[2] Web scraping, [En línea]. Available: https://es.wikipedia.org/wiki/Web_scraping. [Último acceso: 10 9 2021].

[3] JavaScript, [En línea]. Available: https://developer.mozilla.org/es/docs/Web/JavaScript. [Último acceso: 10 9 2021].

[4] Portfolio performance, [En línea]. Available: https://www.portfolio-performance.info/. [Último acceso: 10 9 2021].

[5] Finect, [En línea]. Available: https://www.finect.com/. [Último acceso: 10 9 2021].

[6] Allfunds, [En línea]. Available: https://allfunds.com/es/. [Último acceso: 10 9 2021].

[7] PER ratio, [En línea]. Available: https://economipedia.com/definiciones/per-ratio.html. [Último acceso: 10 9 2021].

[8] Bootstrap, [En línea]. Available: https://getbootstrap.com/docs/5.1/getting-started/introduction/. [Último acceso: 10 9 2021].

[9] Python, [En línea]. Available: https://docs.python.org/3/. [Último acceso: 10 9 2021].

[10] Flask, [En línea]. Available: https://flask.palletsprojects.com/en/2.0.x/. [Último acceso: 10 9 2021].

[11] Jinja2. [En línea]. Available: https://jinja.palletsprojects.com/en/3.0.x/. [Último acceso: 10 9 2021].

[12] SQLAlchemy, [En línea]. Available: https://docs.sqlalchemy.org/en/14/. [Último acceso: 10 9 2021].

[13] Selenium with Python, [En línea]. Available: https://selenium-python.readthedocs.io/. [Último acceso: 10 9 2021].

[14] Patrón Modelo Vista Controlador, [En línea]. Available: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador. [Último acceso: 10 9 2021].

[15] JQuery, [En línea]. Available: https://api.jquery.com/. [Último acceso: 10 9 2021].

[16] Chart.js, [En línea]. Available: https://www.chartjs.org/docs/latest/. [Último acceso: 19 9 2021].

[17] Django, [En línea]. Available: https://docs.djangoproject.com/en/3.2/. [Último acceso: 10 9 2021].

[18] ORM, [En línea]. Available: https://www.fullstackpython.com/object-relational-mappers-orms.html. [Último acceso: 10 9 2021].

[19] XAMPP, [En línea]. Available: https://www.apachefriends.org/es/faq_windows.html. [Último acceso: 10 9 2021].

[20] Apache server, [En línea]. Available: https://httpd.apache.org/. [Último acceso: 10 9 2021].

[21] MySQL, [En línea]. Available: https://www.mysql.com/. [Último acceso: 10 9 2021].

[22] PhpMyAdmin, [En línea]. Available: https://www.phpmyadmin.net/. [Último acceso: 10 9 2021].

[23] Visual studio code, [En línea]. Available: https://code.visualstudio.com/. [Último acceso: 10 9 2021].

[24] Git, [En línea]. Available: https://git-scm.com/doc. [Último acceso: 10 9 2021].

[25] Virtual enviorments in Python, [En línea]. Available: https://docs.python.org/3/library/venv.html. [Último acceso: 10 9 2021].

[26] Modelo Entidad Relacion, [En línea]. Available: https://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n. [Último acceso: 10 9 2021].

[27] Rentabilidades Acumuladas, [En línea]. Available: https://www.bestinver.es/terminos/rentabilidad-acumulada/. [Último acceso: 10 9 2021].

[28] Rentabilidad YTD, [En línea]. Available: https://es.wikipedia.org/wiki/YTD. [Último acceso: 10 9 2021].

[29] Volatilidad, [En línea]. Available: https://economipedia.com/definiciones/volatilidad.html. [Último acceso: 10 9 2021].

[30] Arquitectura cliente/servidor, [En línea]. Available: https://www.ecured.cu/Arquitectura_Cliente_Servidor. [Último acceso: 10 9 2021].

[31] P. Decorador. [En línea]. Available: https://refactoring.guru/es/design-patterns/decorator. [Último acceso: 10 9 2021].

[32] Google Fonts, [En línea]. Available: https://fonts.google.com/. [Último acceso: 10 9 2021].

[33] Flask Session, [En línea]. Available: https://testdriven.io/blog/flask-sessions/. [Último acceso: 10 9 2021].

[34] Flask-Bcrypt, [En línea]. Available: https://flask-bcrypt.readthedocs.io/en/latest/. [Último acceso: 10 9 2021].

[35] Código ISIN, [En línea]. Available: https://www.cnmv.es/portal/ANCV/CodigoISIN.aspx. [Último acceso: 10 9 2021].

[36] URL, [En línea]. Available: https://developer.mozilla.org/es/docs/Learn/Common_questions/What_is_a_URL. [Último acceso: 10 9 2021].

[37] API, [En línea]. Available: https://www.xataka.com/basics/api-que-sirve. [Último acceso: 10 9 2021].

Anexo I. Guía de uso de la aplicación

Un usuario no registrado tiene acceso únicamente a las vistas de inicio de sesión y registro de la aplicación, por lo que en primer lugar es necesario registrarse para hacer uso de ella.

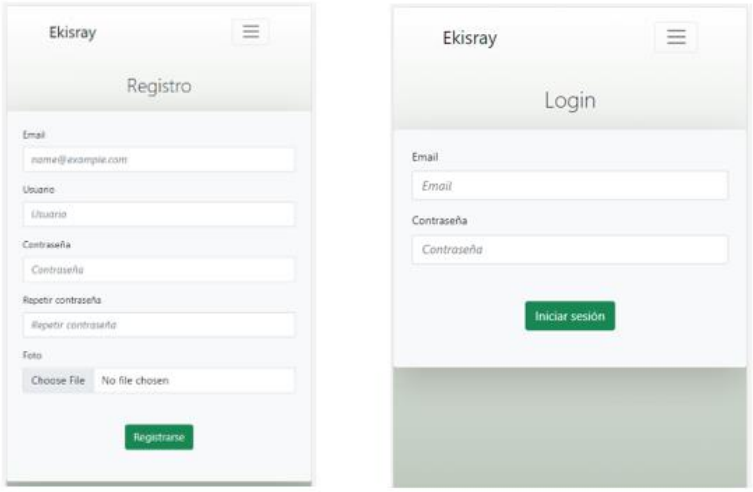

Figura 44 - Vista de Registro y de Login

En caso de intentar acceder al resto de opciones del menu se mostrara un mensaje de error y el usuario sera redirigido a la vista de iniciar sesion. Una vez que nos hemos registrado en la aplicación podemos hacer uso del resto de sus funcionalidades.

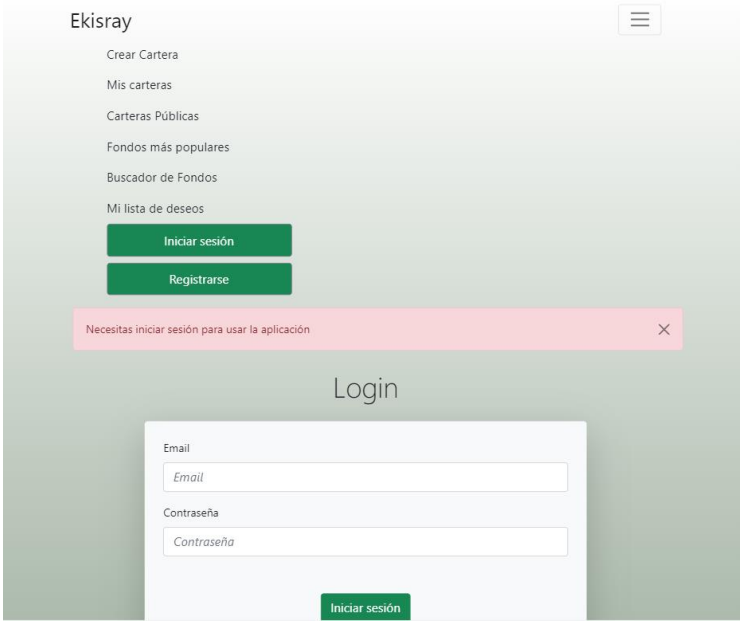

Figura 45 - Mensaje de error para usuarios no registrados

La primera vista que nos aparece una vez que iniciamos sesión es la vista de crear cartera(Figura 46). En dicha vista podemos introducir los códigos y pesos de los fondos que componen nuestra cartera. Si todavía no sabemos que fondos nos interesa incluir en nuestra cartera debemos ir en primer lugar al buscador de fondos(Figura 47).

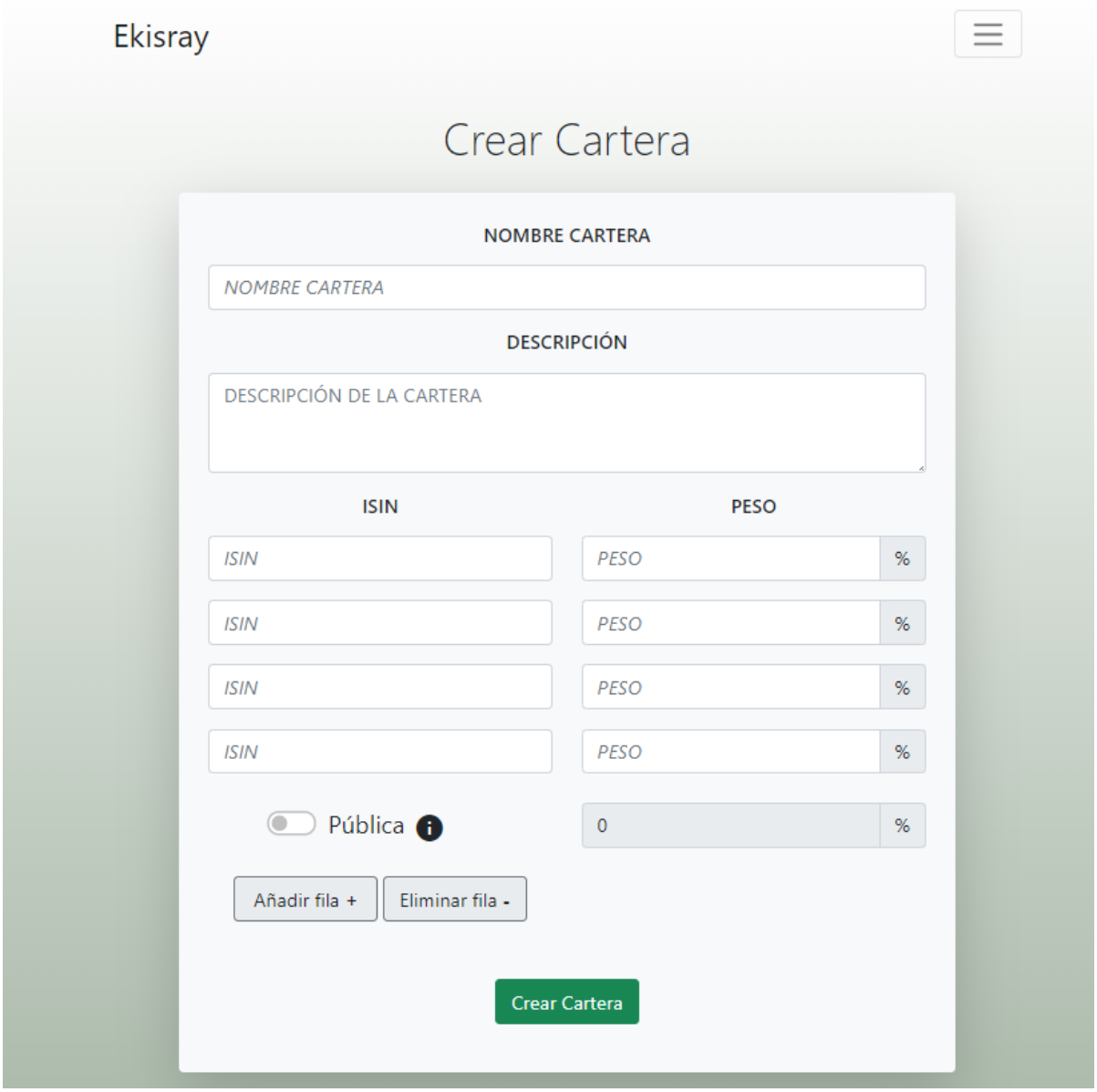

Figura 46 - Vista de Crear una cartera

En el buscador de fondos, podemos buscar cualquier fondo por nombre o código de identificación, además de por temáticas introduciendo palabras clave como “tech”, dicho ejemplo se muestra en la Figura 47.

Figura 47 - Vista del Buscador de Fondos

Para guardar un fondo en una lista llamada “lista de deseos” solo hay que pulsar sobre el nombre de un fondo en los resultados que nos ha proporcionado el buscador, como se observa en la Figura 48.

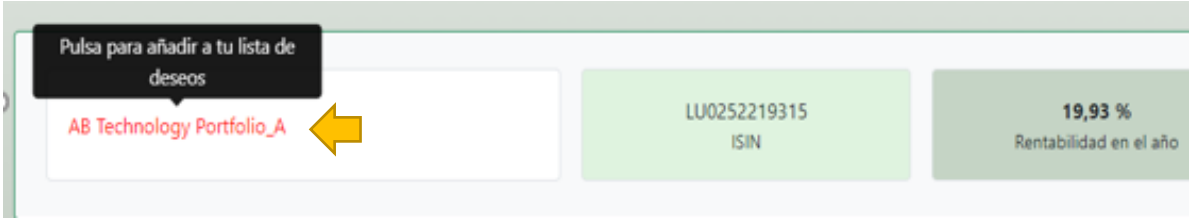

Figura 48 – Vista Añadir Fondo a Lista de Deseos

Una vez que hemos añadido los fondos que nos interesan, accedemos a la lista de deseos mediante el enlace “Mi lista de deseos” del menú(Figura 49) y guardamos los cambios. La lista de deseos podemos modificarla a nuestro gusto, eliminando y añadiendo fondos. En la Figura 50 se puede observar la lista de deseos, a la que hemos añadido 4 fondos.

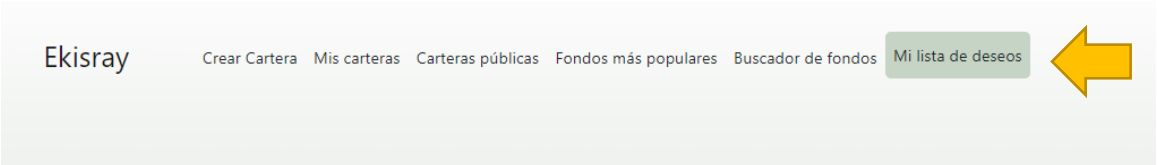

Figura 49 - Enlace a Mi Lista de Deseos

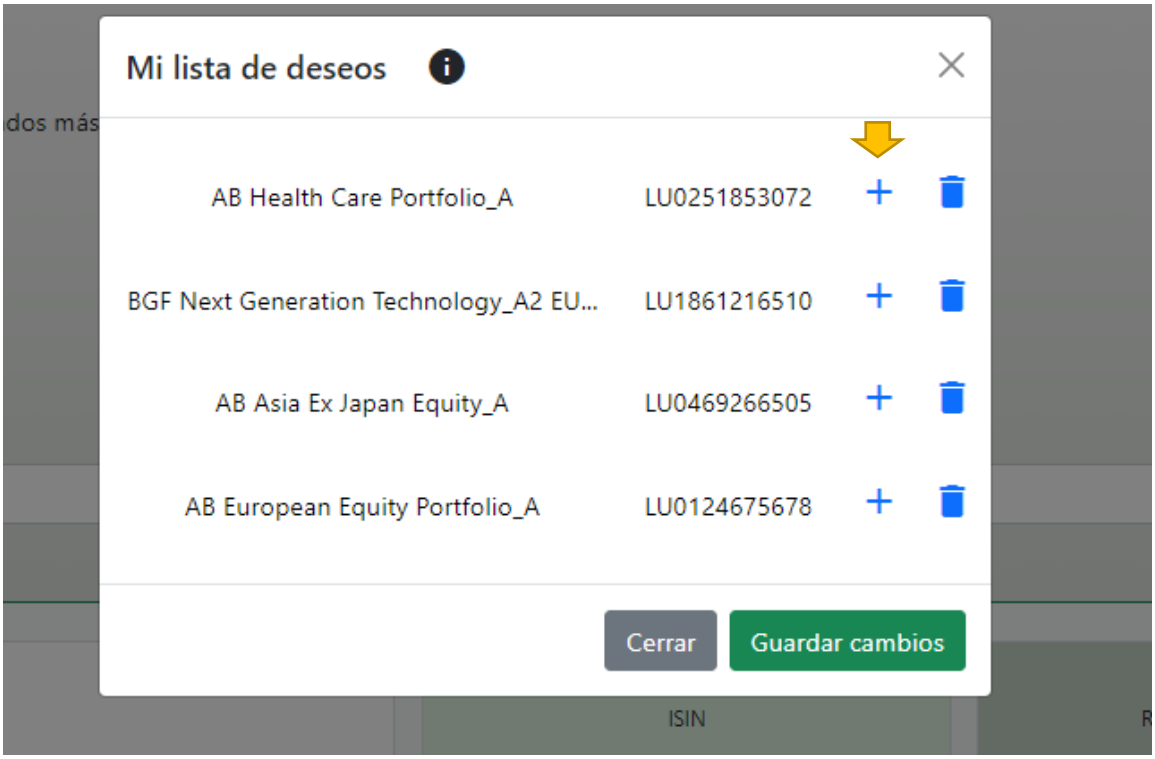

Figura 50 - Vista de la Lista de Deseos

Ya sabemos que fondos nos interesan, porque los hemos guardado en la lista de deseos o porque hemos encontrado sus códigos identificativos en internet, ahora solo nos queda añadirlos al formulario de crear cartera. Podemos añadirlos directamente desde la lista de deseos, pulsando en el símbolo “+”(Figura 50) o escribiendo sus códigos en el formulario.

Una vez que hemos añadido los fondos que nos interesan ya solo falta introducir los pesos que van a representar en nuestra cartera, el nombre de la cartera, la descripción y si queremos que sea pública o privada. Un ejemplo del formulario completo se muestra en la Figura 51.

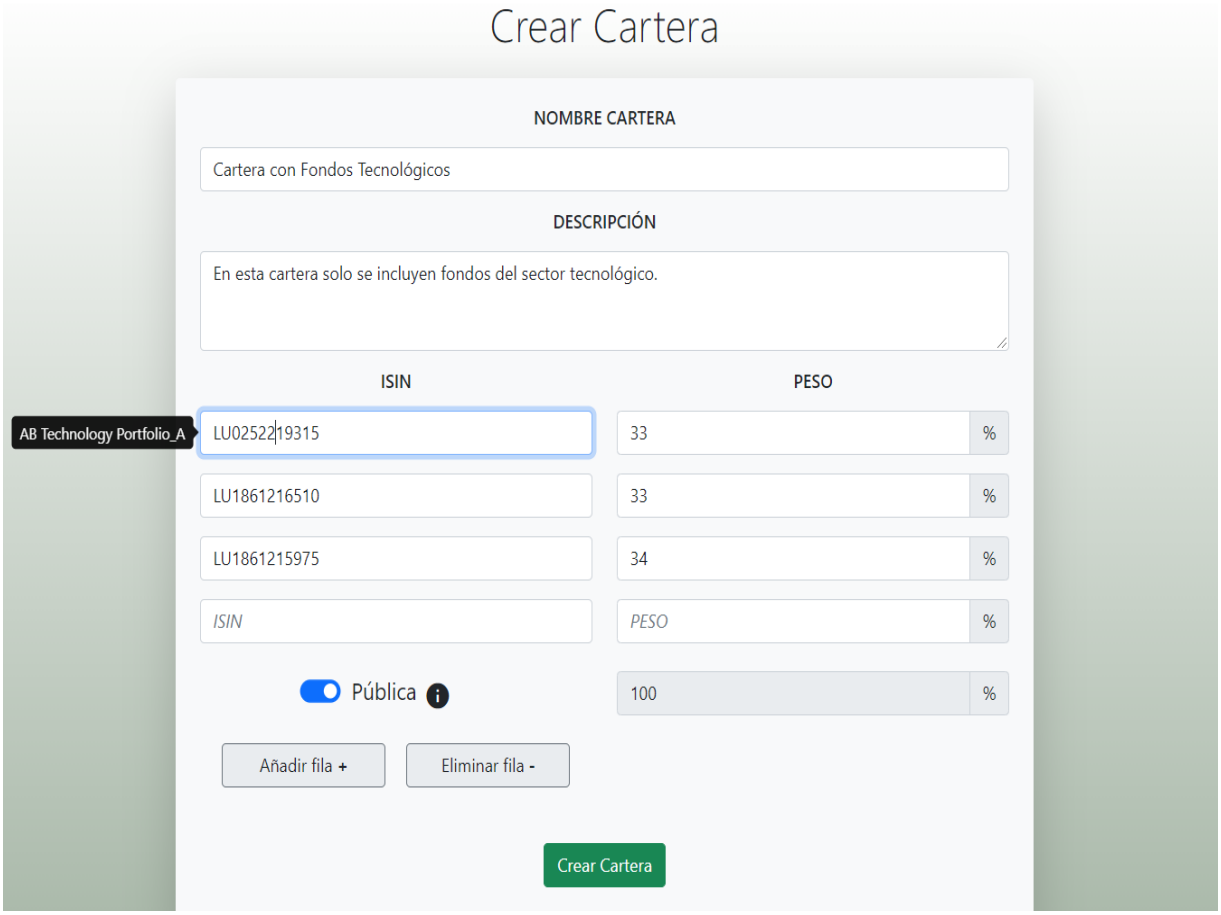

Figura 51 - Ejemplo de formulario completo para crear una cartera

De esta forma podemos crear multitud de ideas de carteras para despues visualizarlas en forma de lista en la opcion del menu “Mis carteras”(Figura 52).

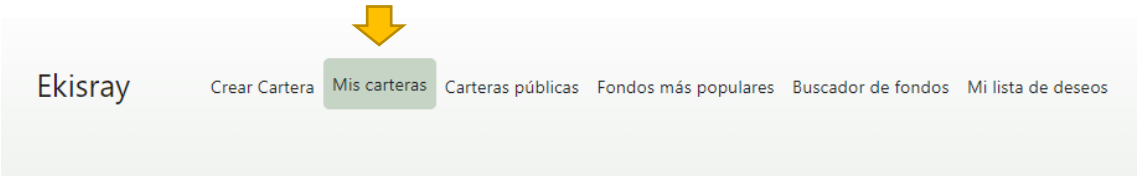

Figura 52 - Enlace a Mis Carteras

“Mis carteras” contiene una lista con todas las carteras que hemos creado hasta el momento, mostrando de cada cartera los 4 fondos de mayor peso y sus parametros mas relevantes. La lista la podemos ordenar por todos y cada uno de los parametros, por lo que es de gran utilidad para comparar carteras. Ademas en esta lista podemos editar o borrar cualquier cartera. Un ejemplo de listado de carteras se muestra en la Figura N.

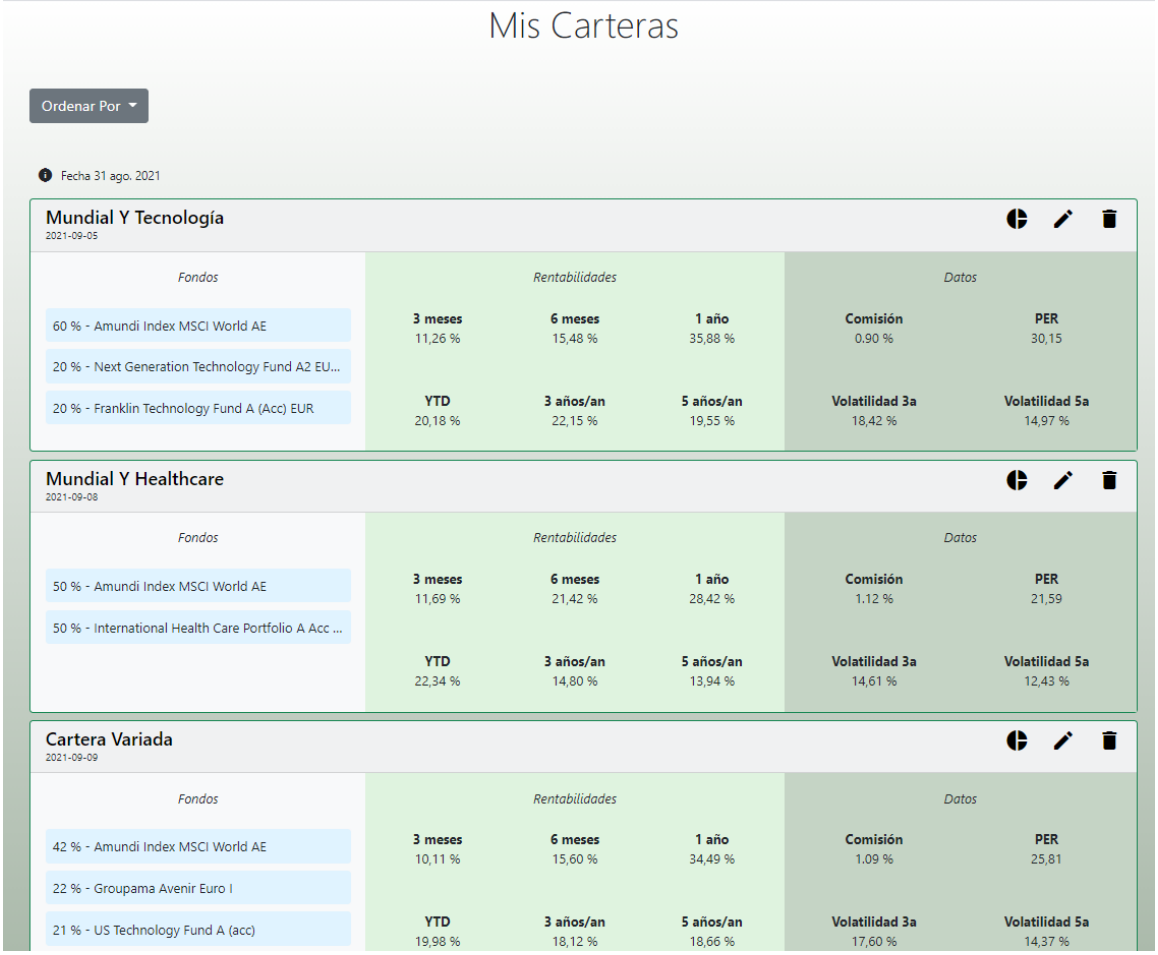

Figura 53 - Ejemplo de Vista de Mis Carteras

Una de las funcionalidades mas importantes de la aplicación es el poder ver en detalle la composicion o estructura de una cartera. En nuestra lista de carteras podemos visualizar la composicion al detalle de cada una de nuestras carteras pulsando en el primer icono, el cual tiene forma de circulo(Figura 54). Los otros dos iconos sirven para la edicion y el borrado de las carteras.

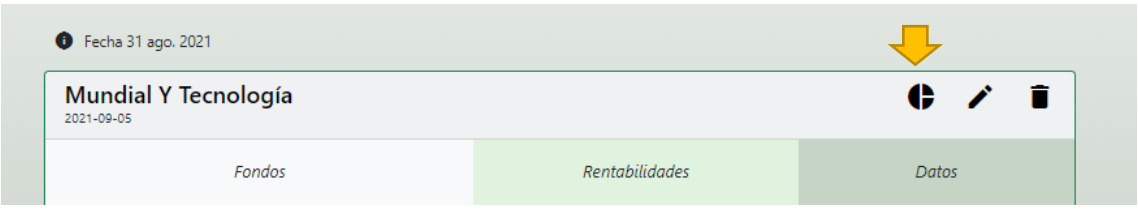

Figura 54 - Icono para acceder a los detalles de un cartera

Un ejemplo de los detalles de una cartera se puede observar en la Figura 55 que se muestra a continuacion.

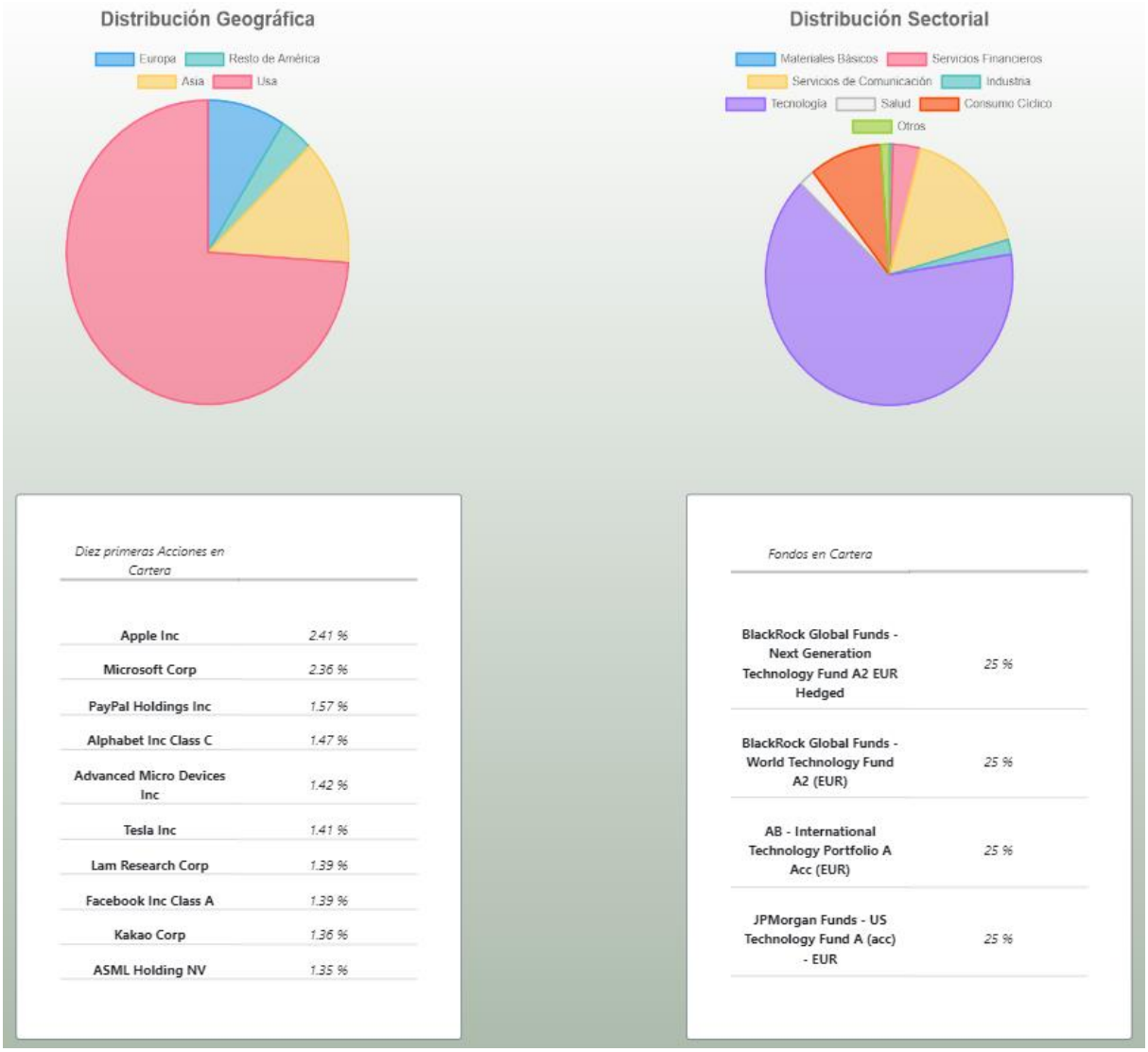

Figura 55 - Ejemplo de la vista en detalle de una cartera

Si queremos ver las carteras publicas del resto de usuarios podemos acceder a la opcion del menu “Carteras Publicas”(Figura 56), en la cual tendremos acceso a todas las carteras que han hecho publicas los usuarios de la aplicación, pudiendolas ordenar tambien por sus parametros mas relevantes, ademas de filtrarlas por nombre. Podemos visualizar la vista de carteras publicas en la Figura 57.

Figura 56 - Enlace a Cartera públicas

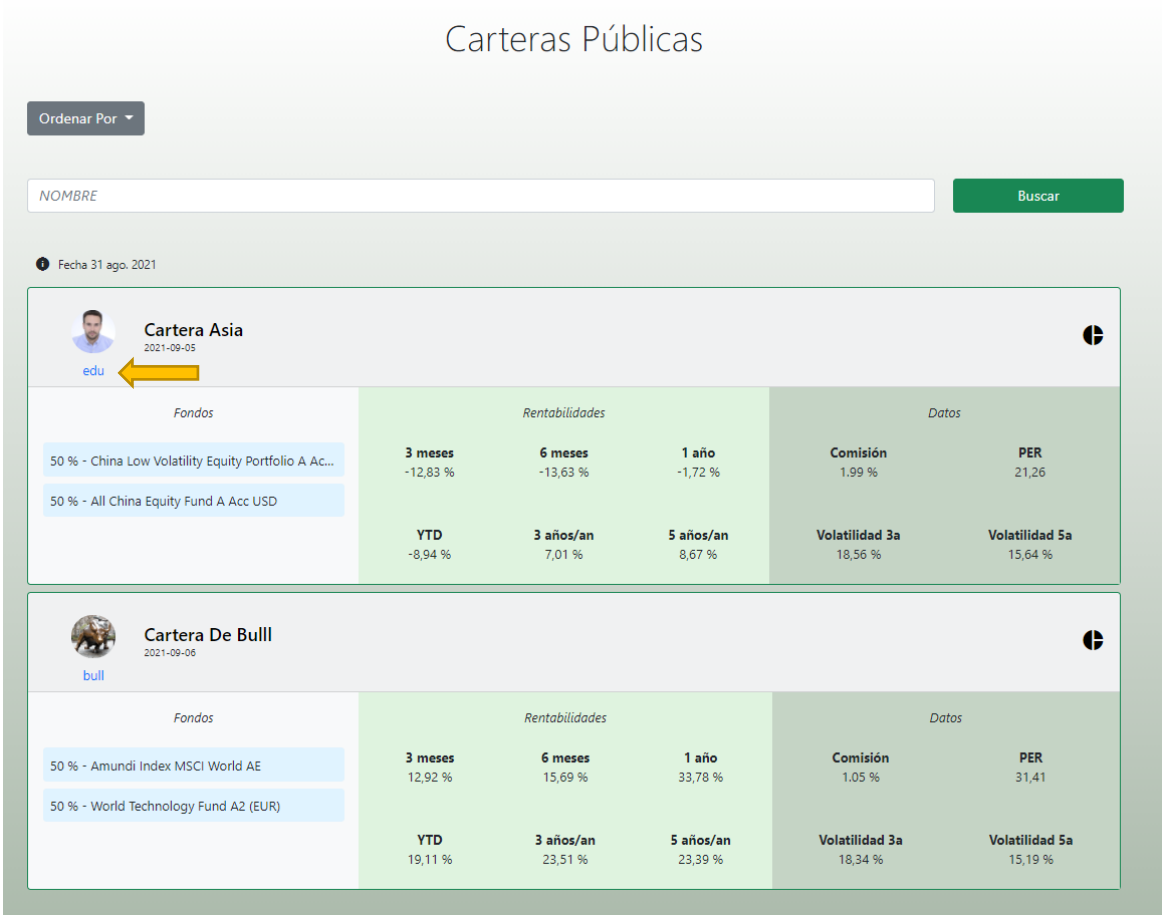

Figura 57 - Ejemplo de vista de carteras públicas

En la lista de carteras publicas podemos acceder a los perfiles de los usuarios que han publicado sus carteras, pulsando sobre su nombre de usuario (Figura 57). En un perfil(Figura 58), ademas del email y el nombre del usuario, aparece el listado de carteras publicadas por el usuario, informacion muy interesante para conocer las carteras publicas de un usuario en particular.

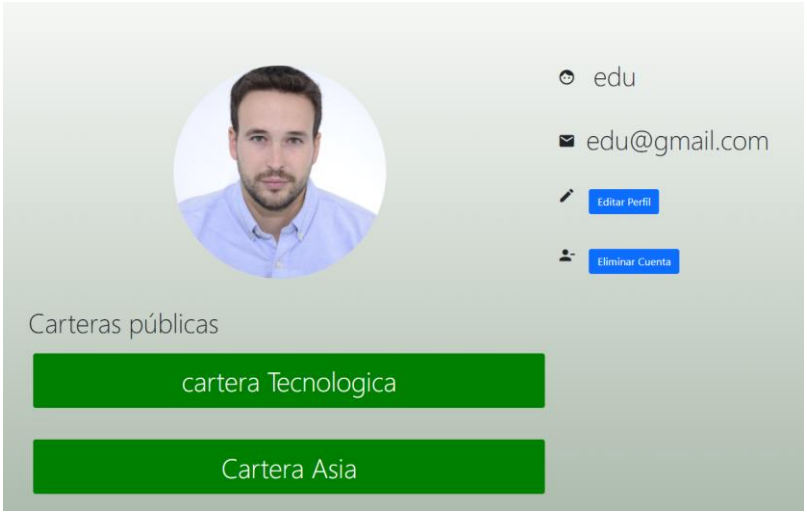

Figura 58 - Ejemplo de perfil de usuario

De cada cartera publica se pueden ver sus detalles y comentarla. Se puede observar un comentario realizado en una cartera publica en la Figura 59.

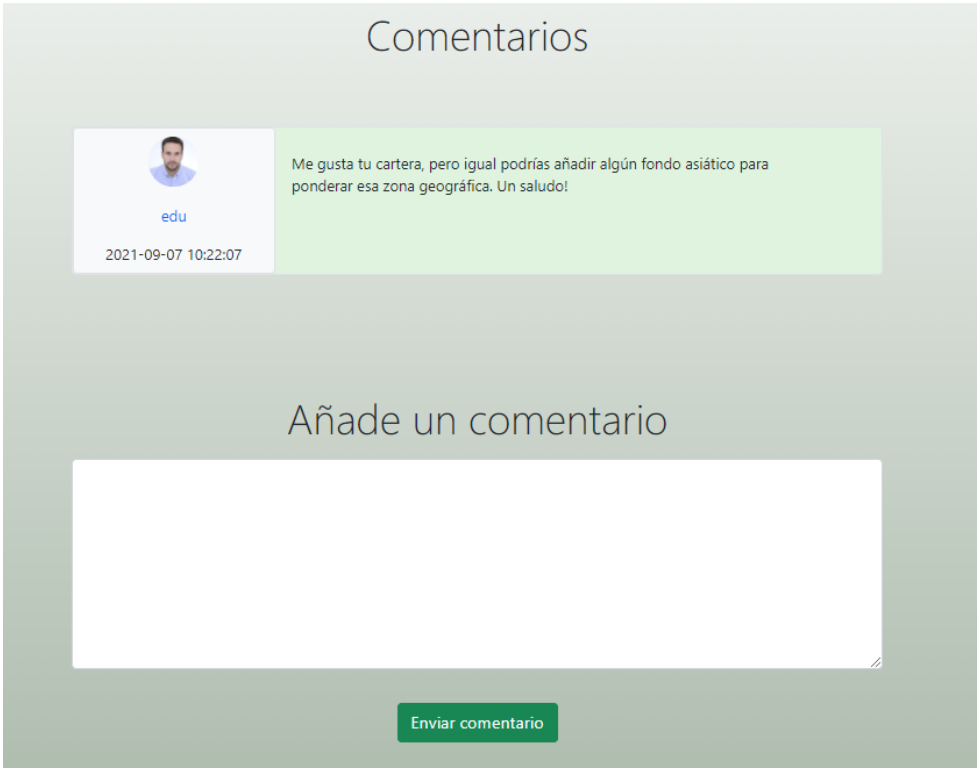

Figura 59 - Comentario realizado en cartera pública

Por ultimo podemos ver los fondos mas populares de la aplicación en la opcion del menu "Fondos mas populares"(Figura 60), como se muestra en la Figura 61.

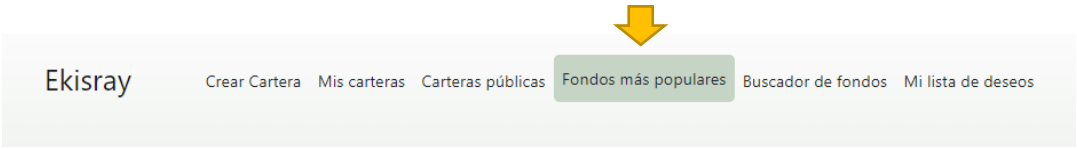

Figura 60 - Enlace a fondos más populares

Figura 61 - Vista de los fondos más populares