UNIVERSIDAD DE OVIEDO

ESCUELA UNIVERSITARIA DE INGENIERÍA TÉCNICA EN INFORMÁTICA DE OVIEDO

PROYECTO FIN DE CARRERA

"AGPA: Aplicación para Gestión de Portafolio de Acciones"

DIRECTOR: Alberto Manuel Fernández Álvarez

AUTOR: Cristóbal Solar Fernández

Vº Bº del Director del Proyecto

Agradecimientos

A mis padres y a mi hermana por su apoyo constante, por enseñarme que siempre hay que intentar terminar lo que uno empieza y a mi tutor Alberto por su paciencia infinita.

Resumen

Este proyecto surge de la necesidad de saber tomar la mejor decisión ante una futura posible inversión en el mundo bursátil.

En un momento dado una persona podrá sentirse atraída por las inversiones en bolsa, y no saber qué pasos dar para atacar el problema. El objetivo de “AGPA” es darle al usuario un pequeño espacio personal donde poder trabajar sus inversiones de la mejor manera posible.

Esta aplicación web permite al usuario crear su propio índice bursátil de empresas con información relevante a ellas, crear sus propias operaciones bursátiles virtuales, acceder a las noticias relacionadas con las empresas pudiendo saber cuándo tenemos noticias nuevas y mantener un historial de nuestras operaciones.

Palabras Clave

Índice bursátil, Python, Angular, Asíncrono, Corrutinas, Portafolio de acciones.

Abstract

This project arises from the need to know how to make the best decision regarding a possible future investment in the stock market world.

At a given moment, a person will feel attracted to investments in the stock market and will not know what steps to take to attack the problem. The objective of "AGPA" is to give the user a small personal space where they can work their investments in the best possible way.

This web application allows the user to create their own stock market index of companies with information relevant to them, create their own virtual stock operations, access news related to companies, being able to know when we have new news and maintain a history of our operations.

Keywords

Stock Index, Python, Angular, Asynchronous, Coroutines, Stock Portfolio.

Índice

1. Introducción 19
1.1. Justificación del proyecto 19
1.2. Objetivos del proyecto 19
1.3. Estudio de la situación actual 20
1.4. Solución propuesta 20
1.5. Estudio de alternativas 20
1.5.1. Finviz.com 20
1.5.2. Yahoo Finance 21
1.5.3. Morningstar 22
2. Aspectos teóricos 24
2.1. Shares 24
2.2. Ticker 24
2.3. Buy/Sell 24
2.4. Mercado bursátil 24
2.5. Open Price 24
2.6. Last Price 24
2.7. Profit 24
2.8. Profitability 24
2.9. Portfolio 25
2.10. Stock index 25
2.10.1. Ibex35 25
2.11. Balance 26
3. Planificación del proyecto y presupuesto 27
3.1. Planificación 27
3.1.1. Planificación general 27
3.1.2. Planificación inicial 28
3.1.3. Planificación final 31
3.1.4. Resumen de categorías 35
3.2. Presupuesto 36
3.2.1. Factores de amortización 36
3.2.2. Presupuesto de costes 36
3.2.3. Presupuesto para el cliente 37
4. Análisis 38
4.1. Identificación de actores del sistema 38

4.2. Requisitos del sistema 38
4.2.1. Requisitos funcionales 38
4.2.2. Requisitos no funcionales 39
4.3. Especificación de Casos de Uso 40
4.3.1. Diagrama de contexto 40
4.3.2. Casos de Uso 41
4.4. Análisis de interfaces de usuario 50
4.4.1. Primera versión 51
4.4.2. Segunda versión 60
4.5. Trazabilidad Interfaces de usuario – Casos de uso 62
4.6. Especificación del plan de pruebas 63
4.6.1. Caso de uso 1: Autentificarse 63
4.6.2. Caso de uso 2: Salir 63
4.6.3. Caso de uso 3: Registrarse 64
4.6.4. Caso de uso 4: Recuperar contraseña 64
4.6.5. Caso de uso 5: Cambiar contraseña 64
4.6.6. Caso de uso 6: Borrar cuenta 64
4.6.7. Caso de uso 7: Vista de Portafolio 65
4.6.8. Caso de uso 8: Crear índice personal 65
4.6.9. Caso de uso 9: Borrar empresa en el índice personal 65
4.6.10. Caso de uso 10: Borrar índice personal 65
4.6.11. Caso de uso 11: Crear operación en el portafolio 65
4.6.12. Caso de uso 12: Cerrar operación en el portafolio 66
4.6.13. Caso de uso 13: Filtrar empresa en el historial 66
4.6.14. Caso de uso 15: Seleccionar noticias 66
4.6.15. Caso de uso 16: Actualizar estado del usuario 66
4.6.16. Caso de uso 17: Actualizar balance del usuario 67
5. Diseño del sistema 68
5.1. Justificación de la plataforma seleccionada 68
5.1.1. Python 68
5.1.2. FastApi 69
5.1.3. Angular 70
5.1.4. Typescript 71
5.1.5. MongoDB 71
5.2. Arquitectura del sistema 72
5.2.1. Patrones de diseño 72

5.2.2. Diagrama de componentes 75
5.3. Diseño de la base de datos 76
5.3.1. Descripción del SGBD NoSQL usado 76
5.3.2. Modelo conceptual 77
6. Implementación del sistema 78
6.1. Estándares y normas seguidas 78
6.1.1. Python PEP 8 78
6.1.2. TypeScript Style 78
6.2. Herramientas y programas usados para el desarrollo 78
6.2.1. Pycharm 78
6.2.2. Visual Studio Code 79
6.2.3. MongoDB Compass 79
6.3. Librerías y frameworks utilizados 80
6.3.1. Yfinance 80
6.3.2. Newspaper3k 80
6.3.3. Docker 80
6.3.4. Pytest-asyncio 80
6.3.5. RxJS 81
6.3.6. Motor 81
6.3.7. Ng2-charts 81
6.3.8. Httpx 81
6.3.9. Uvicorn 82
6.3.10. Nginx 82
6.3.11. Bootstrap 83
6.3.12. Angular Material 83
6.4. Creación del sistema 84
6.4.1. Problemas encontrados 84
6.4.2. Diagrama de paquetes 86
6.5. Descripción detallada de las clases 87
6.5.1. Backend 87
6.5.2. Frontend 90
7. Desarrollo de las pruebas 93
7.1. Pruebas de aceptación 93
7.1.1. Primer pase de la batería de pruebas 93
7.1.2. Segundo pase de la batería de pruebas 95
7.2. Pruebas unitarias 97

7.2.1. Capa de servicio 98
7.2.2. Capa de los endpoints 101
7.2.3. Capa de la persistencia 105
7.2.4. Test Agregador 108
8. Manuales del sistema 109
8.1. Manual de despliegue 109
8.1.1. Base de datos 109
8.1.2. Backend 109
8.1.3. Frontend 110
9. Conclusiones y ampliaciones 112
9.1. Conclusiones 112
9.2. Ampliaciones 112
10. Referencias Bibliográficas 114
11. Anexos 115
11.1. Contenido entregado 115
11.1.1. Aplicación desplegada 115
11.1.2. Backend 115
11.1.3. Frontend 116
11.1.4. Agregador 117
11.1.5. Código fuente 118
11.1.6. Record_route.py 118
11.1.7. Record_service.py 119
11.1.8. Record_persistence.py 120

Índice de figuras

Figura 1: Captura de la página web Finviz 21
Figura 2: Captura de la página web de Yahoo Finance 22
Figura 3: Captura de la página web de MorningStar 23
Figura 4: Captura del índice Ibex35 26
Figura 5: Planificación general 27
Figura 6: Estudios iniciales y aprendizaje de la planificación inicial 28
Figura 7: Gantt fases de estudios iniciales y aprendizaje fase inicial 28
Figura 8: Fase de análisis de la planificación inicial 29
Figura 9: Gantt fase de análisis planificación inicial 29
Figura 10: Fase de diseño de la planificación inicial 30
Figura 11: Gantt fase de diseño de la planificación inicial 30
Figura 12: Fase de implementación de la planificación inicial 31
Figura 13: Gantt de la fase de implementación de la planificación inicial 31
Figura 14: Fases de estudios iniciales y aprendizaje de la planificación final 31
Figura 15: Gantt de estudios iniciales y aprendizaje de la planificación final 32
Figura 16: Fase de análisis de la planificación final 32
Figura 17: Gantt de la fase de análisis de la planificación final 33
Figura 18: Fase de diseño de la planificación final 33
Figura 19: Gantt de la fase de diseño de la planificación final 34
Figura 20: Fase de implementación de la planificación final 34
Figura 21: Gantt de la fase de implementación de la planificación final 35
Figura 22: Gráfica de las fases 36
Figura 23: Diagrama de contexto 40
Figura 24: Prototipo del Login de la primera versión 51
Figura 25: Prototipo Registro de la primera versión 52
Figura 26: Prototipo Recuperar contraseña de la primera versión 52
Figura 27: Prototipo de la Página principal de la primera versión 53
Figura 28: Prototipo Menú de navegación de la primera versión 54
Figura 29: Prototipo de índice de empresas de la primera versión 55
Figura 30: Prototipo Cambiar contraseña de la primera versión 56
Figura 31: Prototipo Borrar cuenta de la primera versión 57
Figura 32: Prototipo Administrador de la primera versión 57
Figura 33: Prototipo Información empresa de la primera versión 58
Figura 34: Prototipo Buy de la primera versión 59
Figura 35: Prototipo Portafolio de la primera versión 59
Figura 36: Prototipo Historial de operaciones de la segunda versión 60
Figura 37: Prototipo Información empresa de la segunda versión 61
Figura 38: Prototipo Portafolio de la segunda versión 62
Figura 39: Captura Ranking Language. Fuente: IEEE Spectrum 68
Figura 40: Logo de Fastapi 69
Figura 41: Logo de Angular 70
Figura 42: Logo de Typescript 71
Figura: 43 Logo de Mongodb 72
Figura 44: Esquema del patrón MVVM en la aplicación 73
Figura 45: Esquema del patrón Observer en la aplicación 74
Figura 46 Esquema de la arquitectura hexagonal en la aplicación. Fuente: código en casa 74

Figura 47: Diagrama de componentes del backend 75
Figura 48: Diagrama de componentes del frontend 76
Figura 49: Diagrama de las colecciones de la base de datos 77
Figura 50: Captura del programa Pycharm 78
Figura 51: Captura del programa Visual Code 79
Figura 52: Captura del programa Compass 80
Figura 53: Captura del logo de Docker 80
Figura 54: Captura del logo de RxJS 81
Figura 55: Captura del logo de Motor 81
Figura 56: Captura del logo de HTTPx 82
Figura 57: Captura del logo de Uvicorn 82
Figura 58: Captura del logo de Nginx 83
Figura 59: Captura del logo de Bootstrap 83
Figura 60: Captura del logo de Angular Material 83
Figura 61: Diagrama de paquetes del backend 86
Figura 62: Diagrama de paquetes del frontend 87
Figura 63: Pruebas unitarias - admin - capa de servicio 98
Figura 64: Pruebas unitarias - auth - capa de servicio 98
Figura 65: Pruebas unitarias - index - capa de servicio 99
Figura 66: Pruebas unitarias - news - capa de servicio 99
Figura 67: Pruebas unitarias - portfolio - capa de servicio 100
Figura 68: Pruebas unitarias - price - capa de servicio 100
Figura 69: Pruebas unitarias - record - capa de servicio 101
Figura 70: Pruebas unitarias - admin - capa de los endpoints 101
Figura 71: Pruebas unitarias - auth - capa de los endpoints 102
Figura 72: Pruebas unitarias - index - capa de los endpoints 103
Figura 73: Pruebas unitarias - news - capa de los endpoints 103
Figura 74: Pruebas unitarias - portfolio - capa de los endpoints 104
Figura 75: Pruebas unitarias - price - capa de los endpoints 104
Figura 76: Pruebas unitarias - record - capa de los endpoints 105
Figura 77: Pruebas unitarias - admin - capa de la persistencia 105
Figura 78: Pruebas unitarias - auth - capa de la persistencia 106
Figura 79: Pruebas unitarias - index - capa de la persistencia 106
Figura 80: Pruebas unitarias - news - capa de la persistencia 107
Figura 81: Pruebas unitarias - portfolio - capa de la persistencia 107
Figura 82: Pruebas unitarias - record - capa de la persistencia 107
Figura 83: Pruebas unitarias - Agregador 108
Figura 84: Captura del contenedor de docker mongo.yml 109
Figura 85 Captura del logo de Python 110
Figura 86: Arquitectura de capetas importantes del backend 115
Figura 87: Arquitectura de carpetas importantes del frontend 116
Figura 88: Arquitectura de carpetas del agregador de noticias 117

Índice de cuadros

Cuadro 1: Tabla resumen de categorías..... 35
Cuadro 2: Tabla de presupuestos..... 36
Cuadro 3: Tabla de presupuestos finales cliente 37
Cuadro 4: Tabla de requisitos funcionales 38
Cuadro 5: Tabla Requisitos no funcionales 39
Cuadro 6: Tabla trazabilidad interfaces - Casos de uso..... 62

1. Introducción

1.1. Justificación del proyecto

Todos los días millones de personas en todo el mundo se acercan a los mercados financieros en busca de una rentabilidad anual para sus ahorros. No cabe duda de que existen miles y miles de empresas dónde encontrar una buena inversión, pero ese es el problema, donde centralizar toda esa información.

Existen diferentes medios para poder acceder a la información financiera.

- Periódicos
- Radio
- Televisión
- Internet

Gracias a internet existen muchas alternativas que permiten al usuario acceder a la última información actualizada de los mercados financieros donde cada usuario puede hacer uso para encontrar sus mejores oportunidades de inversión, desde portales especializados hasta periódicos online.

Pero la mayoría de estos sitios son muy grandes y no tienen la posibilidad de tener controlada toda esa información por el usuario en la propia página.

1.2. Objetivos del proyecto

El presente proyecto tiene como objetivo la construcción de un prototipo de aplicación web donde el usuario pueda tener un control sobre la información que rodea a una empresa, seguir sus actualizaciones y crear sus propias operaciones virtuales.

Los objetivos para cubrir por la web son:

- Acceder a la información posible de las empresas seleccionadas.
    - Seguimiento diario de la cotización de una acción.
    - Seguimiento diario de las rentabilidades.
    - Acceder a las últimas noticias.
    - Visualización gráfica de sus precios.
- Creación por parte del usuario de sus propias operaciones de bolsa relacionadas con una empresa de forma virtual.
- Acceso al historial de operaciones ya cerradas por el usuario provenientes del portafolio.

1.3. Estudio de la situación actual

En la actualidad existen diferentes portales web que nos permiten acceder a la información financiera y poder crear nuestros portafolios de acciones. Mas allá de los diferentes periódicos que ofrecen esta información se hayan grandes portales web donde la información está más que accesible al usuario.

El principal problema de estos portales es que tienen planes de pago limitándonos algunos accesos a su información o limitando las operaciones abiertas en sus portafolios.

1.4. Solución propuesta

La solución propuesta, antes los problemas comentados en el apartado anterior, consiste en el desarrollo de un portal web donde el usuario pueda tener su propio índice ilimitado de empresas a las que seguir, estar enterado de la actualidad de esas empresas con un control separando la información nueva de la vieja, y todo esto de forma ilimitada y gratuita.

El usuario podrá crear todas las operaciones virtuales sin límites de operaciones que quiera contando con un historial de operaciones ya cerradas.

La solución propuesta está disponible online para hacer pruebas y explicada en el anexo 11.1.1.

1.5. Estudio de alternativas

1.5.1. Finviz.com

Finviz es uno de los mejores portales financieros de la actualidad, acceso a diferentes mercados, accesos a las noticias más actuales, posibilidad de crear tu propio portafolio hasta 50 operaciones simultaneas. Si se quiere ampliar el portafolio y tener acceso a otras informaciones en tiempo real se tiene que pagar un plan mensual de $24.96/mes.

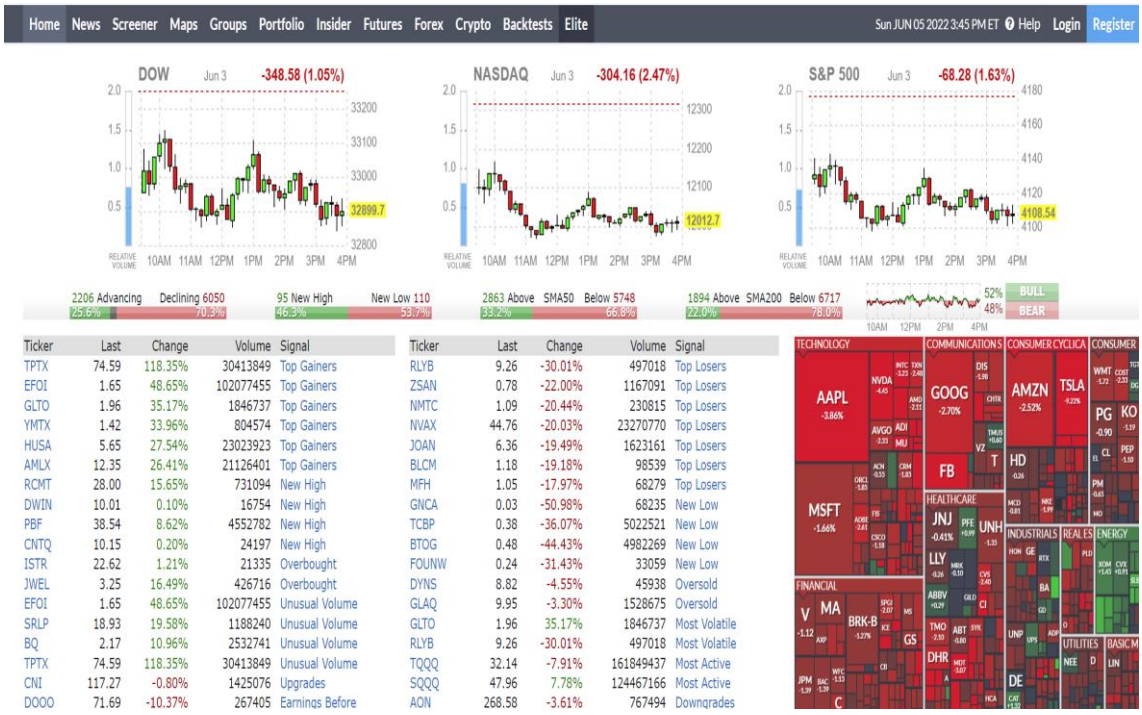

Figura 1: Captura de la página web Finviz[1]

1.5.2. Yahoo Finance

Yahoo Finance es uno de los portales más conocidos en la actualidad, con muchos años funcionando a un alto nivel, el portal te proporciona toda la información de actualidad referente a las empresas, con la opción de poder llevar un seguimiento de las compañías que el usuario quiera y crear tu propio portafolio.

Uno de los problemas que tiene este portal es que no te permite llevar un registro de las noticias leídas y dependiendo de la compañía elegida las noticias pueden ser más bien escasas.

[1] https://finviz.com/

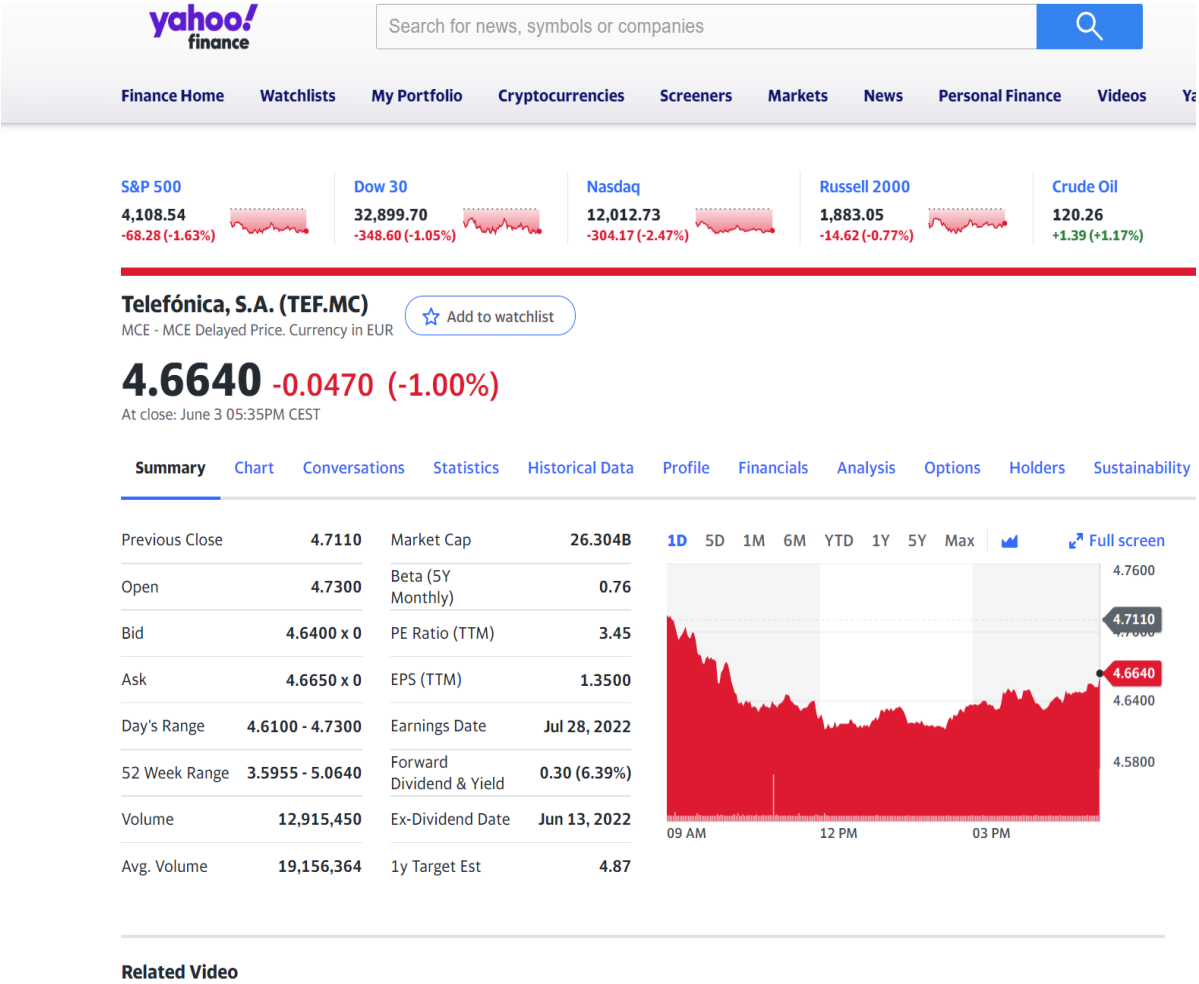

Figura 2: Captura de la página web de Yahoo Finance[2]

1.5.3. Morningstar

Otro gran portal parecido a Yahoo Finance, tienes acceso a mucha información bursátil pero no brinda controles en la información ya leída y si quieres aprovechar todos sus recursos te cobran $249 al año. Lo cual es un muy buen precio para toda la información que proveen.

[2] https://finance.yahoo.com/

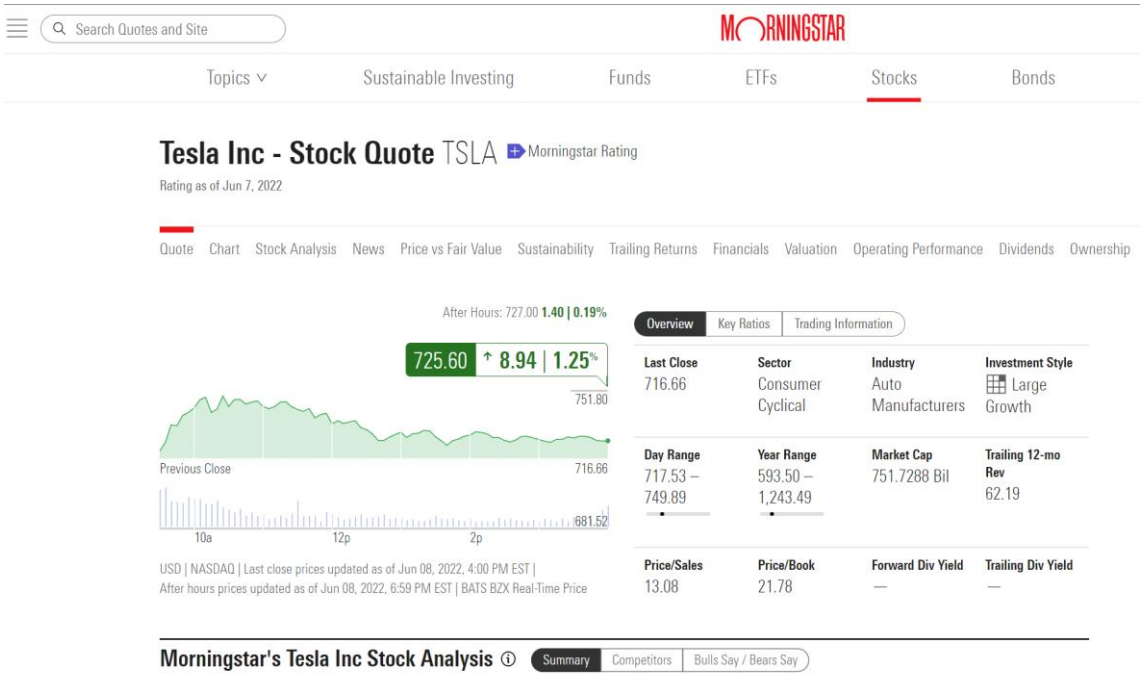

Figura 3: Captura de la página web de MorningStar[3]

[3] https://www.morningstar.com/

2. Aspectos teóricos

2.1. Shares

Shares aquí se traduce como acciones. Una acción representa cada una de las partes en las que se divide el capital de una sociedad anónima.

2.2. Ticker

El ticker es una abreviatura que se usa para identificar de forma única a las empresas que coticen en cualquier mercado bursátil. Siempre suele estar escrito en mayúsculas.

2.3. Buy/Sell

Es el tipo de operación.

Si el usuario elige “buy” cree que el precio va a subir y ganaría dinero con la diferencia entre el precio de salida y el precio de entrada.

Si el usuario elige “sell” cree que el precio va a bajar y ganaría dinero con la diferencia con el precio de entrada y el precio de salida.

2.4. Mercado bursátil

Es el lugar donde se negocian las acciones de las diferentes empresas que cotizan en bolsa.

2.5. Open Price

El precio de apertura al que se abre una operación. Es el primer precio desde donde se fijará las ganancias o pérdidas de una operación.

2.6. Last Price

El último precio o también llamado precio de cotización, es el precio más actual que tenemos para identificar el valor de una acción y el último precio por el que se pagó dicha acción.

2.7. Profit

Es el beneficio que se obtiene en una operación.

2.8. Profitability

Es la rentabilidad diaria que tiene una acción, en caso de que el mercado esté cerrado mostrará la rentabilidad de la acción del día anterior. Cuando el mercado está abierto, muestra la rentabilidad en el momento actual.

2.9. Portfolio

Un portafolio de acciones es el conjunto de activos financieros que están en propiedad de un inversor.

En la aplicación web, en el portafolio se muestran todas las operaciones virtuales que el usuario ha abierto.

2.10. Stock index

Un índice bursátil es un conjunto de empresas que poseen un valor, que se forma a través de todos los precios de cada una de las empresas que forman este índice.

Existen multitud de índices bursátiles en todo el mundo, los más importantes se negocian en Estados Unidos y el índice más conocido e importante es el S&P 500, el cual lo forman las 500 empresas más importantes de todo el mundo.

2.10.1. Ibex35

El Ibex35 es un índice bursátil que representa las 35 empresas más importantes del mercado español.

IBEX 35®

Nombre	Anterior	Último	% Dif.	Máximo	Mínimo	Fecha	Hora	% Dif. Año 2022
▼ IBEX 35®	8.122,50	8.100,30	-0,27	8.137,90	8.034,10	08/07/2022	17:38:00	-7,04

Nombre	Últ.	% Dif.	Máx.	Mín.	Volumen	Efectivo (miles €)	Fecha	Hora
▲ ACCIONA	189,8000	2,21	191,0000	185,1000	159.468	30.166,39	08/07/2022	Cierre
▲ ACCIONA ENER	39,6000	1,23	39,8200	39,1800	234.054	9.223,45	08/07/2022	Cierre
▲ ACERINOX	8,6300	0,91	8,7320	8,3400	4.492.256	38.111,08	08/07/2022	Cierre
▲ ACS	21,6900	1,78	21,8500	21,2000	678.383	14.638,03	08/07/2022	Cierre
▲ AENA	122,0000	2,18	122,0000	119,3000	153.089	18.569,92	08/07/2022	Cierre
▲ AMADEUS	53,5200	0,11	54,1800	52,9200	1.087.532	58.147,08	08/07/2022	Cierre
▲ ARCELORMIT.	21,9000	0,46	22,0350	21,2550	406.794	8.838,18	08/07/2022	Cierre
▼ B.SANTANDER	2,6130	-0,10	2,6490	2,5715	28.474.361	74.309,60	08/07/2022	Cierre
▼ BA.SABADELL	0,6752	-0,32	0,6814	0,6564	55.442.484	37.405,92	08/07/2022	Cierre
▼ BANKINTER	5,1740	-1,26	5,2700	5,0640	6.510.430	33.953,57	08/07/2022	Cierre
▲ BBVA	4,3970	0,79	4,4400	4,3065	15.262.543	66.934,67	08/07/2022	Cierre
▼ CAIXABANK	3,0010	-0,60	3,0270	2,9180	14.432.091	43.067,89	08/07/2022	Cierre
▲ CELLNEX	38,6000	0,52	38,9700	37,8200	4.717.487	180.430,19	08/07/2022	Cierre
▲ ENAGAS	19,6550	0,28	19,9300	19,5400	1.024.038	20.157,98	08/07/2022	Cierre
▼ ENDESA	18,0750	-0,66	18,2050	17,9850	1.351.615	24.451,80	08/07/2022	Cierre
▲ FERROVIAL	26,0000	2,32	26,0900	25,4900	1.045.383	27.070,92	08/07/2022	Cierre
▲ FLUIDRA	19,7000	1,91	19,7200	19,1400	334.321	6.525,30	08/07/2022	Cierre
▼ GRIFOLS CL.A	16,9000	-0,38	17,1700	16,6300	471.362	7.962,01	08/07/2022	Cierre
▲ IAG	1,3005	0,58	1,3105	1,2675	9.698.728	12.559,56	08/07/2022	Cierre
▼ IBERDROLA	9,9120	-1,57	10,1050	9,9080	10.192.450	101.395,40	08/07/2022	Cierre
▲ INDITEX	23,3300	0,78	23,6000	22,9400	3.257.969	75.950,66	08/07/2022	Cierre
▲ INDRA A	9,4050	1,68	9,4600	9,1850	932.212	8.743,73	08/07/2022	Cierre
▲ INM.COLONIAL	6,0850	0,75	6,1050	5,9750	813.204	4.936,10	08/07/2022	Cierre
▲ MAPFRE	1,6530	0,12	1,6530	1,6240	2.883.680	4.737,56	08/07/2022	Cierre
▲ MELIA HOTELS	5,8250	1,39	5,8550	5,7350	859.180	4.987,38	08/07/2022	Cierre
▲ MERLIN	9,6000	1,69	9,6200	9,3850	654.037	6.260,58	08/07/2022	Cierre
▲ NATURGY	28,0500	0,50	28,5800	27,8600	182.950	5.152,31	08/07/2022	Cierre
▲ PHARMA MAR	69,0400	0,15	69,8000	66,5000	45.320	3.128,04	08/07/2022	Cierre
▼ R.E.C.	18,1250	-0,71	18,4400	18,0400	1.392.944	25.280,93	08/07/2022	Cierre
▼ REPSOL	12,9800	-1,22	13,3650	12,8550	14.033.450	183.564,15	08/07/2022	Cierre
▲ ROVI	60,9000	1,00	61,2000	59,7000	55.333	3.348,20	08/07/2022	Cierre
▲ SACYR	2,3240	1,13	2,3240	2,2840	1.378.215	3.180,72	08/07/2022	Cierre
▲ SIEMENS GAME	17,9200	0,20	17,9200	17,8550	2.288.366	40.944,16	08/07/2022	Cierre
▲ SOLARIA	22,1400	1,28	22,3500	21,6600	281.117	6.202,10	08/07/2022	Cierre
▼ TELEFONICA	4,7200	-0,08	4,7290	4,6430	11.677.403	54.927,19	08/07/2022	Cierre

Figura 4: Captura del índice Ibex35[4]

2.11. Balance

El balance es el saldo virtual que posee un usuario en la aplicación. Este saldo es afectado por las operaciones que el usuario realice en el portafolio.

[4] https://www.bolsamadrid.es/esp/aspx/Mercados/Precios.aspx?indice=ESI100000000

3. Planificación del proyecto y presupuesto

3.1. Planificación

3.1.1. Planificación general

Aquí se muestra la planificación general con las fases más importantes.

Proyecto fin de carrera	**525 días**	**lun 08/06/20**	**vie 10/06/22**
Estudio inicial	**10 días**	**lun 08/06/20**	**vie 19/06/20**
Aprendizaje Angular y Fastapi	14 días	lun 22/06/20	jue 09/07/20
Analisis	**103 días**	**vie 10/07/20**	**mar 01/12/20**
Definición de requisitos	**59 días**	**vie 10/07/20**	**mié 30/09/20**
Especificación de casos de uso	**14 días**	**jue 01/10/20**	**mar 20/10/20**
Interfaces de usuario	20 días	mié 21/10/20	mar 17/11/20
Especificación del plan de pruebas	10 días	mié 18/11/20	mar 01/12/20
Diseño	**28 días**	**mié 02/12/20**	**vie 08/01/21**
Arquitectura del sistema	**27 días**	**mié 02/12/20**	**jue 07/01/21**
Diseño de la base de datos	1 día	vie 08/01/21	vie 08/01/21
Implementación	**300 días**	**lun 11/01/21**	**vie 04/03/22**
Documentación	70 días	lun 07/03/22	vie 10/06/22

Figura 5: Planificación general

3.1.2. Planificación inicial

Proyecto fin de carrera	**280 días**	**lun 08/06/20**	**vie 02/07/21**
Estudio inicial	**8 días**	**lun 08/06/20**	**mié 17/06/20**
Evaluar alternativas	3 días	lun 08/06/20	mié 10/06/20
Justificación de alternativa	3 días	jue 11/06/20	lun 15/06/20
Reuniones	2 días	mar 16/06/20	mié 17/06/20
Aprendizaje Angular	7 días	jue 18/06/20	vie 26/06/20

Figura 6: Estudios iniciales y aprendizaje de la planificación inicial

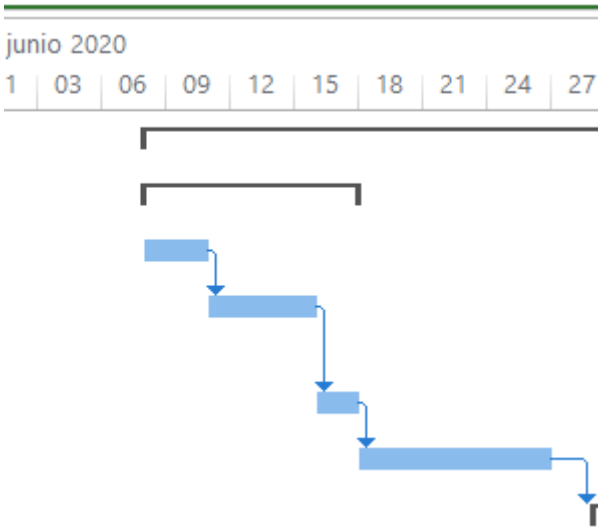

Figura 7: Gantt fases de estudios iniciales y aprendizaje fase inicial

Analisis	**65 días**	**jue 18/06/20**	**mié 16/09/20**
Definición de requisitos	**30 días**	**jue 18/06/20**	**mié 29/07/20**
requisitos funcionales	29 días	jue 18/06/20	mar 28/07/20
requisitos no funcionales	1 día	mié 29/07/20	mié 29/07/20
Especificación de casos de uso	**20 días**	**jue 30/07/20**	**mié 26/08/20**
Diagrama de casos de uso	5 días	jue 30/07/20	mié 05/08/20
Descripción de escenarios de casos de uso	15 días	jue 06/08/20	mié 26/08/20
Interfaces de usuario	10 días	jue 27/08/20	mié 09/09/20
Especificación del plan de pruebas	5 días	jue 10/09/20	mié 16/09/20

Figura 8: Fase de análisis de la planificación inicial

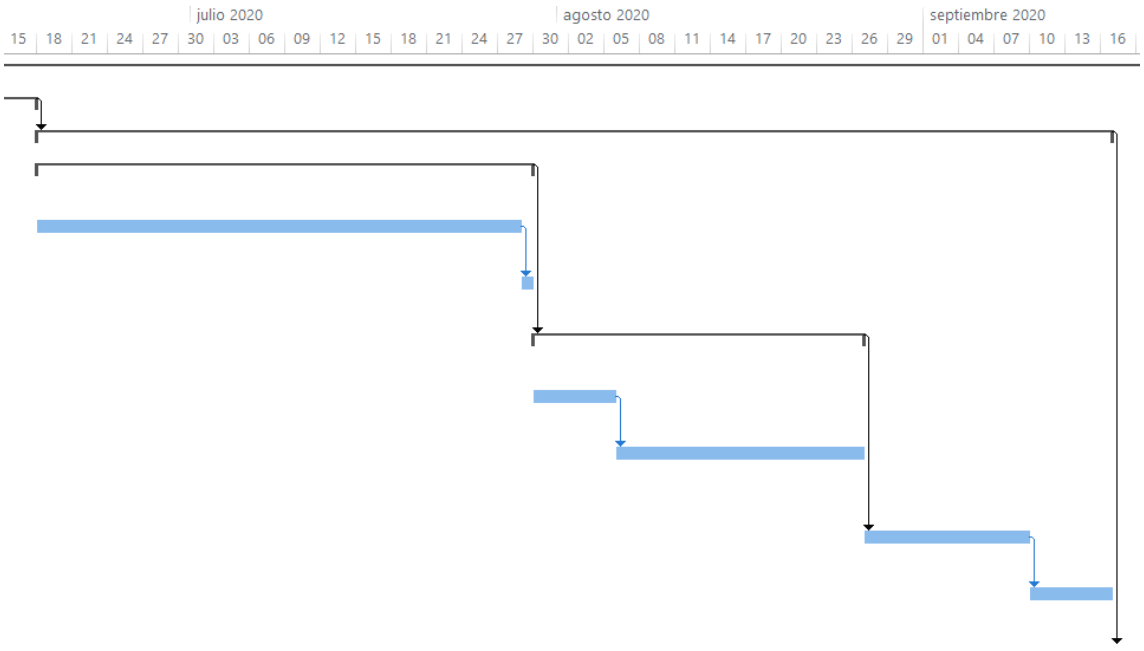

Figura 9: Gantt fase de análisis planificación inicial

Diseño	**15 días**	**jue 17/09/20**	**mié 07/10/20**
Arquitectura del sistema	**14 días**	**jue 17/09/20**	**mar 06/10/20**
Diagrama de componentes	2 días	jue 17/09/20	vie 18/09/20
Diagrama de paquetes	5 días	lun 21/09/20	vie 25/09/20
Diagrama de clases	7 días	lun 28/09/20	mar 06/10/20
Diseño de la base de datos	1 día	mié 07/10/20	mié 07/10/20

Figura 10: Fase de diseño de la planificación inicial

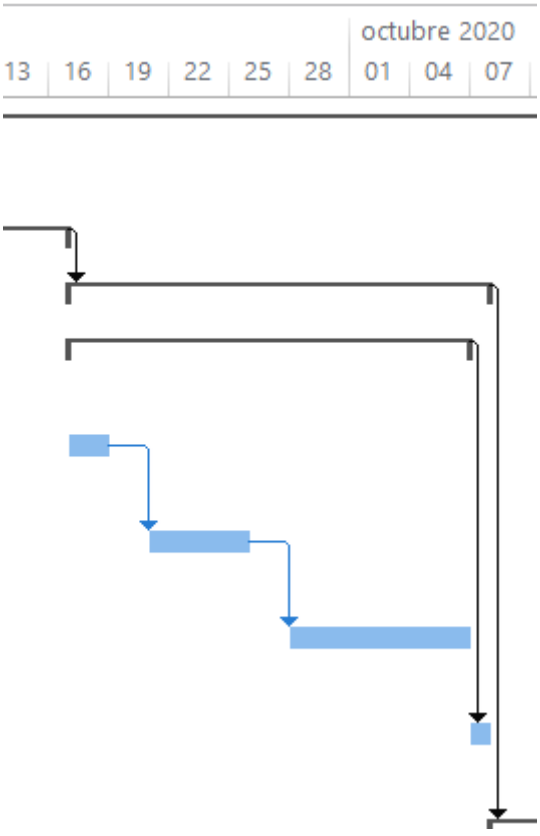

Figura 11: Gantt fase de diseño de la planificación inicial

Implementación	**150 días**	**jue 08/10/20**	**mié 05/05/21**
Backend	100 días	jue 08/10/20	mié 24/02/21
Frontend	100 días	jue 08/10/20	mié 24/02/21
Ejecuccion de pruebas	50 días	jue 25/02/21	mié 05/05/21
Documentación	35 días	jue 06/05/21	mié 23/06/21

Figura 12: Fase de implementación de la planificación inicial

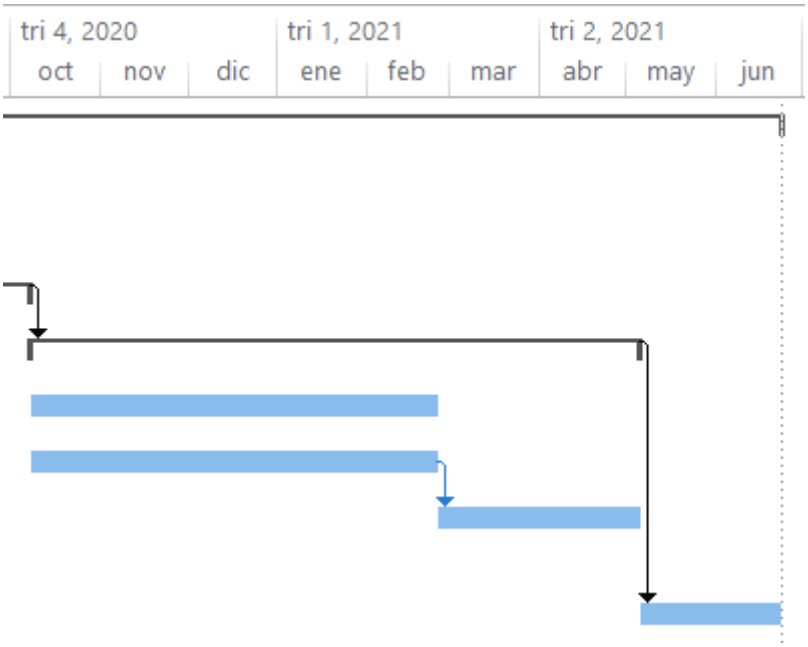

Figura 13: Gantt de la fase de implementación de la planificación inicial

3.1.3. Planificación final

Proyecto fin de carrera	**525 días**	**lun 08/06/20**	**vie 10/06/22**
Estudio inicial	**10 días**	**lun 08/06/20**	**vie 19/06/20**
Evaluar alternativas	3 días	lun 08/06/20	mié 10/06/20
Justificación de alternativa	3 días	jue 11/06/20	lun 15/06/20
Reuniones	4 días	mar 16/06/20	vie 19/06/20
Aprendizaje Angular y Fastapi	14 días	lun 22/06/20	jue 09/07/20

Figura 14: Fases de estudios iniciales y aprendizaje de la planificación final

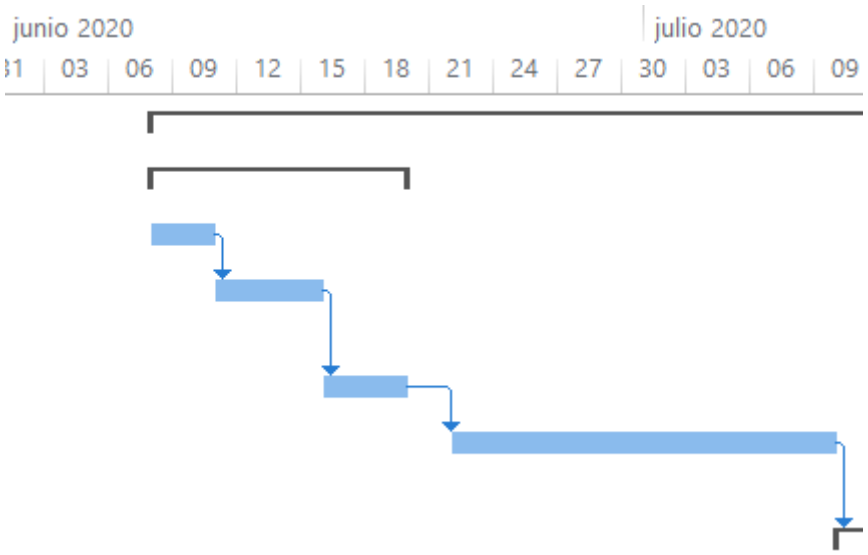

Figura 15: Gantt de estudios iniciales y aprendizaje de la planificación final

Analisis	**103 días**	**vie 10/07/20**	**mar 01/12/20**
Definición de requisitos	**59 días**	**vie 10/07/20**	**mié 30/09/20**
requisitos funcionales	58 días	vie 10/07/20	mar 29/09/20
requisitos no funcionales	1 día	mié 30/09/20	mié 30/09/20
Especificación de casos de uso	**14 días**	**jue 01/10/20**	**mar 20/10/20**
Diagrama de casos de uso	2 días	jue 01/10/20	vie 02/10/20
Descripción de escenarios de casos de uso	12 días	lun 05/10/20	mar 20/10/20
Interfaces de usuario	20 días	mié 21/10/20	mar 17/11/20
Especificación del plan de pruebas	10 días	mié 18/11/20	mar 01/12/20

Figura 16: Fase de análisis de la planificación final

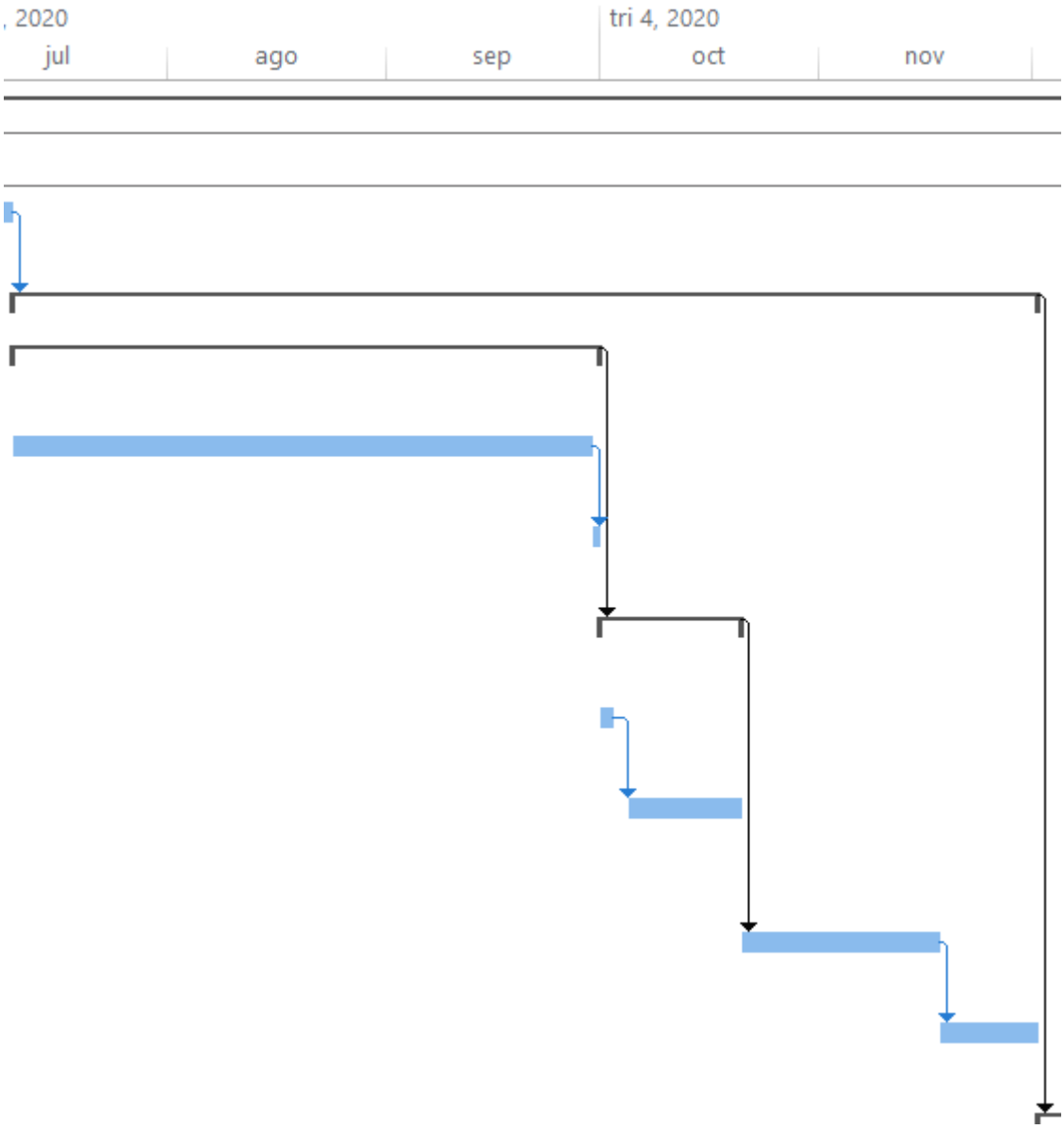

Figura 17: Gantt de la fase de análisis de la planificación final

Diseño	**28 días**	**mié 02/12/20**	**vie 08/01/21**
Arquitectura del sistema	**27 días**	**mié 02/12/20**	**jue 07/01/21**
Diagrama de componentes	3 días	mié 02/12/20	vie 04/12/20
Diagrama de paquetes	10 días	lun 07/12/20	vie 18/12/20
Diagrama de clases	14 días	lun 21/12/20	jue 07/01/21
Diseño de la base de datos	1 día	vie 08/01/21	vie 08/01/21

Figura 18: Fase de diseño de la planificación final

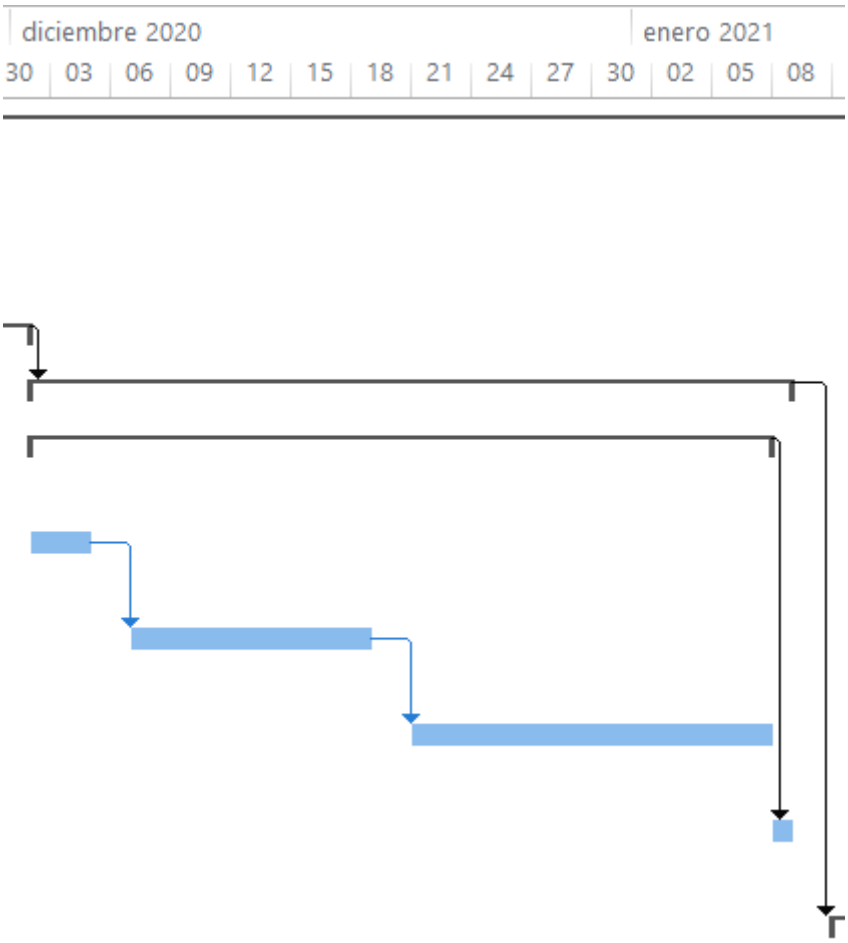

Figura 19: Gantt de la fase de diseño de la planificación final

Implementación	300 días	lun 11/01/21	vie 04/03/22
Backend	200 días	lun 11/01/21	vie 15/10/21
Frontend	200 días	lun 11/01/21	vie 15/10/21
Ejecuccion de pruebas	100 días	lun 18/10/21	vie 04/03/22
Documentación	70 días	lun 07/03/22	vie 10/06/22

Figura 20: Fase de implementación de la planificación final

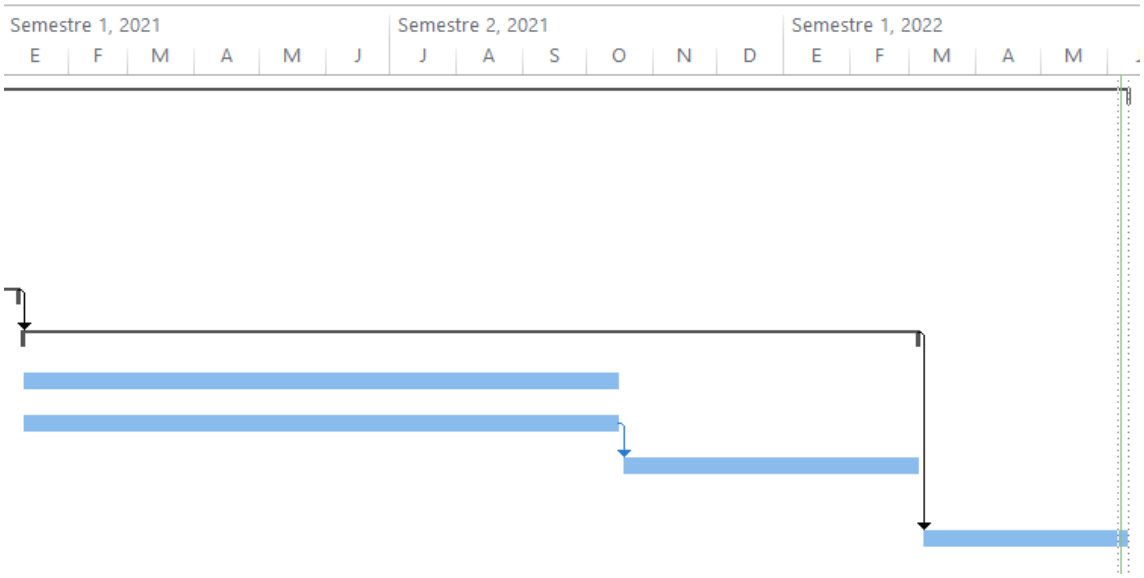

Figura 21: Gantt de la fase de implementación de la planificación final

3.1.3.1. Conclusión

Al ser un proyecto más complejo de los realizados durante la carrera y a mi inexperiencia en la planificación de proyectos, no me fue posible estimar correctamente la duración de las tareas planificadas.

Aparte de lo anterior a los pocos meses de empezar el proyecto tuve una serie de descubrimientos que me proporcionaron crear algo más novedoso en la aplicación por la parte del servidor. Se descubrieron las corrutinas en Python y con ello la posibilidad de crear una aplicación totalmente asíncrona.

3.1.4. Resumen de categorías

Cuadro 1: Tabla resumen de categorías

Fases	Días invertidos	Porcentaje del total
Estudio inicial	10	2%
Aprendizaje	14	3%
Análisis	103	20%
Diseño	28	5%
Implementación	300	57%
Documentación	70	13%
Total días	525	100%

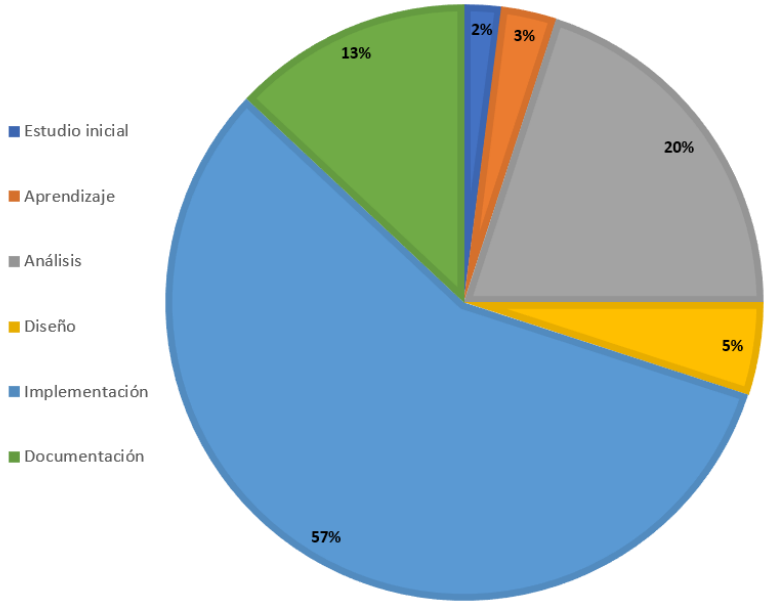

Figura 22: Gráfica de las fases

3.2. Presupuesto

3.2.1. Factores de amortización

Para la elaboración del presupuesto se ha tenido en cuenta las horas empleadas y los gastos de los materiales a la hora de calcular su amortización. En los gastos de elementos de hardware estimamos una vida útil de 5 años y para los elementos de software por sus constantes actualizaciones estimamos una vida útil de 2 años.

El tiempo empleado en la finalización del proyecto ha sido de 2 años.

No se ha calculado en la amortización los gastos de oficina como luz, agua, papel, etc.

3.2.1.1. Fórmulas

```
Amortización anual = Valor de compra / Vida útil estimada

Amortización acumulada = Amortización anual * Años que han
transcurrido desde la compra de los elementos
```

3.2.2. Presupuesto de costes

A continuación, se detalla el presupuesto de costes del proyecto. Este es el presupuesto que utiliza el proveedor del servicio y que posteriormente se usará para calcular el presupuesto del cliente.

Cuadro 2: Tabla de presupuestos

Concepto	Cantidad	Precio unitario	Amortización acumulada	Total

Elementos hardware y software				1.599.59€
Ordenador portátil	1	800,00€	320,00€	480,00€
Servidor VPS	1	336,00€	134,40€	201,60€
Licencia Windows 10 Pro	1	259,00€	259,00€	259,00€
Microsoft Project 2016	1	349,99€	349,99€	349,99€
Enterprise Architect Professional	1	229,00€	229,00€	229,00€
Gastos de oficina	1	80,00€	-	80,00€
Desarrollo de la aplicación				37.450,00€
Análisis	200 horas	40,00€	-	8.000,00€
Diseño	230 horas	40,00€	-	9.200,00€
Implementación	350 horas	30,00€	-	10.500,00€
Pruebas	150 horas	35,00€	-	5.250,00€
Documentación	150 horas	30,00€	-	4.500,00€
Total				39.049,59€

3.2.3. Presupuesto para el cliente

A partir del presupuesto de costes, se genera el presupuesto para el cliente.

En este presupuesto se ocultan los elementos de hardware y software, así como los gastos de oficina, ya que estos no son elementos que vayamos a entregar al cliente.

Para crear este presupuesto prorrateamos el beneficio de la empresa del 15%, imprevistos del 2% y los costes indirectos del proyecto en el precio por hora. Además del cálculo del IVA actual del 21%.

Cuadro 3: Tabla de presupuestos finales cliente

Concepto	Horas	Precio	Total
Análisis	200	48,00€	9.600,00€
Diseño	230	48,00€	11.040,00€
Implementación	350	38,70€	13.545,00€
pruebas	150	44,11€	6.616,50€
Documentación	150	38,70€	5.805,00€
Subtotal			46.606,50€
IVA (21%)			9.787,36€
Total			56.393,87€

4. Análisis

4.1. Identificación de actores del sistema

En el sistema existen tres actores:

- **Usuario**: es la persona que interactúa con la aplicación. Que puede crear su propio índice, crear nuevas operaciones, consultar las noticias de actualidad.
- **Administrador**: además de tener las mismas funcionalidades que el usuario, es la persona encargada de administrar los usuarios, desde activarlos o desactivarlos hasta añadirles más dinero en sus balances.
- **Agregador**: es la aplicación en segundo plano encargada de descargarse las nuevas noticias cada día.

4.2. Requisitos del sistema

4.2.1. Requisitos funcionales

Cuadro 4: Tabla de requisitos funcionales

Código	Nombre	Descripción
RF 1	Gestión de usuarios	La aplicación permitirá gestionar los usuarios que van a interactuar con la aplicación.
RF 1.1	Autentificación	La aplicación debe disponer de un mecanismo de autentificación de usuarios.
RF 1.2	Cerrar sesión	Se podrá salir del sistema de forma manual, cuando se cierre el navegador o automáticamente después de un tiempo de inactividad.
RF 1.3	Selección de vistas	En función del rol del usuario se podrán seleccionar ciertas vistas de la aplicación.
RF 1.4	Consultar usuarios	Listado de usuarios, sus estados y sus balances a petición solamente del administrador.
RF 1.5	Editar estado del usuario	Solo el administrador podrá editar el estado de los usuarios registrados en la aplicación.
RF 1.6	Editar balance del usuario	Solo el administrador podrá editar el balance de los usuarios registrados en la aplicación.
RF 1.7	Registrarse	La aplicación debe disponer de un mecanismo para que los usuarios puedan registrarse en la aplicación.
RF 1.8	Recuperar contraseña	La aplicación debe disponer de un mecanismo para que los usuarios puedan recuperar su contraseña.
RF 1.9	Borrar cuenta	La aplicación debe disponer de un mecanismo para que los usuarios puedan borrar cuenta.

RF 1.10	Cambiar contraseña	La aplicación debe disponer de un mecanismo para que los usuarios puedan cambiar su contraseña.
RF 2	Gestión del índice	La aplicación permitirá gestionar el índice creado por el usuario.
RF 2.1	Listado de empresas del Ibex35	Se proporcionará un listado con las empresas actuales del Ibex35.
RF 2.2	Crear índice	Se podrán crear un índice a partir de las empresas que haya seleccionado.
RF 2.3	Borrar empresa	Se podrá borrar todas las empresas que se desee del índice creado.
RF 2.4	Borrar índice	Se podrá borrar el índice creado.
RF 3	Gestión del portafolio	La aplicación permitirá gestionar el portafolio.
RF 3.1	Crear operación	Se podrán crear operaciones virtuales en el portafolio a través del nuevo índice creado anteriormente y añadiendo el tipo de operación y el número de acciones.
RF 3.2	Borrar Operación	Se podrá borrar una operación del portafolio.
RF 3.3	Consultar Balance	Se podrá consultar el balance del usuario.
RF 4	Gestionar información de una empresa	La aplicación permitirá al usuario visualizar los detalles de una empresa elegida. La información general para mostrar será la siguiente: • Ticker • Precio de cotización • Índice donde cotiza • Rentabilidad • Graficas de precios
RF 4.1	Listar noticias	La aplicación permitirá al usuario ver y seleccionar las noticias referidas a una empresa.
RF 4.2	Listar operaciones del portafolio	La aplicación permitirá al usuario en caso de que existan visualizar las operaciones del portafolio relacionadas con la empresa.
RF 5	Gestionar descarga de noticias	La aplicación en segundo plano se descargará las noticias de cada empresa diariamente.

4.2.2. Requisitos no funcionales

Cuadro 5: Tabla Requisitos no funcionales

Código	Nombre	Descripción
RNF1	Tiempo de respuesta	La aplicación web deberá responder en menos de 19 segundos.
RNF2	Interfaz del sistema	La aplicación presentará una interfaz de usuario sencilla para que sea de fácil manejo para los usuarios.

RNF3	Registro	La aplicación dispondrá de un archivo log que contenga los errores del servidor.
RNF4	Seguridad	La aplicación dispondrá de seguridad en el acceso, no autorizando accesos no deseados
RNF5	Funcionamiento en navegadores web	La aplicación debe funcionar en todos los navegadores web más comunes.
RNF6	Arquitectura	La aplicación debe ser fácilmente escalable pudiendo incorporar nuevas funcionalidades a futuro.

4.3. Especificación de Casos de Uso

4.3.1. Diagrama de contexto

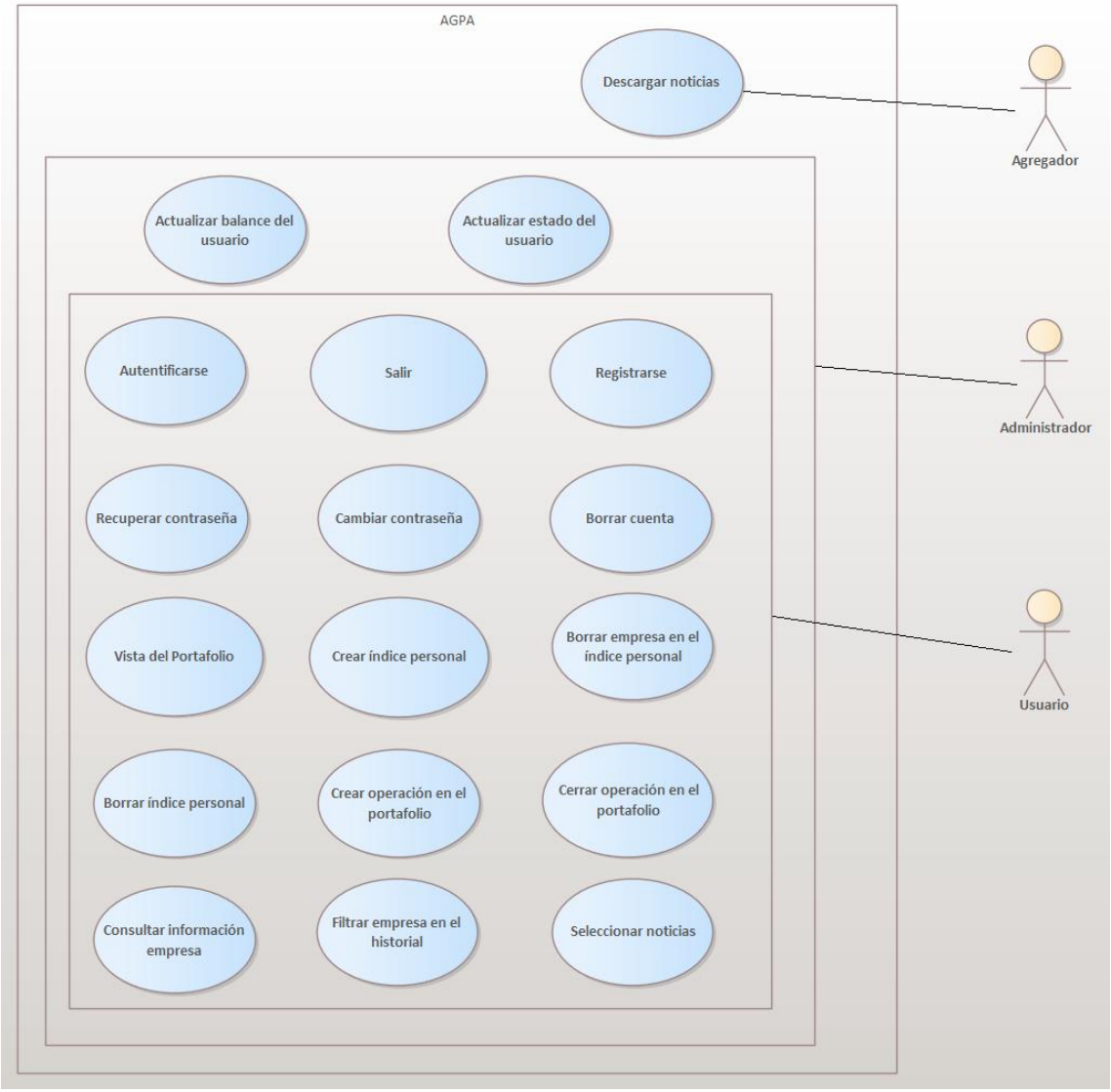

Figura 23: Diagrama de contexto

4.3.2. Casos de Uso

4.3.2.1. Caso de uso 1: Autentificarse

Caso de uso: Autentificarse	
Identificador: CU-AUT	
Descripción	Permite al usuario autentificarse en la aplicación.
Precondición	El usuario ha de haberse añadido con anterioridad a la base de datos.
Actores	Usuario, Administrador
Escenario principal	El usuario accede a la pantalla de inicio de sesión. 1. El usuario introduce usuario y contraseña. 2. El usuario envía el formulario. 3. El usuario es redirigido dentro de la aplicación.
Flujo alternativo	3.1 El usuario introduce mal el usuario o contraseña y es redirigido a una ventana con un mensaje de error.
Flujo alternativo	-

4.3.2.2. Caso de uso 2: Salir

Caso de uso: Salir	
Identificador: CU-SAL	
Descripción	Permite al usuario desconectarse de la aplicación.
Precondición	El usuario debe estar autentificado.
Actores	Usuario, Administrador
Escenario principal	1. En cualquier pantalla el usuario presiona el enlace de “Logout”. 2. El usuario es redirigido a la pantalla de inicio.
Flujo alternativo	1.1 El usuario cierra el navegador y la sesión expira pasado un pequeño periodo de tiempo.
Flujo alternativo	1.2 El usuario está un tiempo sin realizar ninguna acción y el sistema no le permite realizar más acciones.

4.3.2.3. Caso de uso 3: Registrarse

Caso de uso: Registrarse	
Identificador: CU-REG	
Descripción	Permite al usuario registrarse en la aplicación.
Precondición	El usuario no debe estar registrado en la base de datos.
Actores	Usuario, Administrador
Escenario principal	1. El usuario accede a la pantalla de inicio de sesión. 2. El usuario pulsa el botón de “Register”. 3. Se carga una nueva vista donde el usuario debe ingresar su email y su nueva contraseña. 4. El usuario envía el formulario. 5. Se carga una pantalla con un mensaje de éxito. 6. El usuario revisa su correo electrónico y busca el correo asociando a la aplicación web. 7. El usuario pulsa el botón del nuevo correo. 8. Se carga una pantalla con un mensaje de confirmación.
Flujo alternativo	-
Flujo alternativo	-

4.3.2.4. Caso de uso 4: Recuperar contraseña

Caso de uso: Recuperar contraseña	
Identificador: CU-REC	
Descripción	Permite al usuario recuperar su contraseña.
Precondición	El usuario no debe estar autentificado.
Actores	Usuario, Administrador
Escenario principal	1. El usuario accede a la pantalla de inicio de sesión. 2. El usuario hace clic en el enlace “recovery password”. 3. Se carga una nueva vista donde el usuario debe ingresar su email con el que se ha registrado en la aplicación. 4. El usuario envía el formulario. 5. Se carga una nueva vista con un mensaje de confirmación. 6. El usuario revisa su correo electrónico y busca el correo asociando a la aplicación web. 7. El usuario pulsa el botón del nuevo correo. 8. Se carga una nueva vista con un mensaje de confirmación.
Flujo alternativo	4.1 El usuario escribe un email que no existe en la base de datos y es redirigido a una pantalla con un mensaje de error.
Flujo alternativo	-

4.3.2.5. Caso de uso 5: Cambiar contraseña

Caso de uso: Cambiar contraseña	
Identificador: CU-CCO	
Descripción	Permite al usuario cambiar su contraseña.
Precondición	El usuario debe estar autentificado.
Actores	Usuario, Administrador
Escenario principal	1. En cualquier pantalla el usuario pulsa el botón “Resources”. 2. Selecciona “Profile user” y seguido “Change password”. 3. Introduce su nueva contraseña dos veces. 4. Envía el formulario
Flujo alternativo	-
Flujo alternativo	-

4.3.2.6. Caso de uso 6: Borrar cuenta

Caso de uso: Borrar cuenta	
Identificador: CU-BRC	
Descripción	Permite al usuario borrar su cuenta
Precondición	El usuario debe estar autentificado.
Actores	Usuario, Administrador
Escenario principal	1. En cualquier pantalla el usuario pulsa el botón “Resources”. 2. Selecciona “Profile User” y seguido “Delete Account”. 3. Introduce su email y su contraseña. 4. Envía el formulario. 5. Se carga una nueva vista con un mensaje de confirmación. 6. El usuario revisa su correo electrónico y busca el correo asociando a la aplicación web. 7. El usuario pulsa el botón de “Delete Account”. 8. Se carga una nueva vista con un mensaje de confirmación.
Flujo alternativo	4.1 El usuario escribe un email que no existe en la base de datos o escribe mal su contraseña y es redirigido a una ventana con un mensaje de error.
Flujo alternativo	-

4.3.2.7. Caso de uso 7: Vista del portafolio

Caso de uso: Vista del Portafolio	
Identificador: CU-VPF	
Descripción	Permite al usuario poder ver la vista del portafolio sin necesidad de crear ninguna operación.
Precondición	El usuario debe estar autentificado.
Actores	Usuario, Administrador
Escenario principal	En cualquier vista, el usuario selecciona el botón “Resources” y selecciona la opción “Portfolio” y se carga la vista del portafolio.
Flujo alternativo	-
Flujo alternativo	-

4.3.2.8. Caso de uso 8: Crear índice personal

Caso de uso: Crear índice personal	
Identificador: CU-CIN	
Descripción	Permite al usuario crear su propio índice de empresas.
Precondición	El usuario debe estar autentificado.
Actores	Usuario, Administrador
Escenario principal	1. En cualquier pantalla el usuario accede al botón “Resources”. 2. En el nuevo menú, elige “Stock Index” y seguido “Ibex35” 3. Se carga una nueva pantalla con una lista de empresas. 4. El usuario elige las empresas que quiera seleccionar activando sus checkboxes. 5. El usuario pulsa el botón “Add”. 6. El usuario será redirigido a la pantalla principal donde se cargarán las nuevas empresas en su índice donde podrá ver si la empresa o empresas elegidas tienen noticias nuevas sin leer por el usuario, el nombre de cada empresa, el ticker, el ultimo precio actual, la rentabilidad de las últimas 24 horas y un botón de cada empresa para crear operaciones en el portafolio.
Flujo alternativo	-
Flujo alternativo	-

4.3.2.9. Caso de uso 9: Borrar empresa en el índice personal

Caso de uso: Borrar empresa en el índice personal	
Identificador: CU-BEI	
Descripción	Permite al usuario borrar una empresa del índice personal creado por el usuario.
Precondición	El usuario debe estar autentificado. El usuario tuvo que crear su propio índice anteriormente.
Actores	Usuario, Administrador
Escenario principal	1. En la pantalla principal, el usuario elegirá las empresas que quiera borrar pulsando en sus checkboxes. 2. El usuario pulsará el botón “Delete companies”. 3. En la misma pantalla las empresas seleccionadas desaparecerán y se cargarán las restantes.
Flujo alternativo	-
Flujo alternativo	-

4.3.2.10. Caso de uso 10: Borrar índice personal

Caso de uso: Borrar índice personal	
Identificador: CU-BIN	
Descripción	Permite al usuario borrar el nuevo índice entero de una vez.
Precondición	El usuario debe estar autentificado. El usuario debió crear el nuevo índice anteriormente.
Actores	Usuario, Administrador
Escenario principal	1. En la pantalla principal, el usuario pulsará el botón “delete index” 2. En la misma pantalla el índice de empresas se borrará entero.
Flujo alternativo	-
Flujo alternativo	-

4.3.2.11. Caso de uso 11: Crear operación en el portafolios

Caso de uso: Crear operación en el portafolio	
Identificador: CU-COP	
Descripción	Permite al usuario crear una nueva operación virtual en el portafolios.
Precondición	El usuario debe estar autentificado. El usuario debió crear el índice personal anteriormente.
Actores	Usuario, Administrador
Escenario principal	1. En la pantalla principal el usuario seleccionará una de las empresas seleccionadas pulsando el botón de "Action" de una de las empresas. 2. Se carga una nueva vista donde el usuario tiene que seleccionar si quiere comprar o vender y añadir el número de acciones que quiere en la operación. 3. El usuario pulsa el botón "Create" si este está habilitado. 4. La página se recarga añadiendo su nueva operación. 5. El usuario verá su nueva operación con la fecha de apertura, si fue compra o venta, el ticker de la empresa, el precio de apertura, la cantidad de acciones que ha elegido y la rentabilidad actual.
Flujo alternativo	3.1 El usuario intenta seleccionar un número mayor de acciones al que le son permitidos por su balance y el botón permanece inhabilitado.
Flujo alternativo	-

4.3.2.12. Caso de uso 12: Cerrar operación en el portafolio

Caso de uso: Cerrar operación en el portafolio	
Identificador: CU-CCP	
Descripción	Permite al usuario cerrar una operación en el portafolio.
Precondición	El usuario debe estar autentificado. El usuario debió crear una operación en el portafolio.
Actores	Usuario, Administrador
Escenario principal	1. En cualquier pantalla el usuario hará clic en el botón "Resources". 2. Elegirá la opción "portfolio". 3. Seleccionará la operación abierta que desee pulsando el botón "Close". 4. Se recargará la pantalla sin esa operación.
Flujo alternativo	-
Flujo alternativo	-

4.3.2.13. Caso de uso 13: Filtrar empresa en el historial

Caso de uso: Filtrar empresa en el historial	
Identificador: CU-VHO	
Descripción	Permite al usuario filtrar por sus ticker para buscar una empresa en el historial de operaciones.
Precondición	El usuario debe estar autentificado. El usuario debió cerrar con anterioridad alguna operación del portafolio.
Actores	Usuario, Administrador
Escenario principal	1. En cualquier pantalla el usuario pulsara el botón “Resources”. 2. En el menú desplegable, el usuario seleccionara “Records”. 3. Se carga una pantalla con cero o más operaciones donde se muestra la fecha de inicio de la operación, si fue compra o venta, el ticker, el precio de apertura, el número de acciones seleccionadas, la fecha de cierre, el precio de cierre y la rentabilidad generada. 4. El usuario escribe un ticker y pulsa el botón “Filter”. 5. Se le mostrarán las operaciones cerradas referentes a ese ticker.
Flujo alternativo	4.1 El usuario no necesita filtrar por ninguna empresa pudiendo ver todas las operaciones cerradas anteriormente.
Flujo alternativo	-

4.3.2.14. Caso de uso 14: Consultar información empresa

Caso de uso: Consultar información empresa	
Identificador: CU-CIE	
Descripción	Permite al usuario consultar más información sobre las empresas creadas en el índice creado por él.
Precondición	El usuario debe estar autentificado. El usuario tuvo que añadir una empresa en el índice personal.
Actores	Usuario, Administrador
Escenario principal	1. En la pantalla principal, el usuario verá un listado de empresas 2. El usuario pinchará en el nombre de una empresa. 3. Se cargará una pantalla con la siguiente información de la empresa: • El nombre de la empresa, su ticker, el índice donde cotiza, su precio de cotización y su rentabilidad diaria. • Un historial de precios en forma de 3 gráficos, cada uno de una temporalidad distinta (1 mes, 3 meses y 6 meses). • Las operaciones, en caso de que haya, de la empresa en el portafolios, con su ticker y su rentabilidad. • Las noticias relacionadas con la empresa.
Flujo alternativo	3.1 La pantalla principal puede no tener empresas, ya que el usuario previamente no las añadió.
Flujo alternativo	-

4.3.2.15. Caso de uso 15: Seleccionar noticias

Caso de uso: Seleccionar noticias	
Identificador: CU-SNN	
Descripción	Permite al usuario seleccionar las noticias que quiera ver
Precondición	El usuario debe estar autentificado.
Actores	Usuario, Administrador
Escenario principal	1. En la pantalla principal, el usuario verá un listado de empresas 2. El usuario pinchará en el nombre de una empresa. 3. En la siguiente pantalla el usuario podrá seleccionar las noticias que quiera ver a través de los títulos, que permanecerán en rojo mientras no se pinchen en ellas. 4. Al seleccionar una noticia, el usuario podrá ver la fecha en la que fue publicada, un resumen y un enlace a la noticia original. 5. Cuando el usuario pinche en el enlace, el título de la noticia seleccionada cambiará a verde.
Flujo alternativo	1.1 El usuario puede encontrase con cero noticias que ver.
Flujo alternativo	-

4.3.2.16. Caso de uso 16: Actualizar estado del usuario

Caso de uso: Actualizar estado del usuario	
Identificador: CU-AOD	
Descripción	Permite al administrador activar o desactivar a los usuarios
Precondición	El administrador debe estar autentificado.
Actores	Administrador
Escenario principal	6. En cualquier pantalla el administrador hará clic en el botón “Admin”. 7. Se cargará una pantalla donde el administrador verá una lista con los usuarios registrados en la aplicación donde aparecerán sus emails, sus estados y sus balances actuales. 8. El administrador pulsará el botón “Activate/Deactivate”. 9. Se recargará la pantalla cambiando el estado del campo a “enabled” si previamente era “disabled” o a “disabled” si previamente era “enabled”.
Flujo alternativo	-
Flujo alternativo	-

4.3.2.17. Caso de uso 17: Actualizar balance del usuario

Caso de uso: Actualizar balance del usuario	
Identificador: CU-ABL	
Descripción	Permite al administrador actualizar el balance de un usuario.
Precondición	El administrador debe estar autentificado.
Actores	Administrador
Escenario principal	1. En cualquier pantalla el administrador hará clic en el botón “admin”. 2. Se cargará una pantalla donde el administrador verá una lista con los usuarios registrados en la aplicación donde aparecerán sus emails, su estado y sus balances actuales. 3. El administrador introducirá una cantidad en el campo “Balance” y pulsará el botón “Change Balance”. 4. Se recargará la pantalla con el balance actual del usuario.
Flujo alternativo	-
Flujo alternativo	-

4.3.2.18. Caso de uso 18: Descargar noticias

Caso de uso: Descargar noticias	
Identificador: CU-DCN	
Descripción	Permite al administrador actualizar el balance de un usuario.
Precondición	Debe existir un archivo de configuración donde se almacenan los enlaces RSS desde donde se descargan las noticias.
Actores	Agregador
Escenario principal	1. Cada día el Agregador lee los enlaces RSS contenidos en el archivo “newspaper.json” y se descarga las noticias. 2. Las noticias descargadas se almacenan en la base de datos.
Flujo alternativo	2.1 las noticias repetidas no se guardarán en la base de datos.
Flujo alternativo	-

4.4. Análisis de interfaces de usuario

Tenemos en cuenta dos versiones. Una primera versión que era la idea principal de las vistas que queríamos presentar en la aplicación y una segunda versión donde algunas vistas se han modificado o incluso se han juntado para que el uso de la aplicación fuera mucho más fácil.

4.4.1. Primera versión

En esta primera versión ya se tenía claro que tipo de aplicación se quería hacer, aunque no se sabía bien todas las funcionalidades que tenían.

Todas las vistas una vez que el usuario se ha autentificado en la aplicación tienen un ícono en la esquina superior izquierda con la que pueden navegar por la aplicación.

4.4.1.1. Login

Código: IU-1

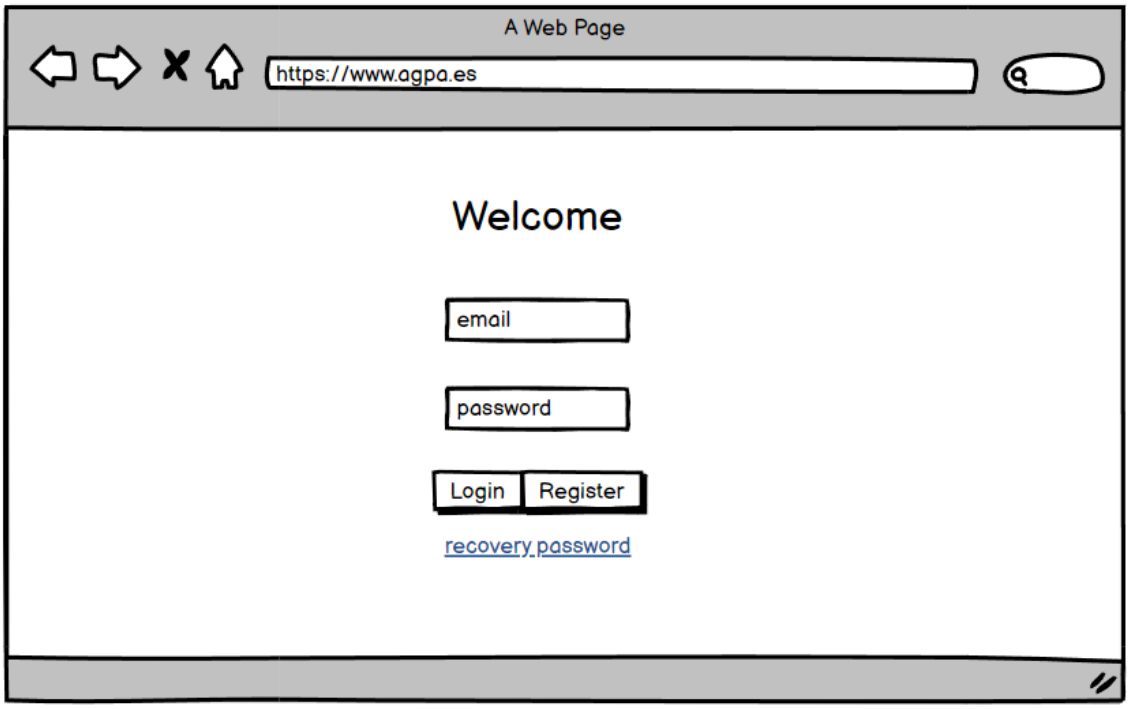

Figura 24: Prototipo del Login de la primera versión

4.4.1.2. Registro

Código: IU-2

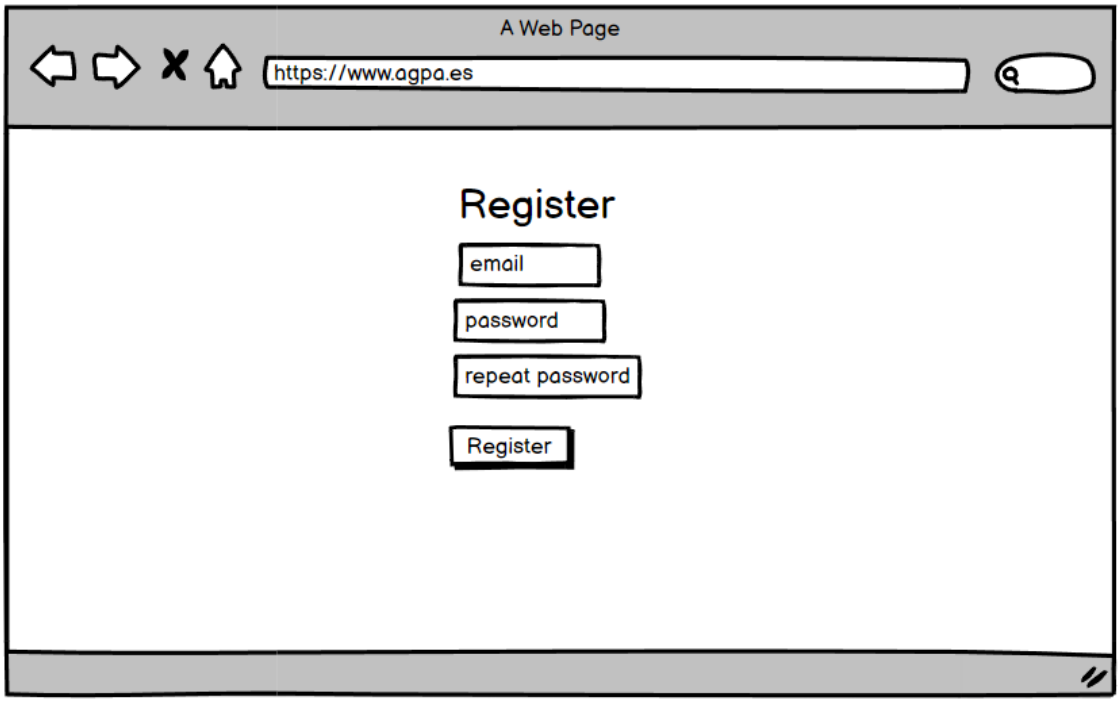

Figura 25: Prototipo Registro de la primera versión

4.4.1.3. Recuperar contraseña

Código: IU-3

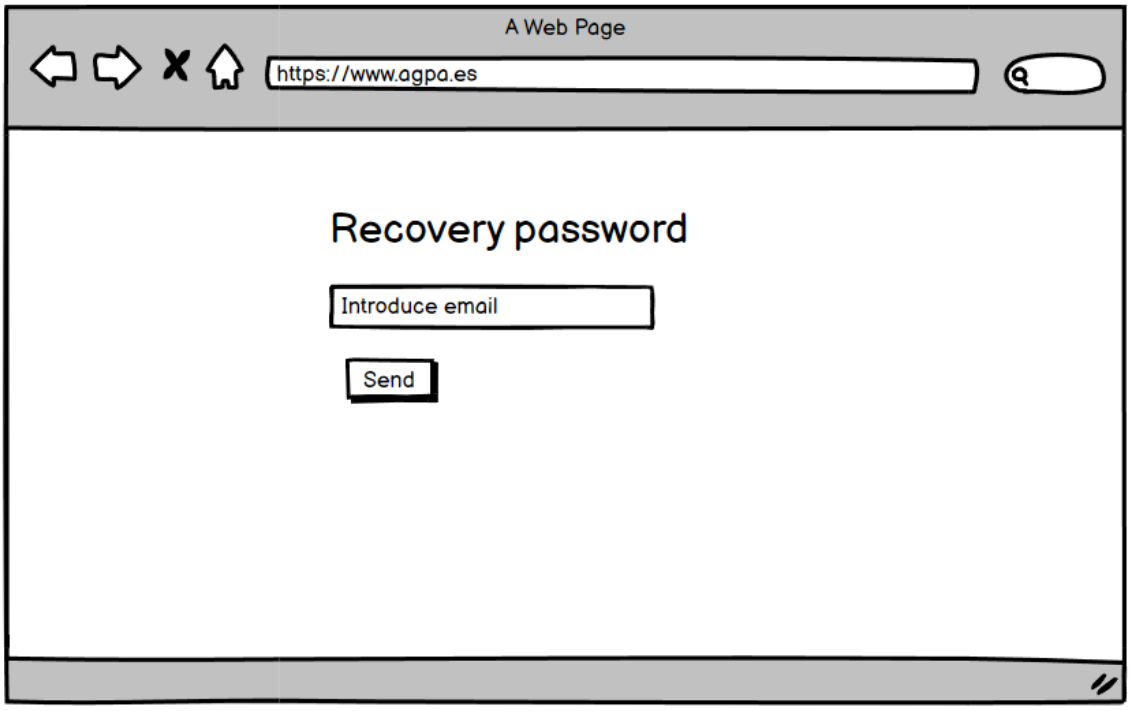

Figura 26: Prototipo Recuperar contraseña de la primera versión

4.4.1.4. Página principal

Código: IU-4

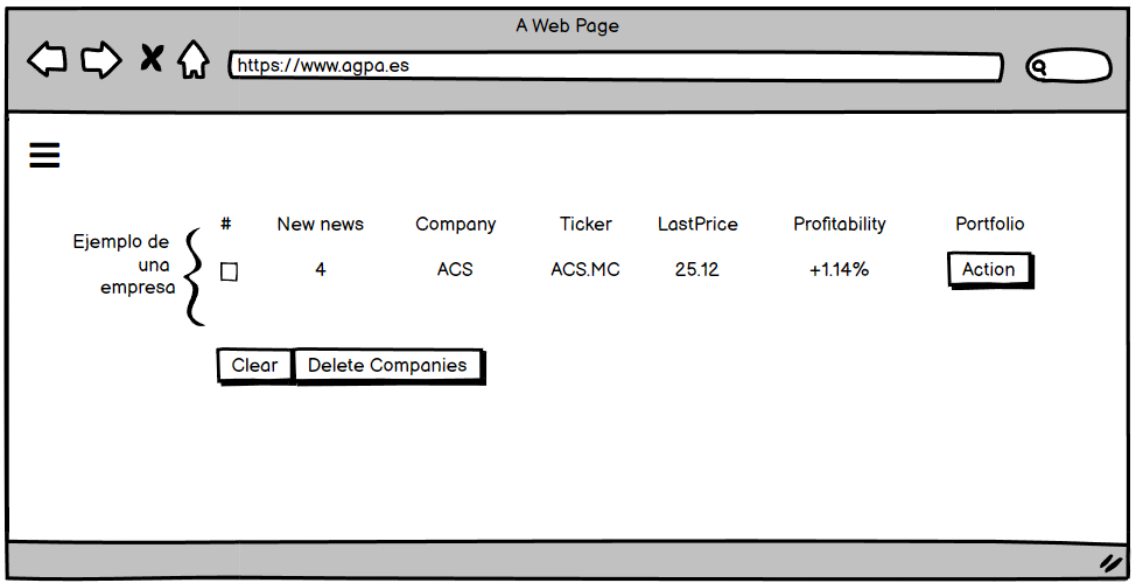

Figura 27: Prototipo de la Página principal de la primera versión

4.4.1.5. Menú de navegación

Código: IU-5

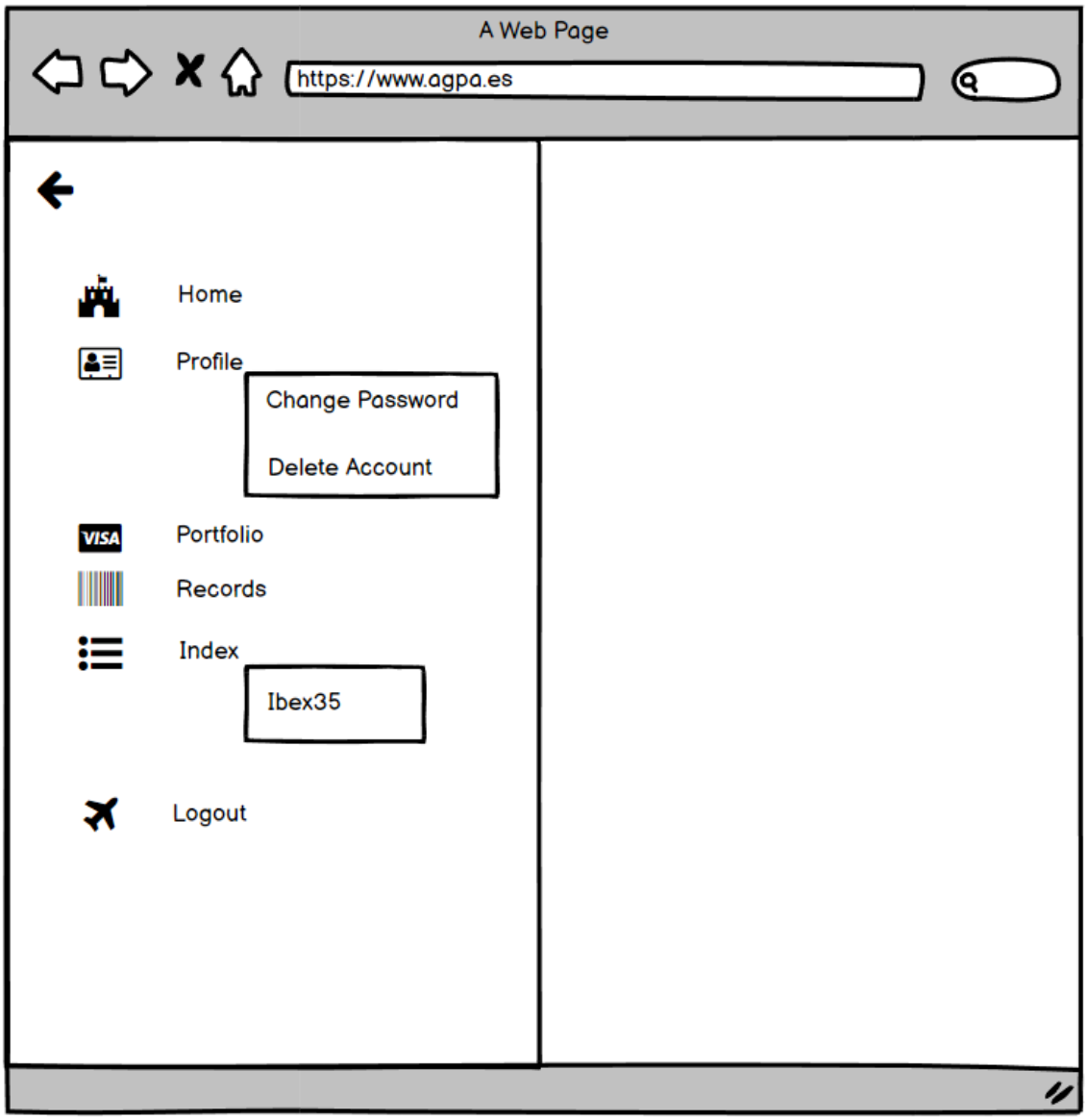

Figura 28: Prototipo Menú de navegación de la primera versión

4.4.1.6. Índice empresas

Código: IU-6

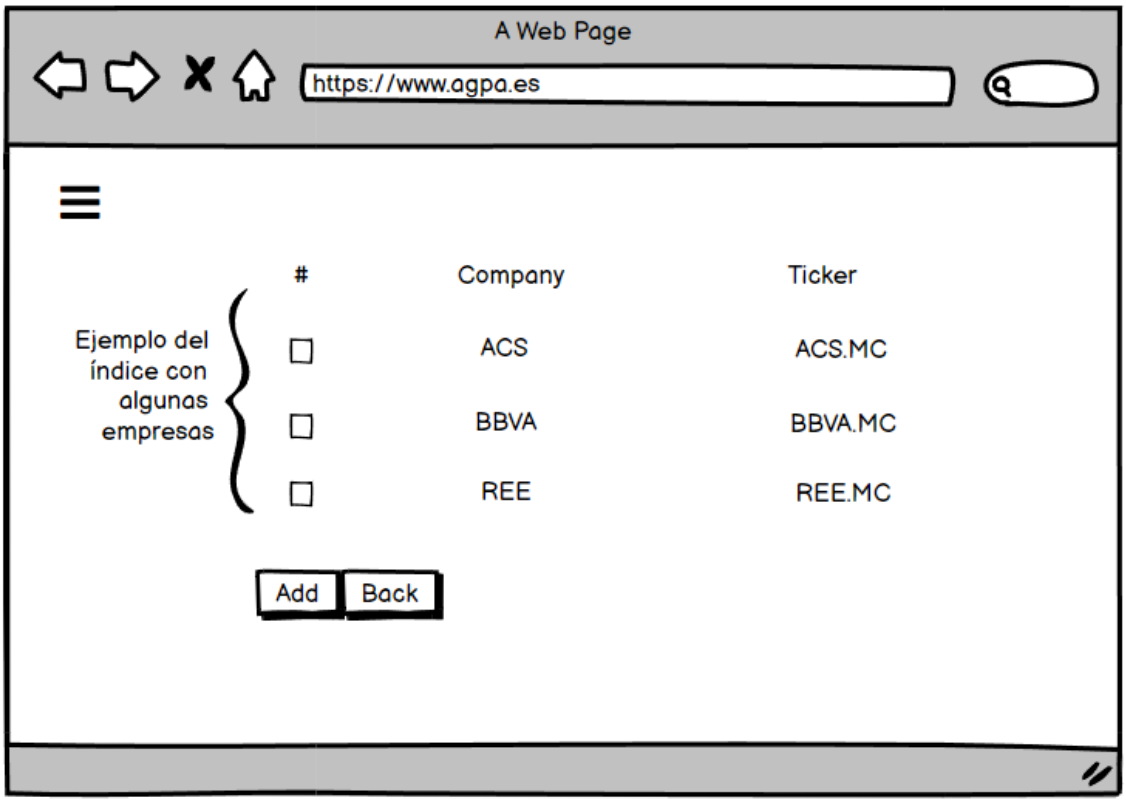

Figura 29: Prototipo de índice de empresas de la primera versión

4.4.1.7. Cambiar contraseña

Código: IU-7

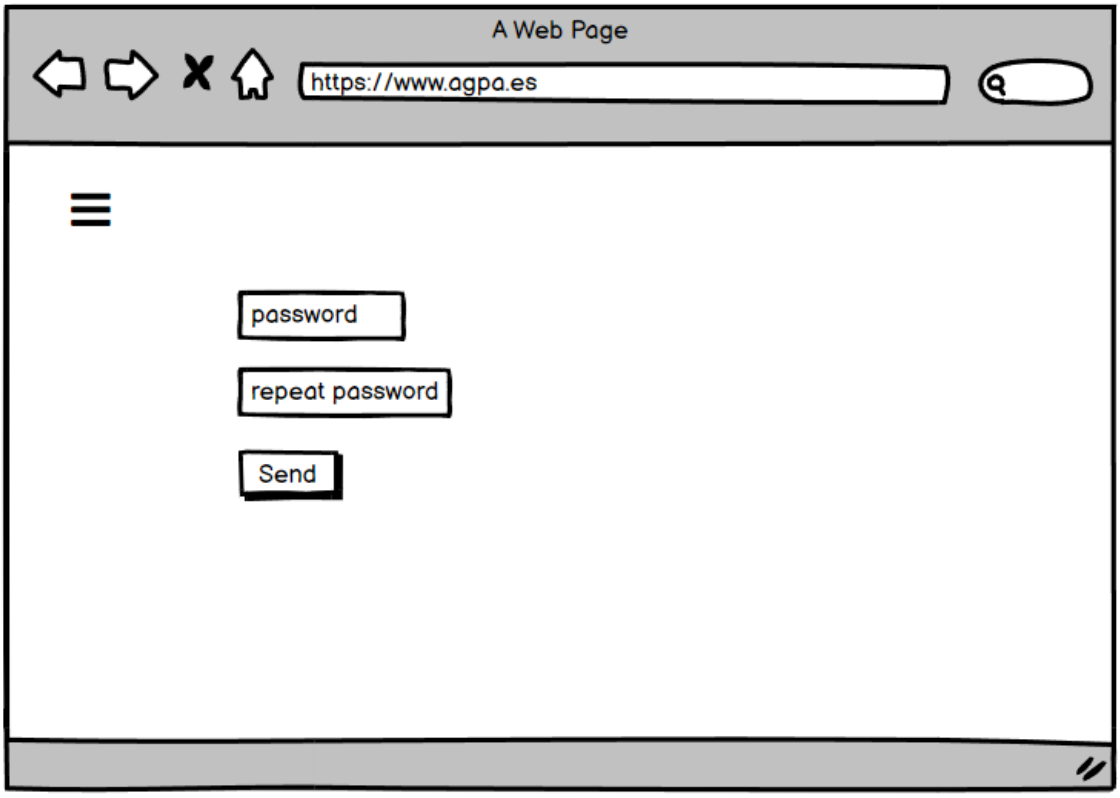

Figura 30: Prototipo Cambiar contraseña de la primera versión

4.4.1.8. Borrar cuenta

Código: IU-8

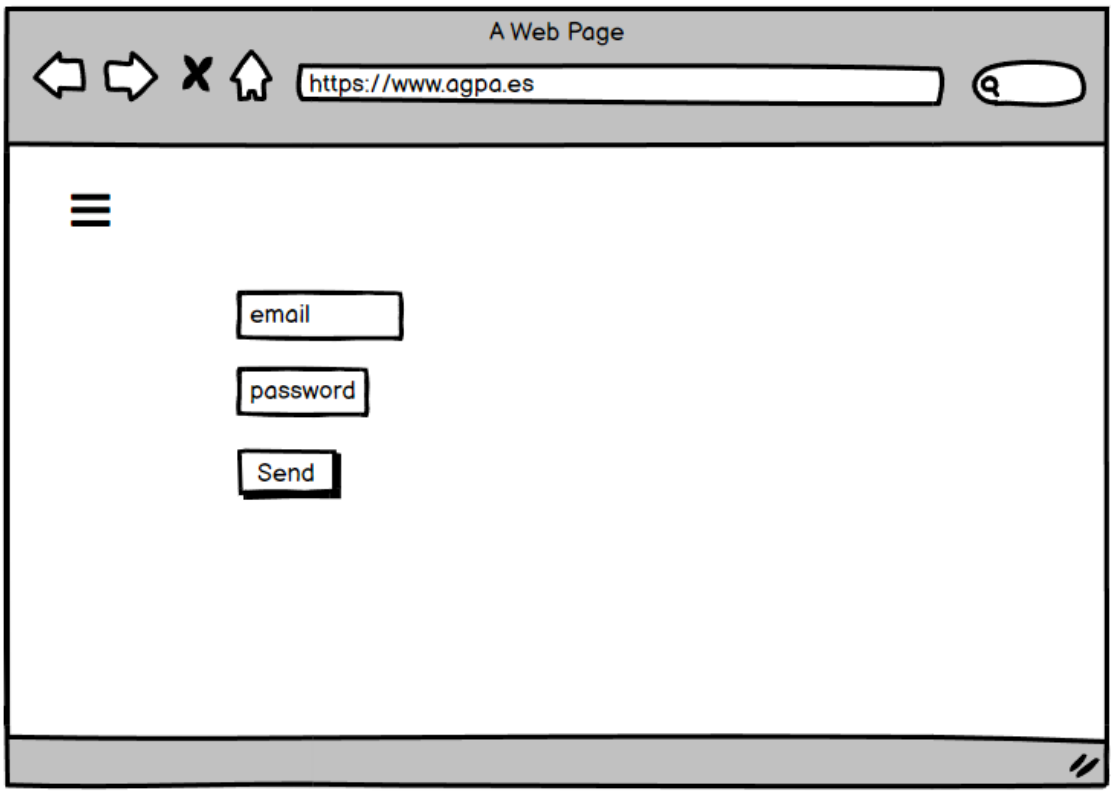

Figura 31: Prototipo Borrar cuenta de la primera versión

4.4.1.9. Administrador

Código: IU-9

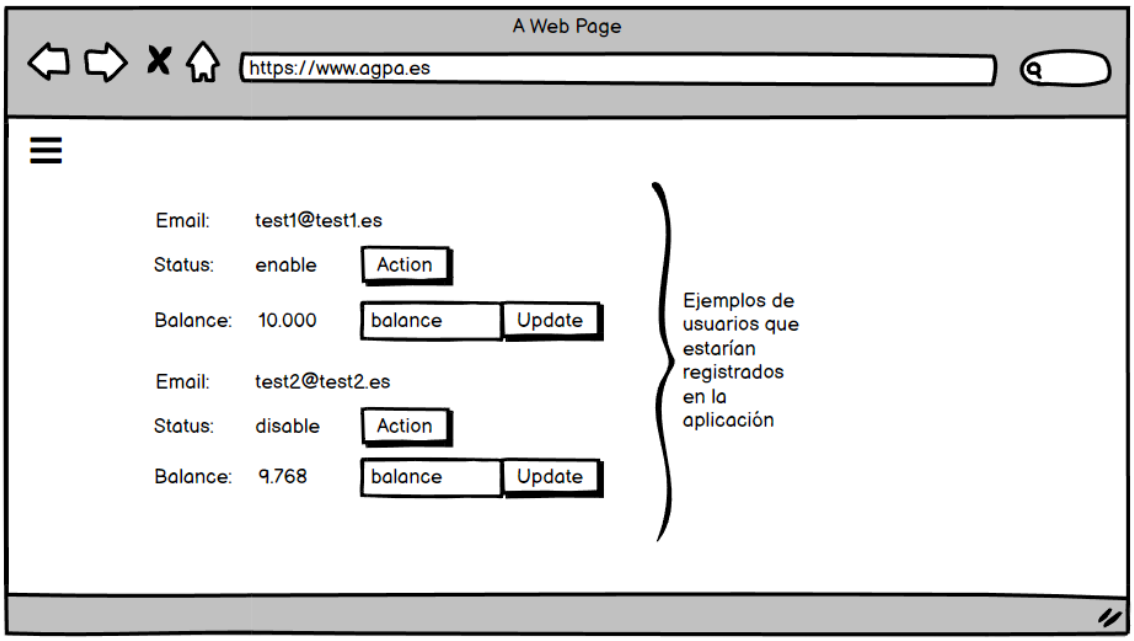

Figura 32: Prototipo Administrador de la primera versión

4.4.1.10. Información empresa

Código: IU-10

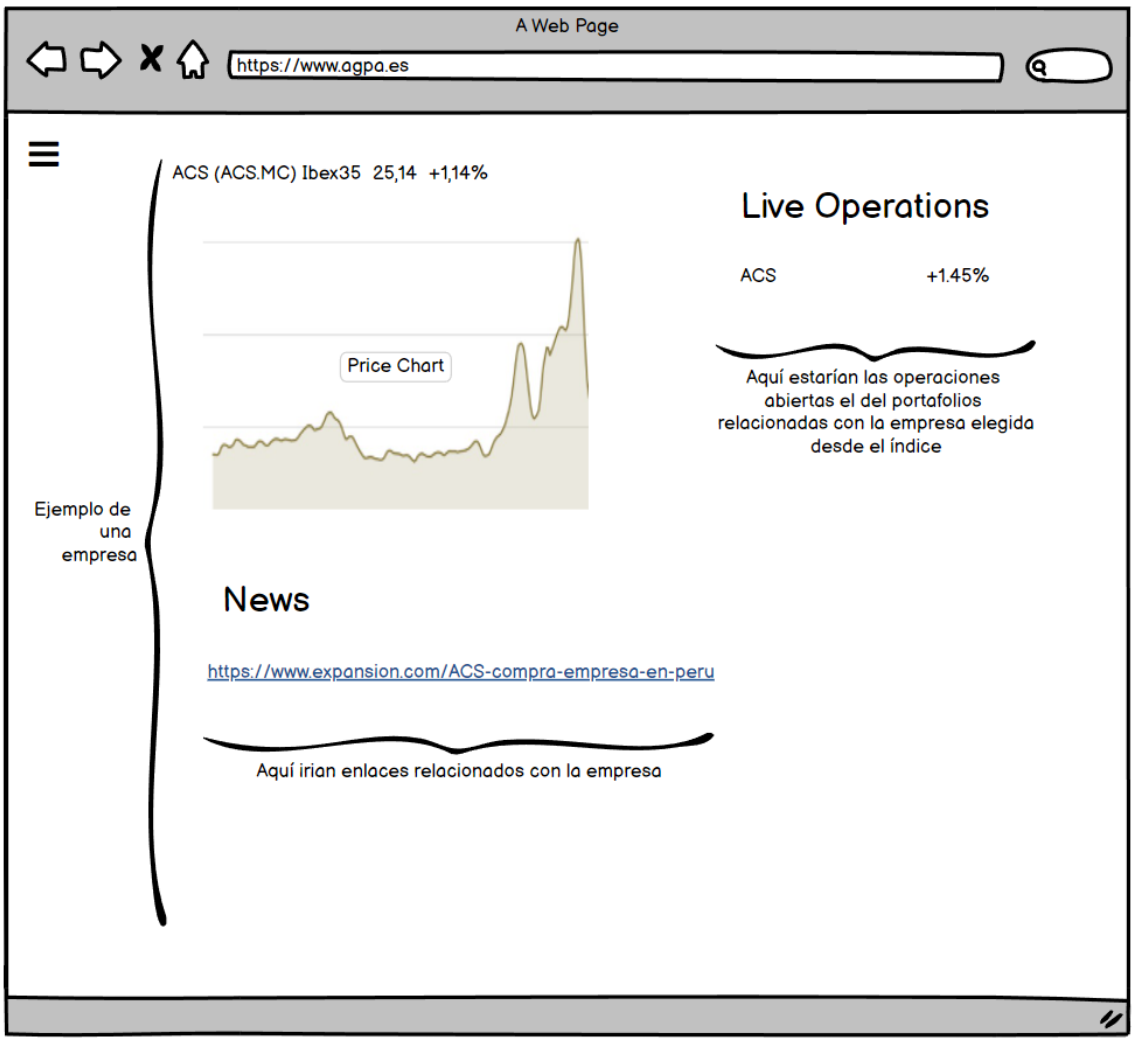

Figura 33: Prototipo Información empresa de la primera versión

4.4.1.11. Buy

Código: IU-11

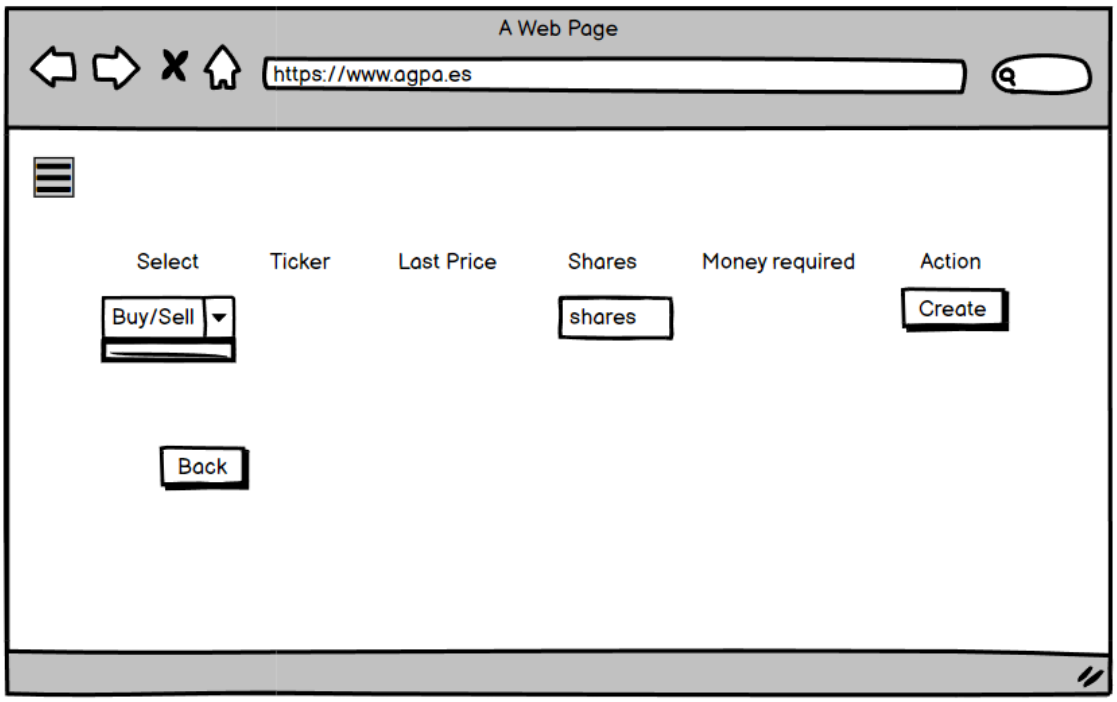

Figura 34: Prototipo Buy de la primera versión

Esta vista se ha integrado en la segunda versión junto con la vista del Portafolio para que fuera más fácil el poder decidir si se querían abrir operaciones o no.

4.4.1.12. Portafolio

Código: IU-12

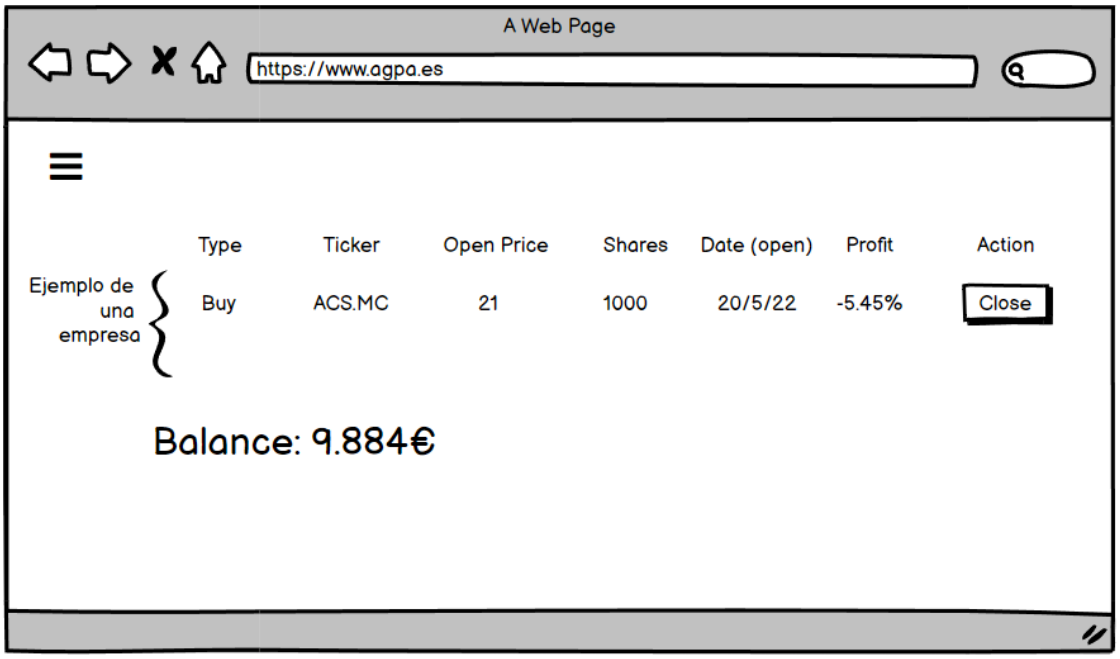

Figura 35: Prototipo Portafolio de la primera versión

4.4.2. Segunda versión

En esta versión añadimos los cambios. Algunas vistas se unifican, se añade un historial de operaciones ya cerradas y se añade una nueva forma de ver las noticias.

4.4.2.1. Historial de operaciones

Código: IU-13

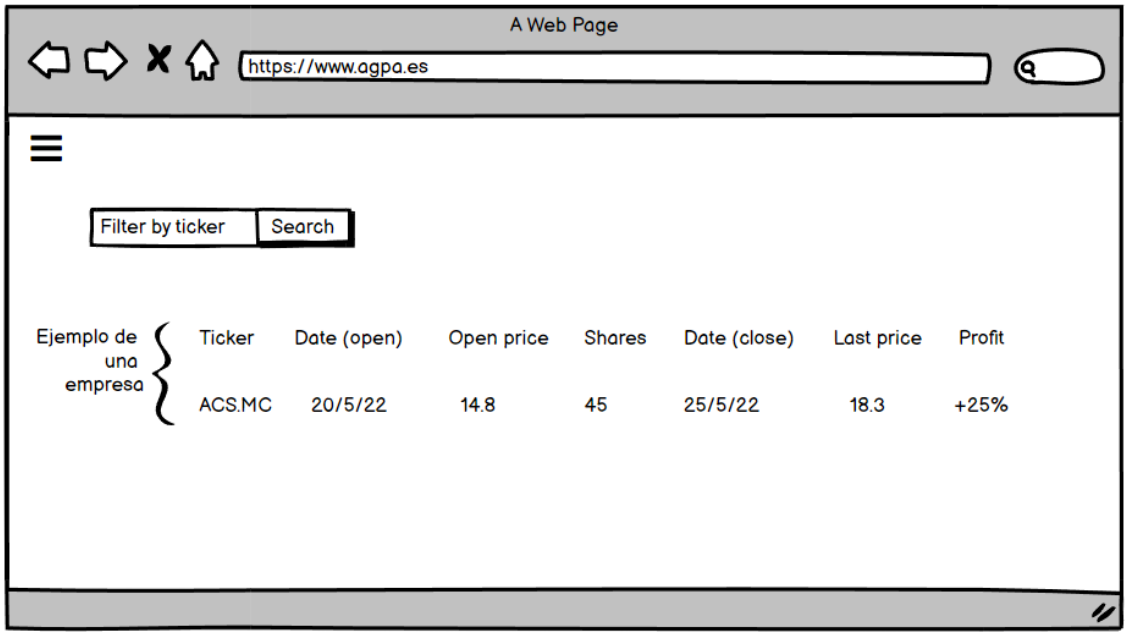

Figura 36: Prototipo Historial de operaciones de la segunda versión

Esta vista se añade nueva, así se tienen guardadas todas las operaciones que ha realizado un usuario.

4.4.2.2. Información empresa

Código: IU-14

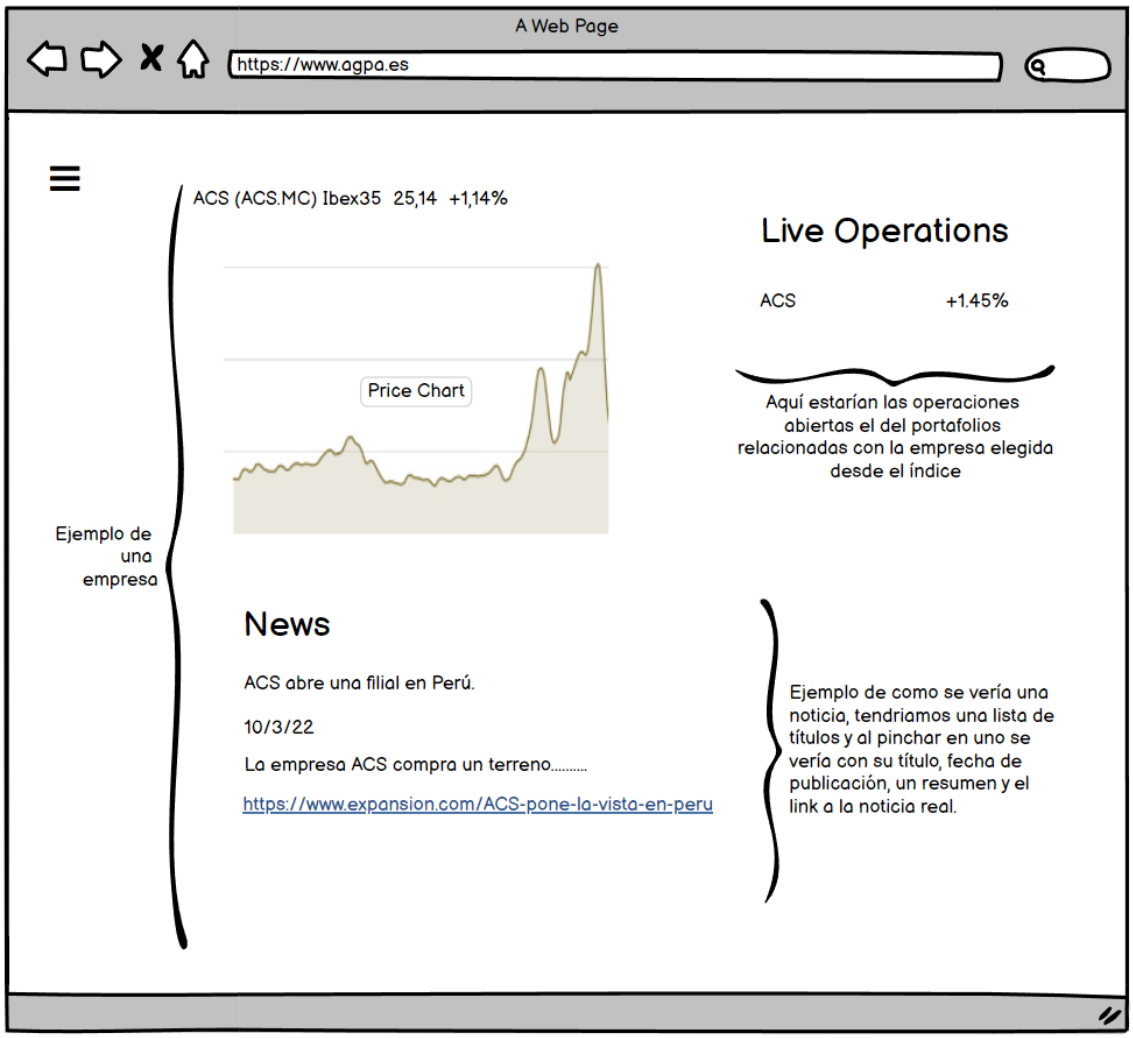

Figura 37: Prototipo Información empresa de la segunda versión

Se ha añadido une mejor forma de poder visualizar las noticias relacionadas con una empresa, de forma que tengamos una lista de noticias y cada vez que el usuario pinche en un enlace se podrá ver más información de ella.

4.4.2.3. Portafolio

Código: IU-15

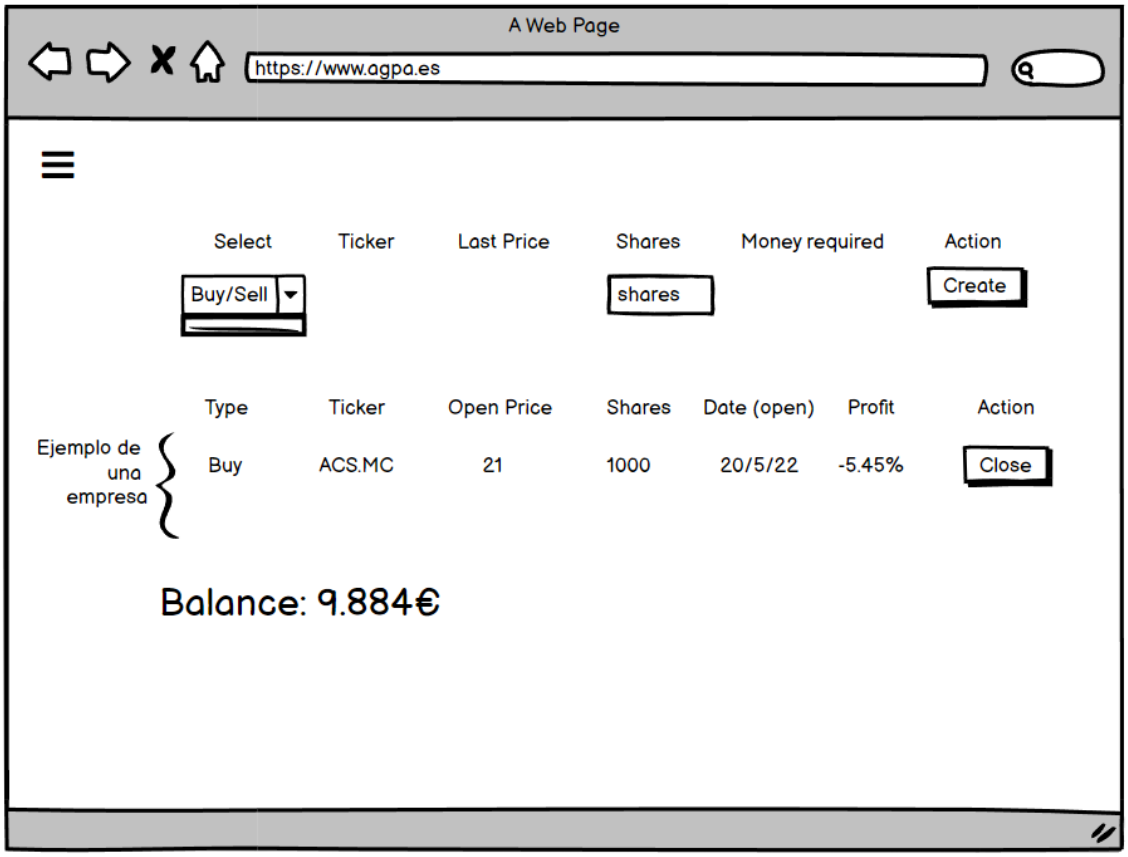

Figura 38: Prototipo Portafolio de la segunda versión

Se ha unificado la vista "buy" con la vista del portafolio de forma que sea más fácil para el usuario agregar las operaciones y estar más seguro de si abrirla o no.

4.5. Trazabilidad Interfaces de usuario – Casos de uso

La matriz se ha hecho con las vistas finales, no se ha tenido en cuenta las vistas de la primera versión que fueron repetidas en la segunda versión con más éxito. Las filas son los casos de uso y las columnas las interfaces.

Cuadro 6: Tabla trazabilidad interfaces - Casos de uso

	IU-1	IU-2	IU-3	IU-4	IU-5	IU-6	IU-7	IU-8	IU-9	IU-13	IU-14	IU-15
CU-AUT	x			x								
CU-SAL	x											
CU-REG	x	x										
CU-REC	x		x									
CU-CCO				x			x					
CU-CIN				x		x						
CU-BEI				x								
CU-BIN				x								

CU-COP				x							x	
CU-VHO									x	x		
CU-CIE				x							x	
CU-CCP												x
CU-AOD									x			
CU-ABL									x			
CU-DCN												
CU-VPF												x

4.6. Especificación del plan de pruebas

4.6.1. Caso de uso 1: Autentificarse

Caso de prueba 1	Resultado esperado
Autentificarse con datos válidos	El sistema inicia la sesión del usuario y lo redirige a la página principal de la aplicación.

Caso de prueba 2	Resultado esperado
Autentificarse con datos inválidos	El sistema no permite la autentificación del usuario respondiéndole con un mensaje con la explicación correspondiente.

Caso de prueba 3	Resultado esperado
Autentificarse con un usuario no activado	El sistema no permite la autentificación del usuario respondiéndole con un mensaje con la explicación correspondiente.

4.6.2. Caso de uso 2: Salir

Caso de prueba 4	Resultado esperado
Salir de la aplicación manualmente	El sistema redirige al usuario a la pantalla de inicio de sesión una vez pulsado el enlace para salir de la aplicación.

Caso de prueba 5	Resultado esperado
Salir de la aplicación al cerrar el navegador	El sistema redirige al usuario a la pantalla de inicio de sesión cuando cierra el navegador.

Caso de prueba 6	Resultado esperado

Salir de la aplicación cuando se consume el tiempo establecido	El sistema redirige al usuario a la pantalla de inicio de sesión cuando lleva un tiempo sin usar la aplicación.

4.6.3. Caso de uso 3: Registrarse

Caso de prueba 7	Resultado esperado
Registrarse con datos válidos	El sistema agrega al usuario en la base de datos con el estado a true y lo redirige y le muestra un mensaje con la explicación correspondiente y le manda un email.

Caso de prueba 8	Resultado esperado
Registrarse con datos inválidos	El sistema lo redirige a una página con un mensaje con la explicación correspondiente.

4.6.4. Caso de uso 4: Recuperar contraseña

Caso de prueba 9	Resultado esperado
Recuperar contraseña con datos validos	El sistema redirige al usuario a una página con un mensaje con la explicación correspondiente y le envía un email.

Caso de prueba 10	Resultado esperado
Recuperar contraseña con datos inválidos	El sistema no permite el registro y redirige al usuario a una página con un mensaje con la explicación correspondiente.

4.6.5. Caso de uso 5: Cambiar contraseña

Caso de prueba 11	Resultado esperado
Cambiar contraseña	El sistema permite al usuario cambiar la contraseña de su cuenta y le redirige a la página principal de la aplicación.

4.6.6. Caso de uso 6: Borrar cuenta

Caso de prueba 12	Resultado esperado
Deshabilitar cuenta con datos validos	El sistema cambia el estado del usuario a false, lo redirige a una página que le muestra el mensaje correspondiente y le manda un email.

Caso de prueba 13	Resultado esperado
Deshabilitar cuenta con datos inválidos	El sistema no permitirá deshabilitar la cuenta y lo redirigirá al usuario a una página con un mensaje con la explicación correspondiente.

4.6.7. Caso de uso 7: Vista de Portafolio

Caso de prueba 14	Resultado esperado
Seleccionar vista para ver el portafolio	El sistema muestra el portafolio con las operaciones abiertas si las hubiera, el balance y el botón “Create” desactivado.

4.6.8. Caso de uso 8: Crear índice personal

Caso de prueba 15	Resultado esperado
Crear un nuevo índice personal	El sistema guarda en la base de datos las empresas seleccionadas y lo redirige a la página principal de la aplicación donde se le mostrarán las empresas seleccionadas.

Caso de prueba 16	Resultado esperado
Crear un nuevo índice personal con empresas repetidas	El sistema guarda en la base de datos las empresas elegidas y cuando el usuario vuelve a elegir las mismas empresas se vuelven a añadir.

4.6.9. Caso de uso 9: Borrar empresa en el índice personal

Caso de prueba 17	Resultado esperado
Seleccionar empresas para borrar del índice personal.	El sistema borra la empresa seleccionada de la base de datos y refresca la página dejando de mostrar la empresa seleccionada.

4.6.10. Caso de uso 10: Borrar índice personal

Caso de prueba 18	Resultado esperado
Borrar el índice personal	El sistema borra el nuevo índice de la base de datos y refrescará la página mostrando el índice vacío.

4.6.11. Caso de uso 11: Crear operación en el portafolio

Caso de prueba 19	Resultado esperado

Crear una nueva operación con una cantidad monetaria inferior al balance del usuario.	El sistema crea la nueva operación en la base de datos y refresca la página mostrando la nueva operación.

Caso de prueba 20	Resultado esperado
Crear una nueva operación con una cantidad monetaria superior al balance del usuario.	El sistema inhabilita el botón “Create”.

4.6.12. Caso de uso 12: Cerrar operación en el portafolio

Caso de prueba 21	Resultado esperado
Cerrar una operación en el portafolio	El sistema borra la operación en la base de datos, refrescara la página no mostrando la operación seleccionada.

4.6.13. Caso de uso 13: Filtrar empresa en el historial

Caso de prueba 22	Resultado esperado
Filtrar empresas por ticker	El sistema muestra las operaciones ya cerradas seleccionadas por su ticker.

4.6.14. Caso de uso 15: Seleccionar noticias

Caso de prueba 23	Resultado esperado
Descontar el número de noticias ya vistas por el usuario	El sistema descuenta en el índice personal las noticias ya leídas por el usuario.

4.6.15. Caso de uso 16: Actualizar estado del usuario

Caso de prueba 24	Resultado esperado
Actualizar estado de un usuario por el administrador	El sistema guarda en la base de datos el nuevo estado y refresca la página mostrando el estado actual del usuario seleccionado.

Caso de prueba 25	Resultado esperado
Actualizar estado de un usuario por un usuario no autorizado	El sistema no permite la entrada al espacio del administrador y responde con un mensaje con la explicación correspondiente.

4.6.16. Caso de uso 17: Actualizar balance del usuario

Caso de prueba 26	Resultado esperado
Actualizar balance de un usuario por el administrador	El sistema guarda en la base de datos el nuevo balance y refresca la página mostrando el balance actual del usuario seleccionado.

Caso de prueba 27	Resultado esperado
Actualizar balance de un usuario por un usuario no autorizado	El sistema no permite la entrada al espacio del administrador y responde con un mensaje con la explicación correspondiente.

5. Diseño del sistema

5.1. Justificación de la plataforma seleccionada

5.1.1. Python

El lenguaje creado por Guido Van Rossum ha sido elegido durante tres años consecutivos como el mejor lenguaje de programación por la conocida revista IEEE Spectrum destinada a tecnología, ingeniería y ciencia.

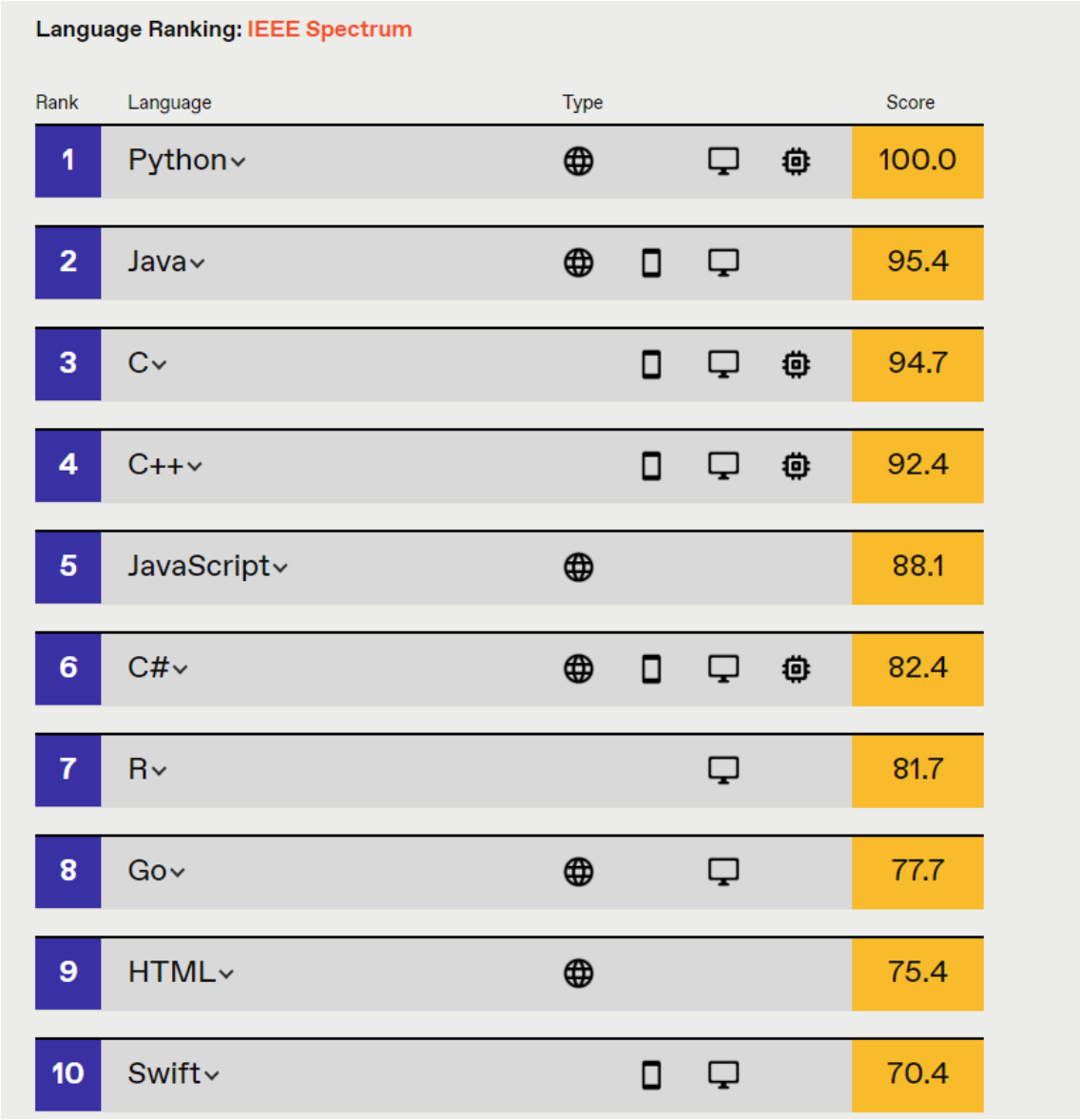

Figura 39: Captura Ranking Language. Fuente: IEEE Spectrum[5]

[5] https://spectrum.ieee.org/top-programming-languages/

A continuación, se explican las razones por las que se ha elegido Python como lenguaje usado en el lado del servidor:

- Es un lenguaje que mantiene la filosofía de "baterías incluidas", es un lenguaje con el que se puede hacer prácticamente de todo, con una biblioteca estándar rica y versátil.
- Es un lenguaje ampliamente respaldado con una comunidad muy activa, cada poco tiempo están sacando versiones actualizadas del lenguaje.
- Es lenguaje open source por lo que no requiere licencias de pago.
- Es multiplataforma, mientras tenga un intérprete para cada plataforma. Linux por ejemplo ya lo incorpora en su núcleo.
- Es polivalente, puede ser usado para diferentes tipos de proyectos como por ejemplo para el desarrollo web, inteligencia artificial, machine learning, análisis de datos, etc.
- Es multiparadigma
- Es un lenguaje fácil de aprender, usar y comprender, con una sintaxis limpia y ordenada.
- Soporta la programación asincrónica. Desde sus últimas actualizaciones, con Python se puede crear aplicaciones asíncronas que realicen operaciones de E/S sin bloqueo.

5.1.2. FastApi

Es un framework web moderno y rápido, desarrollado para construir APIs con las versiones más actualizadas de Python. Ha sido utilizado para construir el backend de esta aplicación web

Figura 40: Logo de Fastapi

A continuación, se explican las razones por las que se ha elegido Fastapi como framework usado en el lado del servidor:

- Es uno de los frameworks web más rápidos de Python que haya actualmente sino el que más, pudiendo competir con NodeJs en velocidad.

- Tiene un sistema de inyección de dependencia que es muy intuitivo y fácil de usar.
- Es muy fácil de depurar y ofrece mucha información detallada.
- Provee una interfaz interactiva y documentada, y un sistema de autentificación basado en OAuth2 que nos ayuda a testear todos los métodos de la API REST de una manera más intuitiva y fácil.
- Posee una comprobación de tipos a nivel de ejecución gracias a Pydantic, muy fácil de implementar que controla todas las entradas y salidas de la API REST.
- Soporta todo tipo de bases de datos, tanto relacionales como no relacionales.
- Soporta las corrutinas pudiendo crear aplicaciones asíncronas.
- Es fácil de usar y aprender pudiendo acceder a documentación de fácil lectura.
- Tiene soporte para diferentes tecnologías como ElasticSearch o Graphql.

5.1.3. Angular

Es uno de los framework web de JavaScript más usados, creado por Google. Permite crear aplicaciones web robustas y permite la creación de SPA (single page application).

Figura 41: Logo de Angular

A continuación, se explican las razones por las que se ha elegido Angular como framework usado en el lado del cliente:

- Posee mucha documentación, lo cual ha favorecido que se haya elegido este framework, ya que ha hecho que la curva de aprendizaje no fuera tan elevada.
- Posee una comunidad muy activa que junto con el punto anterior ha ayudado mucho a la elección de esta tecnología.
- Está basado en componentes, cada vista es un componente, puede haber componentes dentro de componentes, permite tener mucho control sobre la aplicación e incluso ultimar detalles por muy pequeños que sean.
- En los componentes me permite tener separado el código del html.
- Permite la programación reactiva, de forma que pueda realizar aplicaciones asíncronas.

5.1.4. Typescript

Es el lenguaje usado en el lado del cliente. Es un lenguaje creado por Microsoft, es un superset, funcionando como una capa superior a javascript permitiendo al programador crear un código más robusto y fácil de mantener, pudiendo tener un lenguaje estáticamente tipado a contrario de javascript que es dinámicamente tipado.

Figura 42: Logo de Typescript

5.1.5. MongoDB

MongoDB es un tipo de base de datos NoSQL de código abierto y orientado a documentos.

Figura: 43 Logo de Mongodb

A continuación, se explican las razones por las que se ha elegido MongoDB como la base de datos de la aplicación:

- MongoDB permite una gran flexibilidad al no tener que seguir ningún esquema como si hacen las bases de datos relacionales.
- Es fácil de aprender, no lleva mucho tiempo tener un nivel más que aceptable.
- Es gratuita, no se tienen que pagar ningún tipo de licencias.
- Permite tanto la escalabilidad vertical como la horizontal.
- Permite las búsquedas por texto, muy usado en esta aplicación nos ahorra gran tiempo de ejecución.
- MongoDB es muy fácil de conectarse con la aplicación, solo necesita de una URI donde lleva toda la información posible para la conexión.

5.2. Arquitectura del sistema

5.2.1. Patrones de diseño

5.2.1.1. MVVM

El patrón MVVM (Modelo-Vista-Modelo de Vista) es el patrón usado en el framework de Angular. Nos ayuda a separar la lógica de negocios de la interfaz de usuario facilitándonos un buen mantenimiento y la escalabilidad de los proyectos.

Al ser Angular un framework web orientado a componentes, cada componente implementa este patrón y cada componente hace de controlador asociando una vista a un modelo.

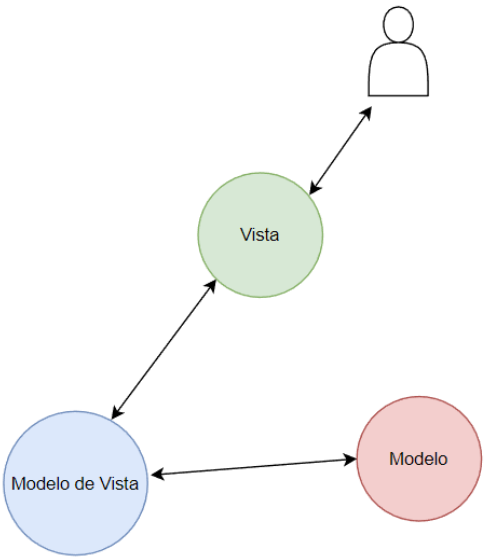

Figura 44: Esquema del patrón MVVM en la aplicación

En el **Modelo** iría toda la lógica de negocio, donde se procesaría todos los datos que vienen o van de la vista.

En la **Vista** tendríamos todo lo relacionado con el código HTML y CSS, es la parte visual de la aplicación.

Modelo de Vista es la capa intermedia entre el Modelo y la Vista procesando todas las peticiones.

5.2.1.2. Observer

El patrón Observer[6] es usado por una librería de Javascript llamada RxJS que permite la programación reactiva usando observables.

El propósito general de los observables es observar el comportamiento de una variable, todo ello mediante la utilización de observables.

[6] https://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o)

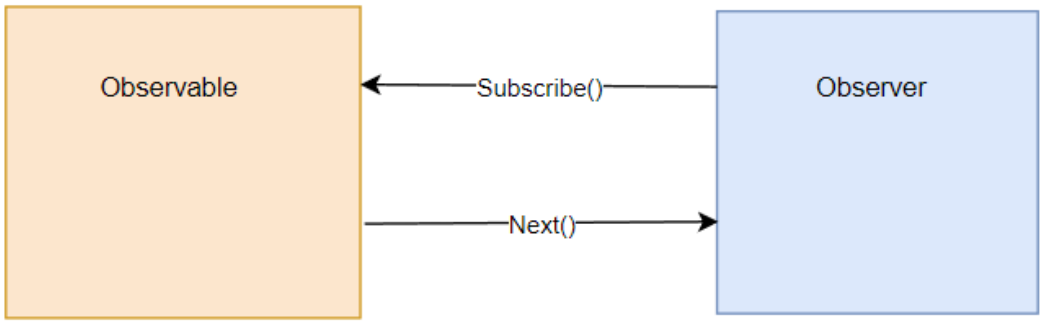

Figura 45: Esquema del patrón Observer en la aplicación

Un Observer se suscribe a un Observable mediante el método “Subscribe”, un Observable solo puede tener un subscriptor y le enviará información al Observer a través del método “Next”.

Al usar Angular, el cual es un framework web orientado a componentes. Los propios componentes harían de observadores pudiendo subscribirse o cancelar la suscripción a un observable.

5.2.1.3. Arquitectura hexagonal

El principio de la arquitectura hexagonal se basa en que se pueda aislar el modelo de negocio, el cual se encontraría en el núcleo del hexágono de los elementos externos de forma que, aunque los elementos externos cambien, estos no afecten a la capa de negocio.

El hexágono representaría las distintas formas que tiene la aplicación de conectarse con el exterior.

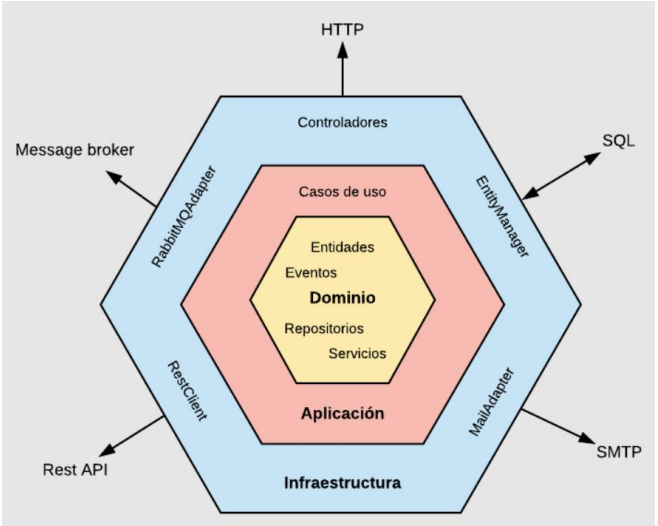

Figura 46 Esquema de la arquitectura hexagonal en la aplicación. Fuente: código en casa[7]

[7] https://medium.com/@edusalguero/arquitectura-hexagonal-59834bb44b7f

5.2.2. Diagrama de componentes

5.2.2.1. Backend

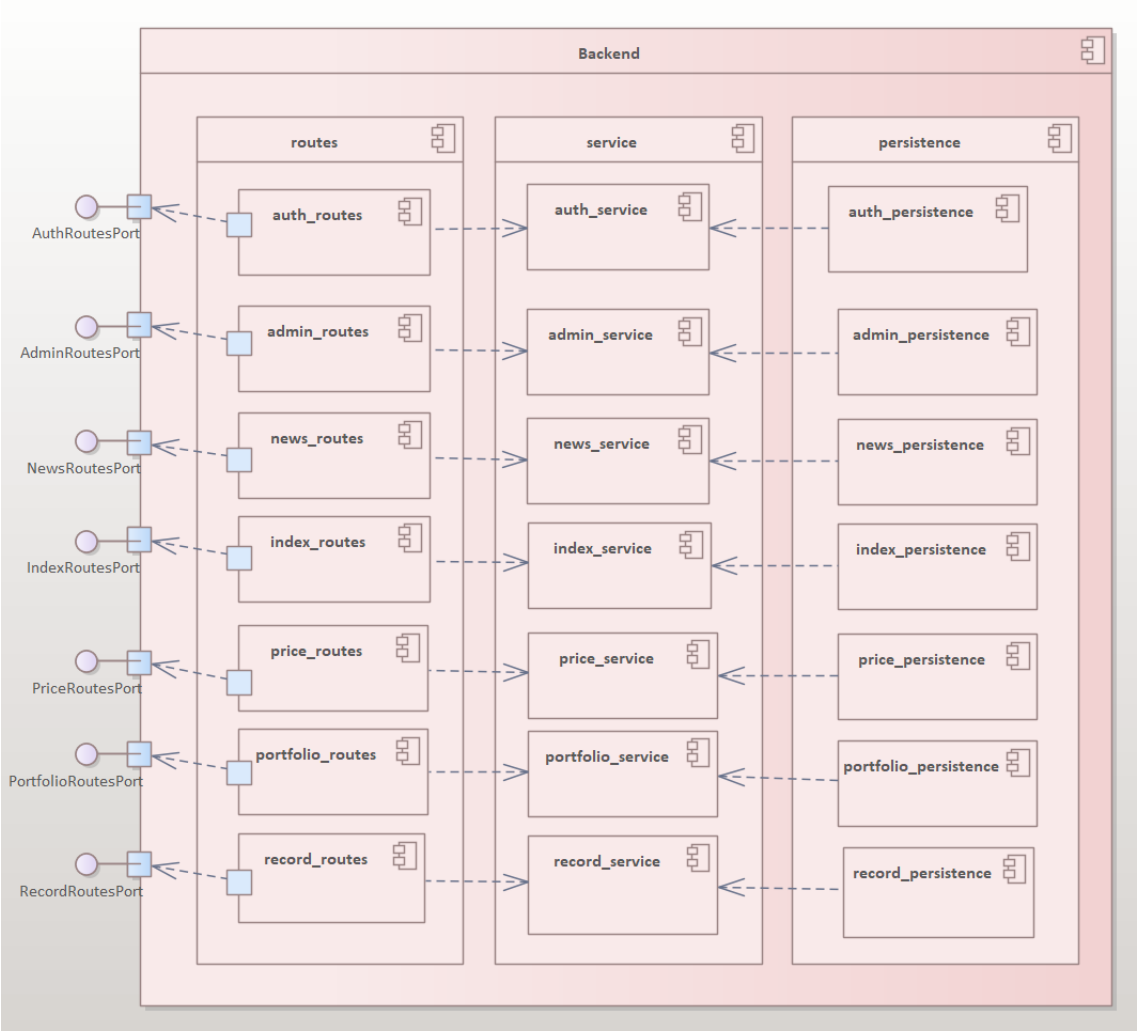

Figura 47: Diagrama de componentes del backend

5.2.2.2. Frontend

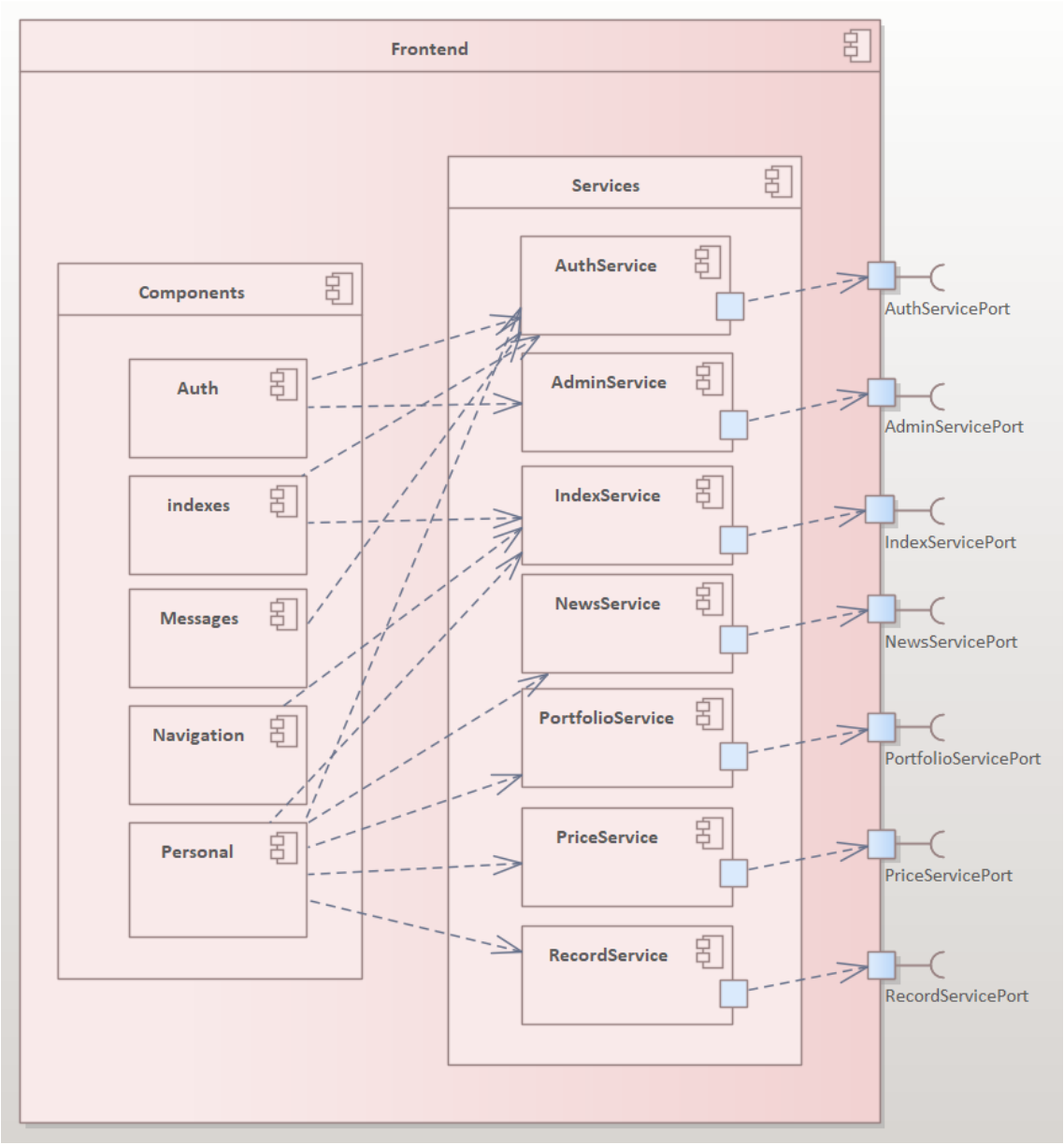

Figura 48: Diagrama de componentes del frontend

5.3. Diseño de la base de datos

5.3.1. Descripción del SGBD NoSQL usado

Para esta aplicación hemos usado MongoDB como gestor de base de datos (SGBD NoSQL).

Se ha usado esta base de datos por el carácter no estructurado que tiene este tipo de bases de datos frente a las bases de datos relacionales. MongoDb lo forman colecciones de documentos, lo cual nos ofrece soluciones más simples y de mayor

rapidez para adaptarnos a los cambios que se han ido haciendo en la aplicación de una forma más dinámica.

La manera de trabajar los datos por MongoDB también es una ventaja si se está usando el lenguaje Python. En este caso el hecho de trabajar con información en formato JSON que es convertible automáticamente a los diccionarios de Python hacen que el guardar y leer datos se haga de la manera más simple posible.

Se ha usado un controlador llamado Motor que está basado en PyMongo (controlador oficial de Python para MongoDB) que soporta peticiones asíncronas.

5.3.2. Modelo conceptual

A continuación, se muestra la estructura principal de los documentos de cada colección. Cada tabla es una colección.

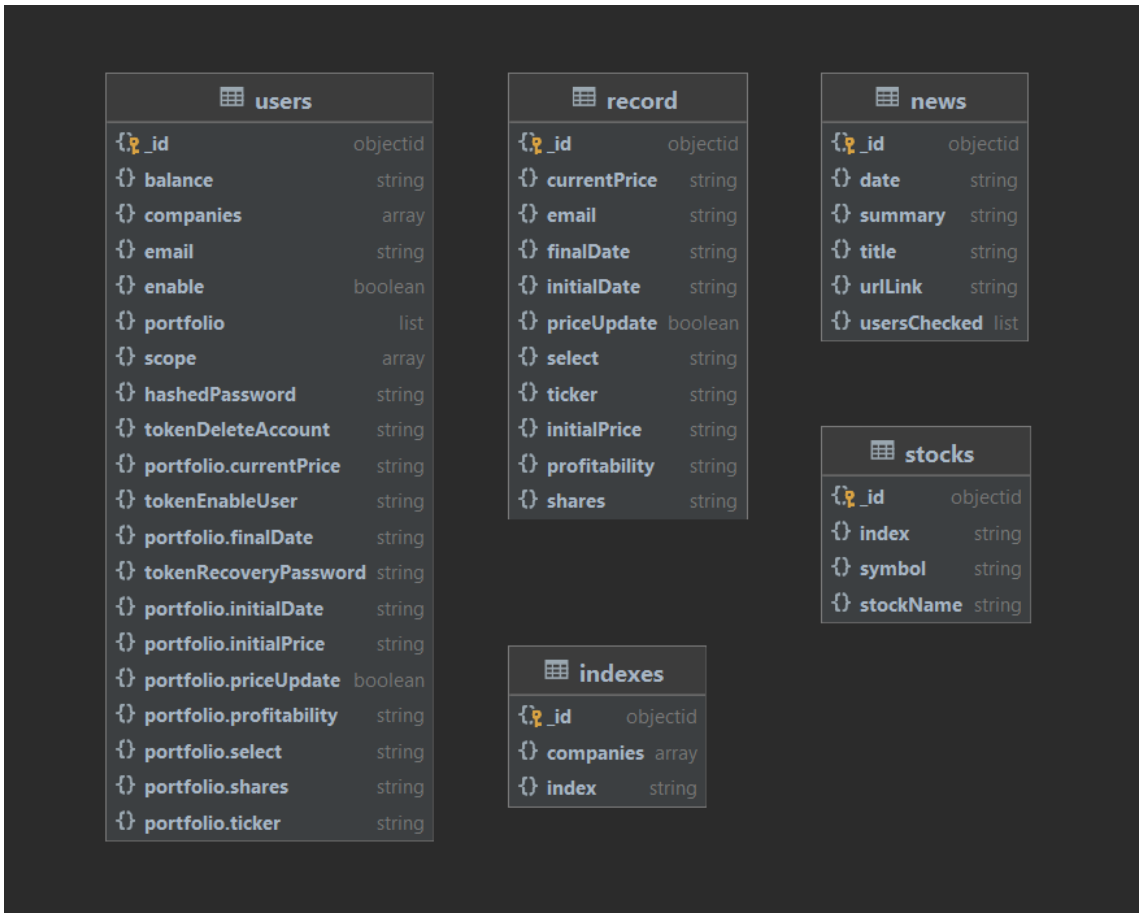

Figura 49: Diagrama de las colecciones de la base de datos

6. Implementación del sistema

6.1. Estándares y normas seguidas

6.1.1. Python PEP 8

El código de la aplicación que se ha desarrollado en este proyecto en el lado del servidor sigue el PEP 8 (la guía de estilos para código Python).

Las pautas proporcionadas por este estilo están destinadas a mejorar la legibilidad del código y hacerlo consistente en el amplio espectro del código de Python.

6.1.2. TypeScript Style

El código de la aplicación que se ha desarrollado en este proyecto en el lado del cliente sigue la guía de estilos utilizando la terminología RFC 2119

6.2. Herramientas y programas usados para el desarrollo

6.2.1. Pycharm

Es el mejor entorno de desarrollo integrado (IDE) para Python, fue creado por la empresa JetBrains.

Figura 50: Captura del programa Pycharm[8]

[8] https://www.jetbrains.com/es-es/pycharm/

Alguna de las características que tiene:

- Asistencia inteligente a la codificación.
- Herramientas de desarrollo integradas.
- Soporta desarrollo web (Backend y Frontend).
- Posee herramientas científicas.
- Es personalizable y multiplataforma.
- Tiene soporte para múltiples bases de datos.
- Depurador Python.

6.2.2. Visual Studio Code

Es un editor de texto creado por Microsoft que tiene soporte nativo para gran variedad de lenguajes.

Figura 51: Captura del programa Visual Code[9]

Algunas características:

- Colores de la sintaxis.
- Seguimiento de cambios.
- Ajustes automáticos de líneas.
- Coincidencias de llaves.

6.2.3. MongoDB Compass

Aplicación multiplataforma que nos permite explorar la estructura de los documentos de las distintas colecciones que tengamos en nuestra base de datos MongDB aparte de permitirnos realizar algunas acciones CRUD.

[9] https://code.visualstudio.com/

Figura 52: Captura del programa Compass[10]

6.3. Librerías y frameworks utilizados

6.3.1. Yfinance

Yfinace es una librería de terceros usada exclusivamente para tratar con la api de Yahoo finance con la cual se extraen los precios históricos y actuales de cada empresa.

6.3.2. Newspaper3k

Newspaper3k[11] es una librería de terceros usada para extraer y limpiar artículos en internet.

6.3.3. Docker

Sistema basado en contenedores que permite desplegar aplicaciones dentro de ellos.

Figura 53: Captura del logo de Docker[12]

6.3.4. Pytest-asyncio

Es una librería escrita en Python que está basada en una librería estándar de Python, como es Pytest y que permite poder testear código asíncrono.

[10] https://www.mongodb.com/es/products/compass
[11] https://newspaper.readthedocs.io/en/latest/
[12] https://www.docker.com/

6.3.5. RxJS

Es una librería de programación reactiva con la que se puede crear código asíncrono usando Observables.

Figura 54: Captura del logo de RxJS[13]

6.3.6. Motor

Es un controlador para aplicaciones asíncronas de Python basado en PyMongo para trabajar con MongoDB y Python. Permite trabajar con corrutinas sin bloquear el acceso a MongoDB.

Figura 55: Captura del logo de Motor[14]

6.3.7. Ng2-charts

Es una librería basada en chart.js con la que se puede crear gráficos en el lado del cliente usando Angular.

6.3.8. Httpx

Es un cliente HTTP escrito en Python, el cual puede proveer de peticiones síncronas como asíncronas.

[13] https://rxjs.dev/
[14] https://motor.readthedocs.io/en/stable/

Figura 56: Captura del logo de HTTPx[15]

Está basado en el popular cliente Requests de Python.

6.3.9. Uvicorn

Es un servidor web de tipo ASGI (Asíncronos Server Gateway Interface) que maneja las conexiones web de un cliente al programa que utilices pudiendo manejar peticiones asíncronas. Es usado en la aplicación para desplegar el backend.

Figura 57: Captura del logo de Uvicorn[16]

6.3.10. Nginx

Es un servidor web de alto rendimiento que ofrece una arquitectura altamente escalable, modular, basada en eventos y asíncrona. Es usado en la aplicación para desplegar el frontend.

[15] https://www.python-httpx.org/
[16] https://www.uvicorn.org/

Figura 58: Captura del logo de Nginx[17]

6.3.11. Bootstrap

Es un conjunto de herramientas para desarrollar aplicaciones web del lado del cliente.

Figura 59: Captura del logo de Bootstrap[18]

Bootstrap ofrece una serie de componentes que facilitan el trabajo del programador web ya que puede ser utilizado en cualquier trabajo de diseño web.

6.3.12. Angular Material

Es una biblioteca de componentes de interfaz de usuario diseñado para trabajar con Angular ya que implementa diseños hechos con componentes y servicios.

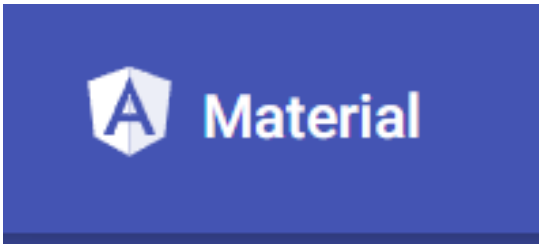

Figura 60: Captura del logo de Angular Material[19]

[17] https://www.nginx.com/
[18] https://getbootstrap.com/
[19] https://material.angular.io/

6.4. Creación del sistema

6.4.1. Problemas encontrados

Realizar aplicaciones full-stack siempre puede ser un poco tedioso por el hecho de que cada vez que se implementa una funcionalidad en el backend siempre es mejor implementarla seguidamente en el frontend, lo cual incrementa ligeramente la carga de trabajo diaria. Aun así, se han encontrado algunos problemas que se pasan a detallar.

6.4.1.1. Lenguajes y frameworks

Python es un lenguaje que se da muy poco en la carrera, solamente utilizado para temas puntuales.

Es más normal que los estudiantes de ingeniería informática acaben diseñando su backend en lenguajes más usados durante la carrera como serian Java o JavaScript.

El hecho de tener que aprender un framework como Fastapi con menos de 4 años de vida hace que el tiempo de aprendizaje haya sido mayor al no tener tantos ejemplos en internet a los que poder consultar.

Con TypesScript/Angular he tenido problemas similares a los tenidos con Fastapi, el hecho de aprender a usar Angular, que no se da en la ingeniería, ha hecho que la curva de aprendizaje haya sido elevada y el tiempo de trabajo se haya excedido.

6.4.1.2. Asíncrono I/O

Al crear una aplicación enteramente asíncrona han surgido algunos problemas al no tener práctica en este tipo de aplicaciones.

A partir de la versión 3.5 de Python las corrutinas fueron implementadas en la librería estándar. La implementación no ha sido complicada, pero se han tenido problemas a la hora de lidiar con los tests unitarios a la hora de llamar a estas funciones especiales.

6.4.1.3. Verificación Gmail en 2 pasos

A partir de mayo del 2022 Gmail cambió su protocolo de autentificación de usuarios, se ha tenido que cambiar la implementación para mandar emails desde la aplicación.

Anteriormente se tenía que crear una cuenta de correo en Gmail y tener las credenciales en la misma aplicación para poder mandar correos desde ese correo a otros correos, lo cual era bastante peligroso a la hora de guardar datos relativamente importantes.

Con el cambio de Gmail a la verificación en dos pasos, seguimos necesitando un correo de Gmail, pero ahora se crea una contraseña a partir de cualquier palabra que el

usuario elija y con esa contraseña generada por el mismo Gmail podemos mandar correos a cualquier correo.

6.4.1.4. Comunicación entre componentes en Angular

Angular es un framework orientado a componentes, cada vista está dentro de un componente. Se sabía que iba a ser un problema encontrar la mejor solución para la comunicación entre componentes que están al mismo nivel y no pertenecen de Padre-a-hijo o de hijo-a-padre.

Ya que estábamos creando una aplicación asíncrona y el backend y la base de datos soportaban peticiones asíncronas, se quería que el código en Angular también las soportase desde el punto de vista del programador.

La librería RxJS es muy utilizada en Javascript para la programación reactiva, el hecho de usar Observables para comunicarse de un componente a otro ha hecho que la curvatura de aprendizaje de angular fuese un poquito más elevada pero muy satisfactoria.

6.4.1.5. Tiempo de respuesta en Python

Python es un lenguaje interpretado y como tal uno de sus problemas es la velocidad de ejecución.

Al no disponer de un api oficial donde bajarse los precios ya que estas suelen ser de pago, hemos tirado de la api que nos proporciona Yahoo Finance y para ello se ha usado una librería de terceros programada en Python que usa esa api y te ordena los datos de salida que vienen desordenados por la api.

En algunas vistas el tiempo de respuesta se amplía por culpa de este problema.

6.4.2. Diagrama de paquetes

6.4.2.1. Backend

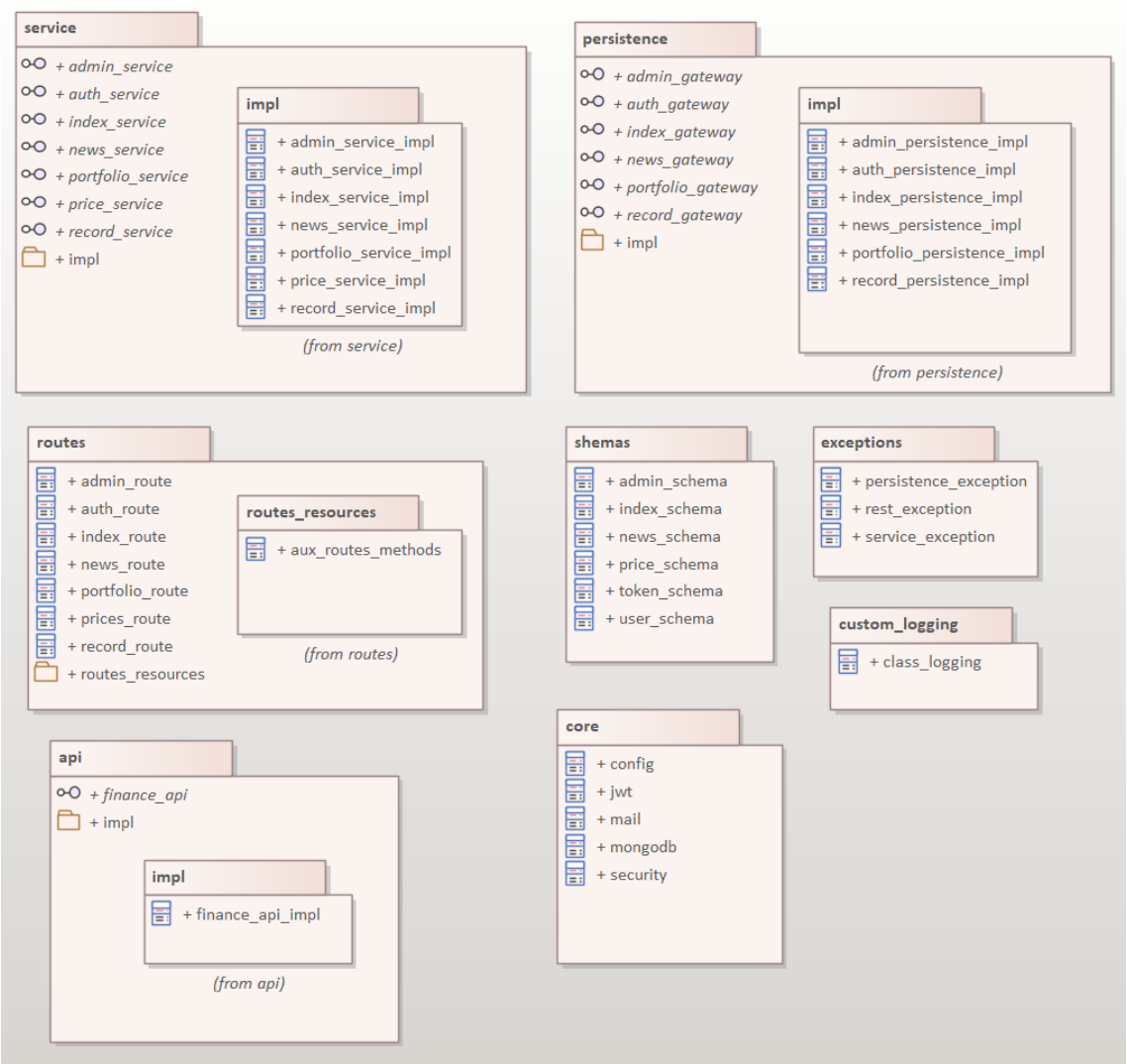

Figura 61: Diagrama de paquetes del backend

6.4.2.2. Frontend

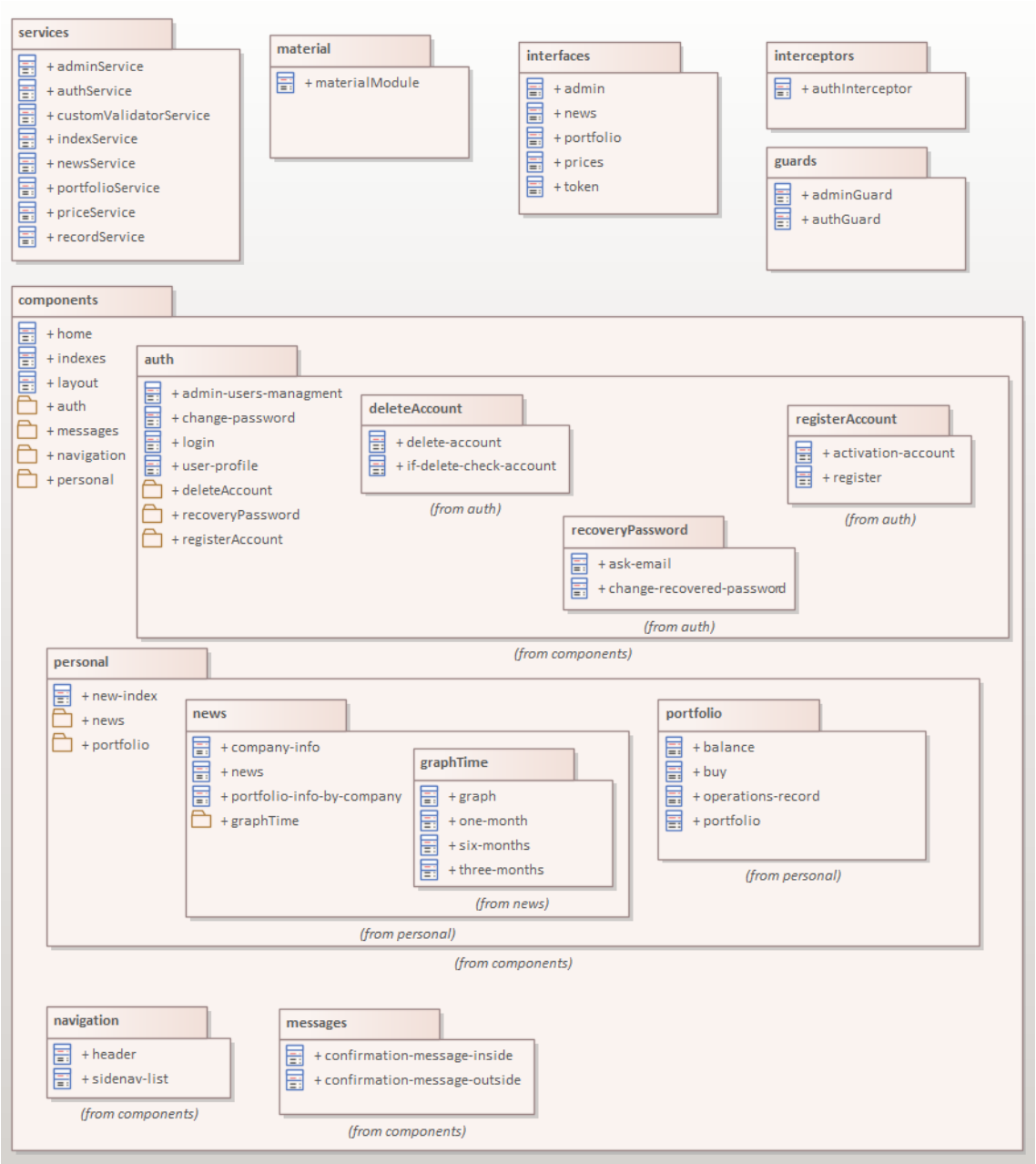

Figura 62: Diagrama de paquetes del frontend

6.5. Descripción detallada de las clases

Aunque la aplicación web se haya escrito en inglés y los comentarios del código también, esta descripción de las clases y los componentes se hará en español.

6.5.1. Backend

6.5.1.1. Admin_service

Interfaz que proporciona operaciones relacionadas con las acciones del administrador.

Métodos	
Tipo	Descripción
List[dict]	get_users_no_admin_serv() Obtiene una lista con todos los usuarios registrados que no son el administrador.
str	Enable_disable_account_serv(email: str) Activa o desactiva según el estado en el que previamente esté, devuelve un mensaje de confirmación.
None	Change_balance_serv(email: str, balance: dict) Cambia el balance de un usuario.

6.5.1.2. Auth_service

Interfaz que proporciona operaciones relacionadas con las acciones del usuario que tienen que ver con su cuenta.

Métodos	
Tipo	Descripción
dict	Get_user_serv(email: str) Devuelve la información del usuario
None	Créate_user_serv(email: str, password: str) Crea un nuevo usuario solicitando su email y su contraseña
dict	Authenticate_user(username: str, password: str) El usuario se loguea en la aplicación introduciendo su nombre de usuario y su contraseña
None	Enable_user_serv(token: str) El usuario es activado en la aplicación mediante un token
None	Créate_token_recovery_password_serv(email: str) Crea un token para recuperar la contraseña solicitando el email del usuario
None	Recovery_password_with_token_serv(token: str, new_password: str) El usuario recupera el acceso a la aplicación mediante un token y ingresando una nueva contraseña
None	Créate_token_delete_account_serv(email: str) Crea un token para borrar su cuenta
None	delete_account_serv(token: str) Se desactiva la cuenta del usuario mediante un token
None	Change_password_serv(email: str, password: str) Cambia la contraseña del usuario solicitándole su email y la nueva contraseña

6.5.1.3. Index_service

Interfaz que proporciona operaciones relacionadas con los índices.

Métodos	
Tipo	Descripción
List[dict]	Find_index_serv(name: str) Devuelve la lista de compañías de un índice concreto
List[str]	Get_tickers_by_name_serv(index_name: str) Devuelve la lista de tickers de un índice concreto
None	Save_new_index_serv(companies: List[str], email: str) Guarda un nuevo índice creado por el usuario
List[str]	Get_tickers_new_index_serv(email: str)

	Devuelve una lista de tickers del nuevo índice creado por el usuario
dict	Get_new_index_serv(email: str) Devuelve la información del usuario
None	Delete_companies_from_index_serv(companies: List[str], email: str) Borra las compañías seleccionadas por el usuario del índice creado por el
None	Delete_new_index_serv(email: str) Borra el nuevo indice
str	Get_index_name_serv(ticker: str) Devuelve el nombre de un índice en concreto solicitando el ticker de una compañía que pertenezca a ese índice.

6.5.1.4. News_service

Interfaz que proporciona operaciones relacionadas con noticias.

Métodos	
Tipo	Descripción
List[dict]	get_news_serv(company_name: str) Devuelve una lista de las noticias relacionadas con una misma compañía
List[int]	Get_number_of_news_serv(email: str) Devuelve una lista con el número de noticias relacionadas con el índice de compañías creado por el usuario
None	Save_user_news_checked_serv(email: str, title: str) Guarda el email de un usuario que haya visto una noticia en concreto

6.5.1.5. Portfolio_service

Interfaz que proporciona operaciones relacionadas con el portafolios

Métodos	
Tipo	Descripción
None	Save_operation_portfolio_serv(email: str, data: dict) Guarda una nueva operación en el portafolios
None	Delete_operation_portfolio_serv(email: str, data: dict) Borra una operación del portafolio
None	Set_current_price_portfolio_serv(email: str) Actualiza los precios actuales de todas las operaciones del portafolio
List[str]	Get_profitability_portfolio_serv(email: str) Devuelve una lista con cada una de las rentabilidades de cada operación abierta en el portafolio
str	Get_balance_serv(email: str) Devuelve el balance de un usuario, calculando la suma o la resta de todas las operaciones abiertas en el portafolio
List[dict]	Get_live_operations_serv(email: str, ticker: str) Devuelve una lista con las operaciones abiertas en el portafolio filtradas por su ticker

6.5.1.6. Price_service

Interfaz que proporciona operaciones relacionadas los precios de cada compañía.

Métodos	
Tipo	Descripción
List[dict]	Get_historical_prices_serv(ticker: str) Devuelve una lista con los precios históricos y sus fechas de una compañía filtrándolos por su ticker
List[str]	Get_last_prices_serv(email: str) Devuelve una lista de los precios actuales de las compañías pertenecientes al nuevo índice creado por el usuario
List[str]	Get_profitabilities_serv(email: str) Devuelve una lista de las rentabilidades de las compañías pertenecientes al nuevo índice creado por el usuario

6.5.1.7. Record_service

Interfaz que proporciona operaciones relacionadas con el historial de operaciones.

Métodos	
Tipo	Descripción
List[dict]	get_records_serv(email: str) Devuelve una lista con las operaciones cerradas del portafolio del usuario
List[dict]	Get_records_by_ticker_serv(email: str, ticker: str) Devuelve una lista con las operaciones cerradas del portafolio del usuario filtradas por el ticker de la compañía

6.5.2. Frontend

Estas son las distintas clases del paquete service que contienen la lógica de los componentes.

6.5.2.1. Admin.service

Métodos	
Tipo	Descripción
Observable<IAdminInfoUser>	getUsers() Método que devuelve una lista de usuarios con su información.
Observable<string>	updateStateUser(email: string) Método que actualiza el estatus de un usuario.
Observable<string>	changeBalanceService(balance: string, email: string) Método que actualiza el balance de un usuario.

6.5.2.2. Auth.service

Métodos	
Tipo	Descripción
Observable<string>	register(formData: FormGroup) Método que manda el email del usuario y una contraseña para registrarse en la aplicación.

Observable<IToken>	login(formData: FormGroup) Método que manda las credenciales de un usuario devolviéndole un token de usuario.
Observable<string>	activateAccount(token: string) Método que manda un token.
Observable<string>	askEmail(formData: FormGroup) Metodo que manda un email para crear un token.
Observable<string>	sendTokenAndPassword(formData: FormGroup) Metodo que manda un token y una contraseña.
Observable<string>	checkAccount(formData: FormGroup) Metodo que manda las credenciales de un usuario.
Observable<string>	deleteAccount(token: string) Método que manda un token para borrar desactivar una cuenta.
Observable<string>	changePassword(formData: FormGroup) Método que manda la contraseña de un usuario.

6.5.2.3. Index.service

Métodos	
Tipo	Descripción
Observable<string[]>	getIndex() Método que devuelve los nombres de las empresas del Ibex35.
Observable<string[]>	getTickersToMainIndex() Método que devuelve una lista con los tickers del ibex35.
Observable<string>	saveNewIndex(formData: FormGroup) Método que manda las empresas que van a formar el nuevo índice.
Observable<string[]>	getNewIndex() Método que devuelve información de las compañías del nuevo índice.
Observable<string[]>	getTickers() Método que devuelve una lista con los tickers de las empresas del nuevo índice.
Observable<string>	deleteCompaniesFromIndexByUser(formData: FormGroup) Método que mandas las empresas que son borradas del nuevo índice.
Observable<string>	deleteNewIndex() Método que borra un nuevo índice entero de una vez.
Observable<string>	getIndexName(ticker: string) Método que devuelve el nombre del índice donde cotiza una empresa.

6.5.2.4. News.service

Métodos	
Tipo	Descripción

Observable<INews[]>	getNews() Método que devuelve las noticias relacionadas con una empresa.
Observable<number[]>	getNumberOfNews() Método que devuelve una lista con las noticias no leídas de cada empresa del nuevo índice.
Observable<string>	saveUserNewsCheked(title: string) Método que envía un título de una noticia para saber que ha sido vista por un usuario.

6.5.2.5. Portfolio.service

Métodos	
Tipo	Descripción
Observable<string>	savePortfolio(data: IPortfolio) Método que envía una nueva operación del portafolio.
Observable< IPortfolio[] >	getPortfolio() Método que devuelve una lista de operaciones en el portafolio.
Observable<string>	deleteTrade(data: IPortfolio) Método que borra una operación del portafolio.
Observable<string[]>	getPortfolioProfitability() Método que devuelve las rentabilidades de las operaciones del portafolio.
Observable<number>	getBalance() Método que devuelve el balance de un usuario.
Observable<IPortfolio[]>	getLiveOperations(ticker: string) Método que devuelve una lista de las operaciones abiertas en el portafolio.
Observable<string>	setPriceCurrent() Método que actualiza los precios de cotización de las empresas del nuevo índice.

6.5.2.6. Prices.service

Métodos	
Tipo	Descripción
Observable<string[]>	getLastPrices() Método que devuelve los últimos precios de cotización de cada empresa guardada en el nuevo indice.
Observable<string[]>	getProfitabilities() Método que devuelve las rentabilidades de todas las empresas guardadas en el nuevo índice.
Observable<IHistoricalPrices[]>	getGraph(ticker: string, timer: string) Método que devuelve una lista con los precios de un ticker junto a sus fechas.

6.5.2.7. Record.service

Métodos

Tipo	Descripción
Observable<IPortfolio[]>	getRecords() Método que devuelve una lista con las operaciones cerradas y guardadas en el historial de un usuario.
Observable< IPortfolio[]>	getRecordsByTicker(ticker: string) Método que devuelve una lista con las operaciones cerradas y guardadas en el historial de un usuario filtradas por un ticker.

7. Desarrollo de las pruebas

7.1. Pruebas de aceptación

7.1.1. Primer pase de la batería de pruebas

A continuación, se muestra los resultados de la evaluación de las pruebas diseñadas en el apartado 4.6, divididos según los casos de uso.

7.1.1.1. Caso de uso 1: Autentificarse

Caso de prueba	Resultado	Observaciones
1	✓	
2	✓	
3	Fallo	Se permite el acceso a un usuario no activado.

7.1.1.2. Caso de uso 2: Salir

Caso de prueba	Resultado	Observaciones
4	✓	
5	✓	
6	✓	

7.1.1.3. Caso de uso 3: Registrarse

Caso de prueba	Resultado	Observaciones
7	✓	
8	✓	

7.1.1.4. Caso de uso 4: Recuperar contraseña

Caso de prueba	Resultado	Observaciones
9	✓	
10	✓	

7.1.1.5. Caso de uso 5: Cambiar contraseña

Caso de prueba	Resultado	Observaciones
11	✓	

7.1.1.6. Caso de uso 6: Borrar cuenta

Caso de prueba	Resultado	Observaciones
12	✓	
13	✓	

7.1.1.7. Caso de uso 7: Vista del portafolio

Caso de prueba	Resultado	Observaciones
14	Fallo	El botón “Create” permanece habilitado.

7.1.1.8. Caso de uso 8: Crear índice personal

Caso de prueba	Resultado	Observaciones
15	✓	
16	Fallo	Se permite añadir empresas repetidas.

7.1.1.9. Caso de uso 9: Borrar empresa en el índice personal

Caso de prueba	Resultado	Observaciones
17	✓	

7.1.1.10. Caso de uso 10: Borrar índice personal

Caso de prueba	Resultado	Observaciones
18	✓	

7.1.1.11. Caso de uso 11: Crear operación en el portafolio

Caso de prueba	Resultado	Observaciones
19	✓	
20	Fallo	El sistema no inhabilita el botón.

7.1.1.12. Caso de uso 12: Cerrar operación en el portafolio

Caso de prueba	Resultado	Observaciones
21	✓	

7.1.1.13. Caso de uso 13: Filtrar empresa en el historial

Caso de prueba	Resultado	Observaciones
22	✓	

7.1.1.14. Caso de uso 15: Seleccionar noticias

Caso de prueba	Resultado	Observaciones
23	Fallo	El sistema no descuenta en el índice personal las noticias ya leídas por el usuario.

7.1.1.15. Caso de uso 16: Actualizar estado del usuario

Caso de prueba	Resultado	Observaciones
24	✓	
25	Fallo	El sistema permite a un usuario no autorizado cambiar el estado de un usuario.

7.1.1.16. Caso de uso 17: Actualizar balance del usuario

Caso de prueba	Resultado	Observaciones
26	✓	
27	Fallo	El sistema permite a un usuario no autorizado cambiar el balance de un usuario.

7.1.2. Segundo pase de la batería de pruebas

Una vez resueltos los fallos del primer pase de pruebas, se realiza otro pase de pruebas con los siguientes resultados.

7.1.2.1. Caso de uso 1: Autentificarse

Caso de prueba	Resultado	Observaciones
1	✓	
2	✓	
3	✓	

7.1.2.2. Caso de uso 2: Salir

Caso de prueba	Resultado	Observaciones
4	✓	
5	✓	
6	✓	

7.1.2.3. Caso de uso 3: Registrarse

Caso de prueba	Resultado	Observaciones
7	✓	
8	✓	

7.1.2.4. Caso de uso 4: Recuperar contraseña

Caso de prueba	Resultado	Observaciones
9	✓	
10	✓	

7.1.2.5. Caso de uso 5: Cambiar contraseña

Caso de prueba	Resultado	Observaciones
11	✓	

7.1.2.6. Caso de uso 6: Borrar cuenta

Caso de prueba	Resultado	Observaciones
12	✓	
13	✓	

7.1.2.7. Caso de uso 7: Vista del portafolio

Caso de prueba	Resultado	Observaciones
14	✓	

7.1.2.8. Caso de uso 8: Crear índice personal

Caso de prueba	Resultado	Observaciones
15	✓	
16	✓	

7.1.2.9. Caso de uso 9: Borrar empresa en el índice personal

Caso de prueba	Resultado	Observaciones
17	✓	

7.1.2.10. Caso de uso 10: Borrar índice personal

Caso de prueba	Resultado	Observaciones
18	✓	

7.1.2.11. Caso de uso 11: Crear operación en el portafolio

Caso de prueba	Resultado	Observaciones
19	✓	
20	✓	

7.1.2.12. Caso de uso 12: Cerrar operación en el portafolio

Caso de prueba	Resultado	Observaciones

21	✓	

7.1.2.13. Caso de uso 13: Filtrar empresa en el historial

Caso de prueba	Resultado	Observaciones
22	✓	

7.1.2.14. Caso de uso 15: Seleccionar noticias

Caso de prueba	Resultado	Observaciones
23	✓	

7.1.2.15. Caso de uso 16: Actualizar estado del usuario

Caso de prueba	Resultado	Observaciones
24	✓	
25	✓	

7.1.2.16. Caso de uso 17: Actualizar balance del usuario

Caso de prueba	Resultado	Observaciones
26	✓	
27	✓	

7.2. Pruebas unitarias

Las pruebas unitarias están divididas entre capas.

A su vez las pruebas unitarias están divididas en módulos, estos son los correspondientes módulos:

- **Admin**: tiene que ver con todo lo relacionado a las funcionalidades de los administradores.
- **Auth**: tiene que ver con todo lo relacionado a las funcionalidades básicas de un usuario.
- **Index**: tiene que ver con todo lo relacionado con el uso de índices.
- **News**: tiene que ver con todo lo relacionado con el uso de las noticias.
- **Portfolio**: tiene que ver con todo lo relacionado con el portafolio de acciones.
- **Price**: tiene que ver con todo lo relacionado con la obtención de precios de cotización de las distintas empresas.
- **Record**: tiene que ver con todo lo relacionado con el historial de operaciones.

7.2.1. Capa de servicio

7.2.1.1. Test_admin_serv.py

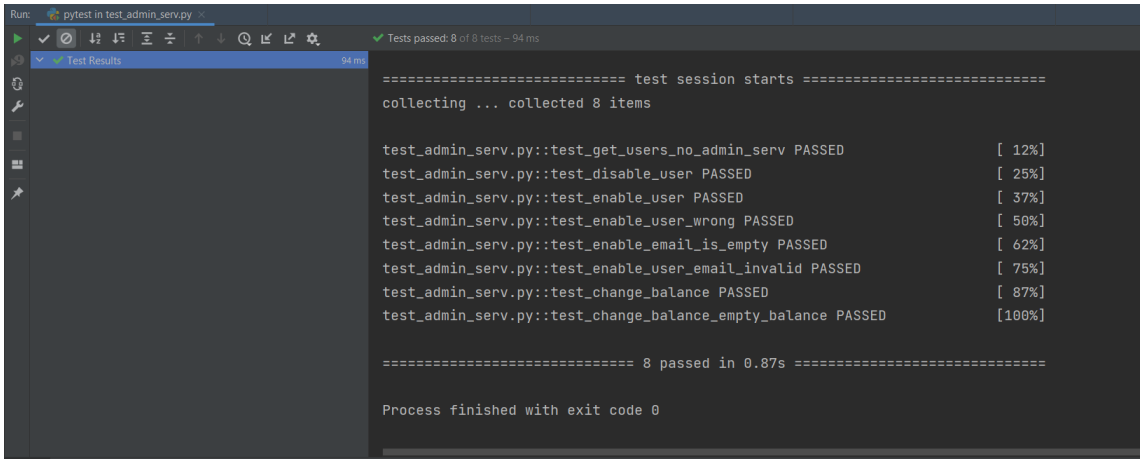

Figura 63: Pruebas unitarias - admin - capa de servicio

7.2.1.2. Test_auth_serv.py

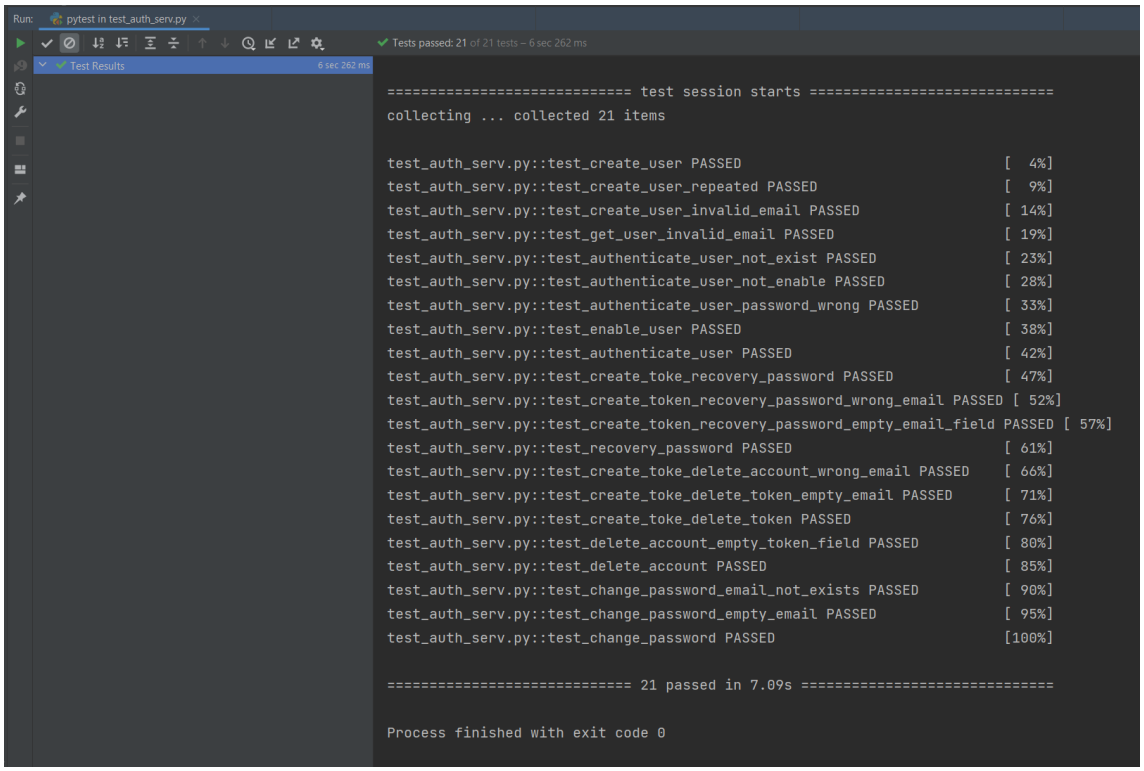

Figura 64: Pruebas unitarias - auth - capa de servicio

7.2.1.3. Test_index_serv.py

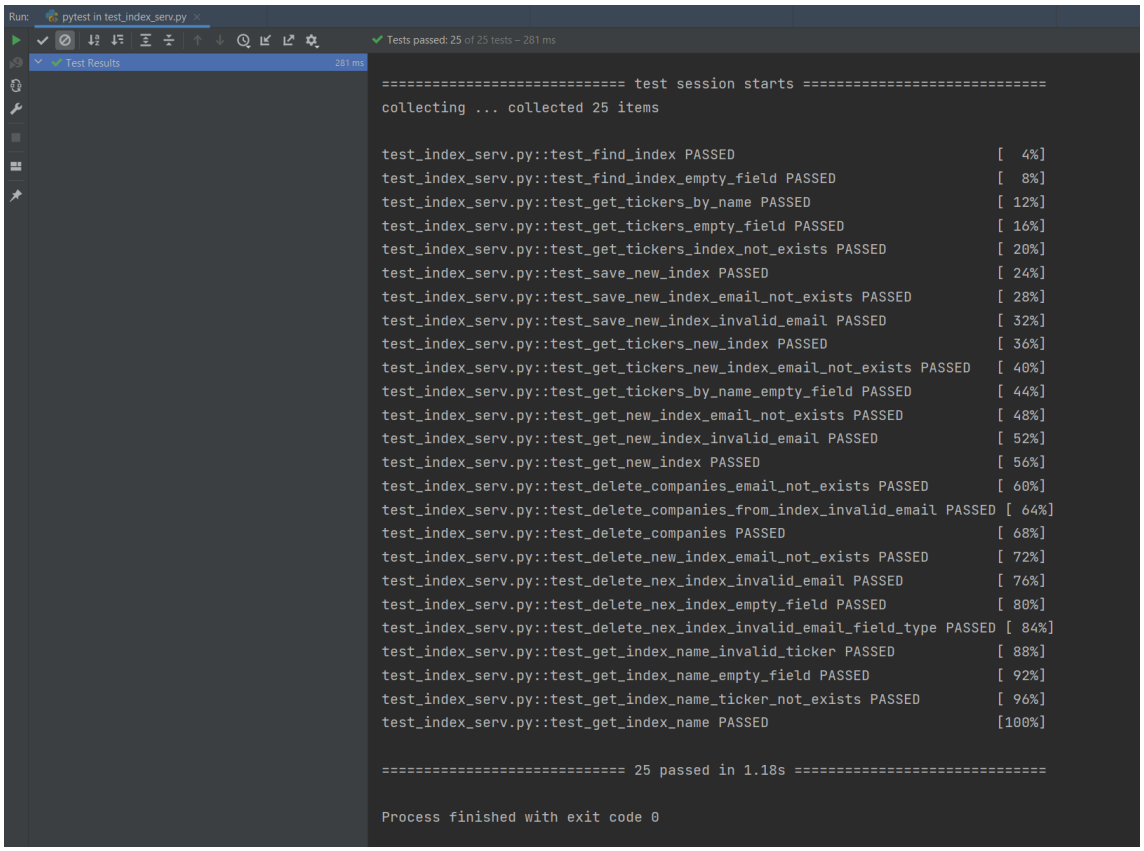

Figura 65: Pruebas unitarias - index - capa de servicio

7.2.1.4. Test_news_serv.py

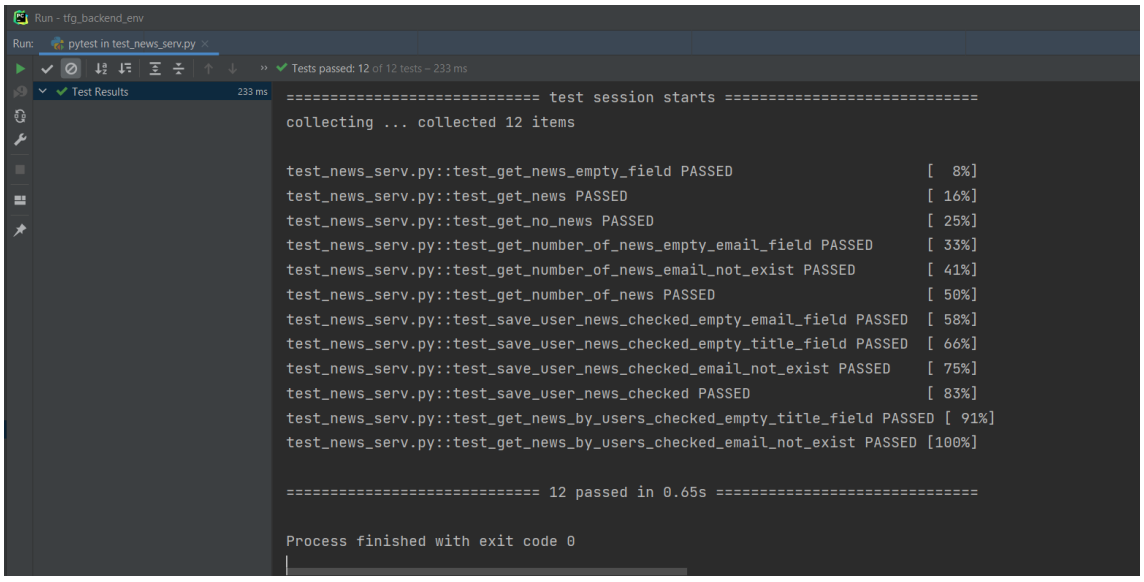

Figura 66: Pruebas unitarias - news - capa de servicio

7.2.1.5. Test_portfolio.serv.py

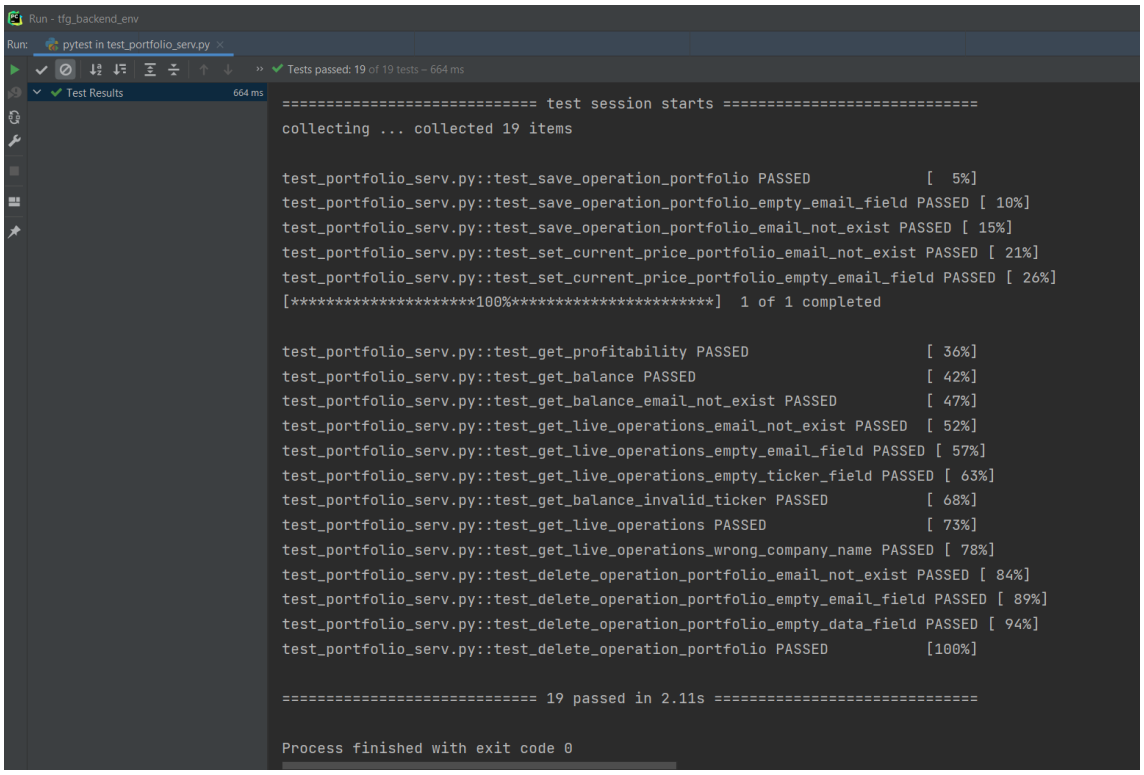

Figura 67: Pruebas unitarias - portfolio - capa de servicio

7.2.1.6. Test_price_serv.py

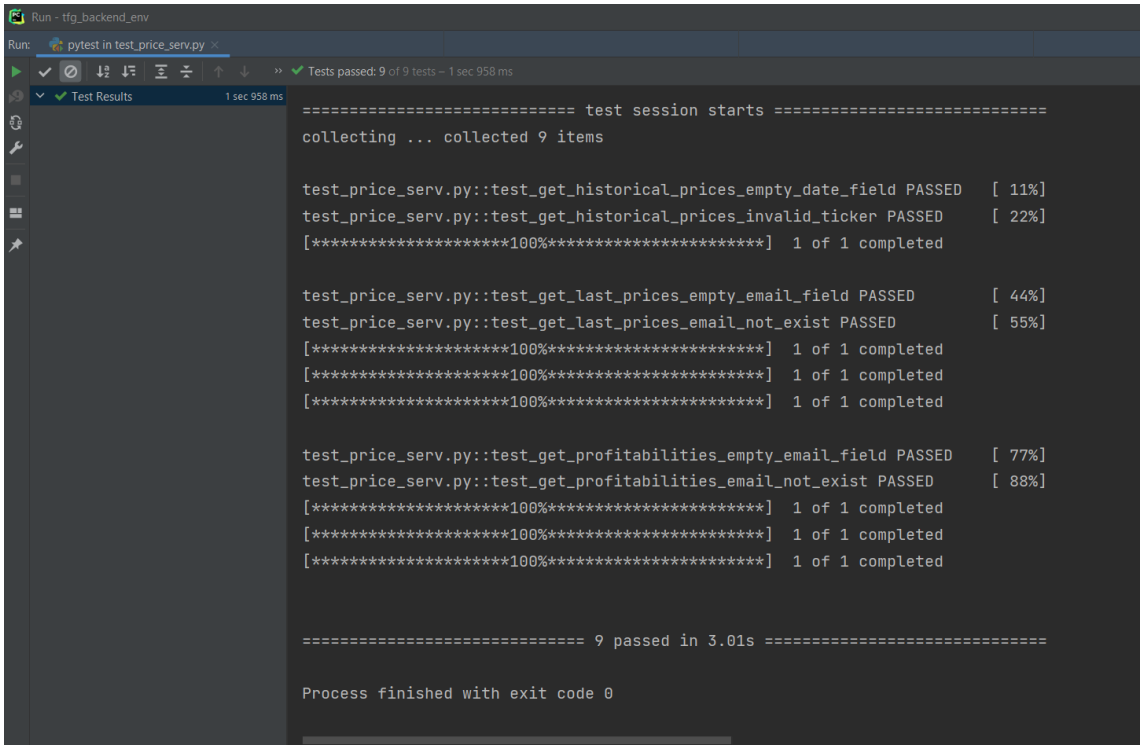

Figura 68: Pruebas unitarias - price - capa de servicio

7.2.1.7. Test_record_serv.py

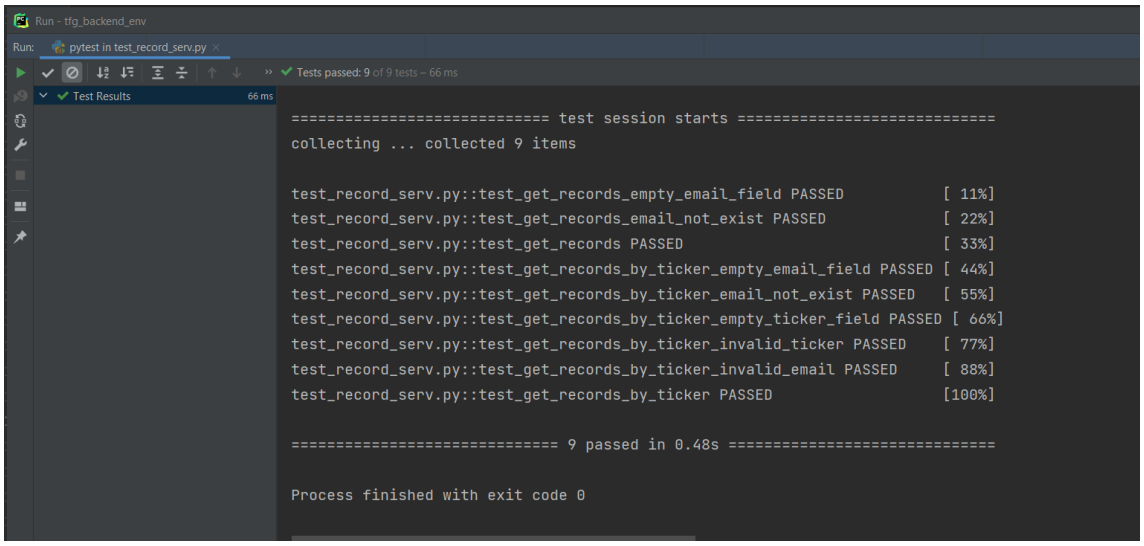

Figura 69: Pruebas unitarias - record - capa de servicio

7.2.2. Capa de los endpoints

7.2.2.1. Admin_test_routes.py

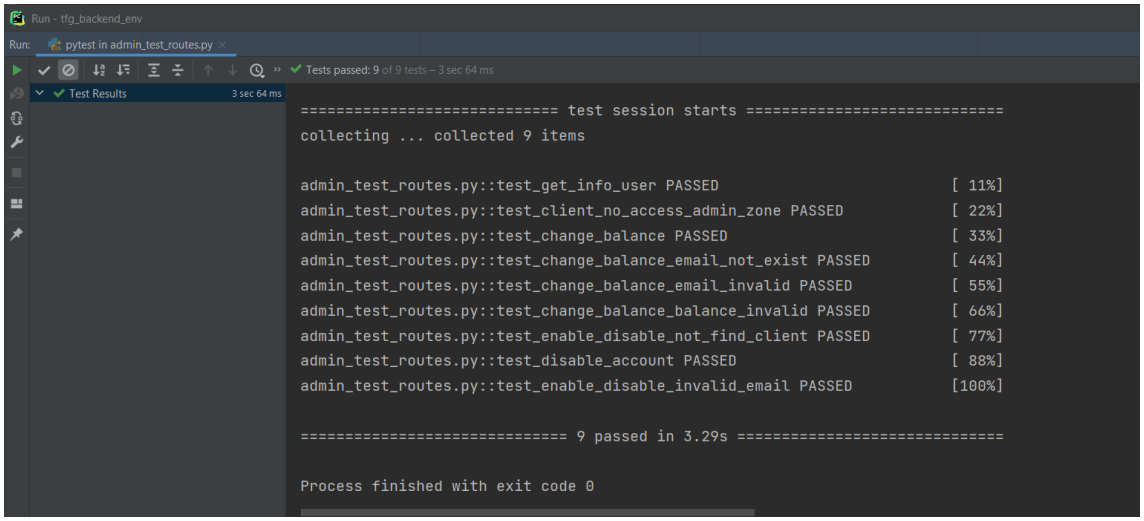

Figura 70: Pruebas unitarias - admin - capa de los endpoints

7.2.2.2. Auth_test_routes.py

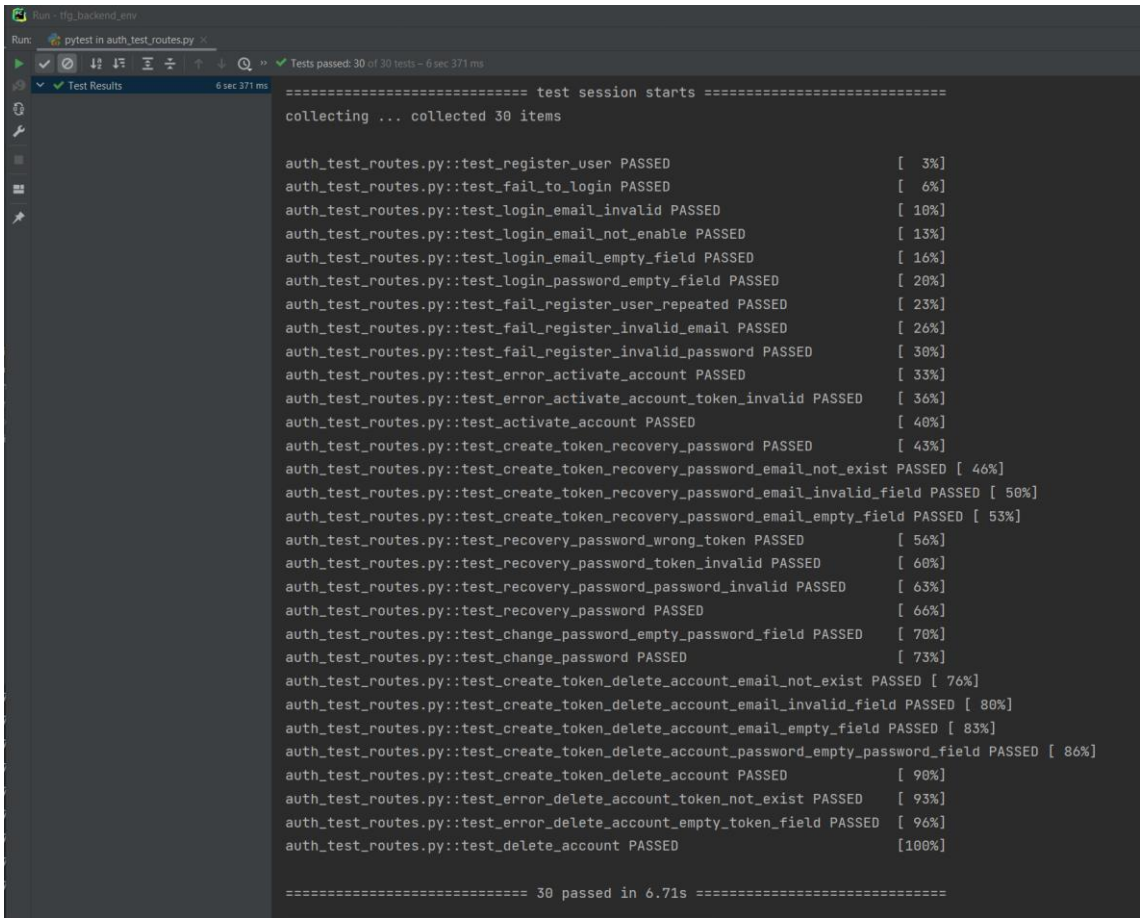

Figura 71: Pruebas unitarias - auth - capa de los endpoints

7.2.2.3. Index_test_routes.py

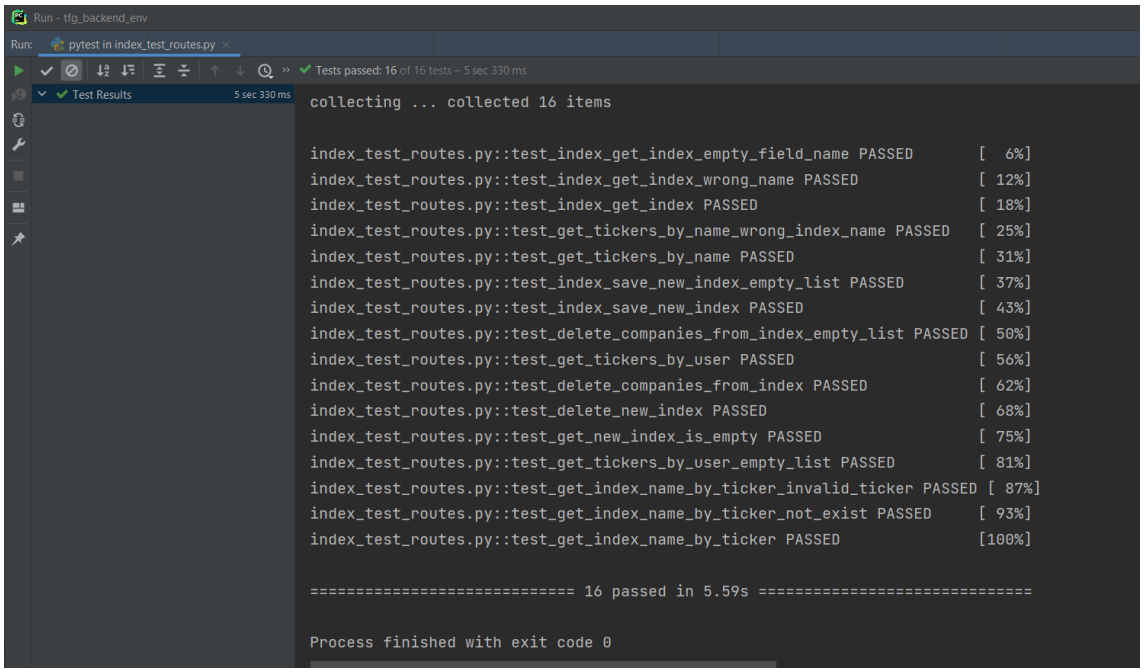

Figura 72: Pruebas unitarias - index - capa de los endpoints

7.2.2.4. News_test_routes.py

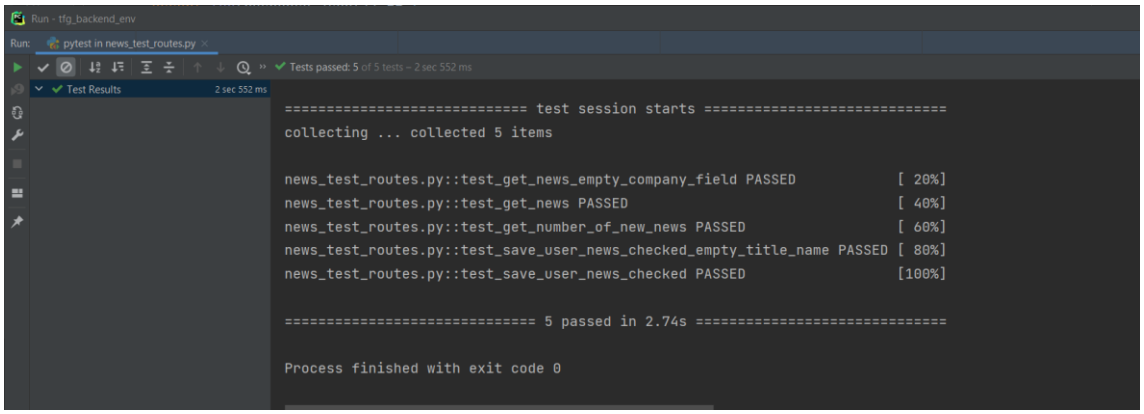

Figura 73: Pruebas unitarias - news - capa de los endpoints

7.2.2.5. Portfolio_test_routes.py

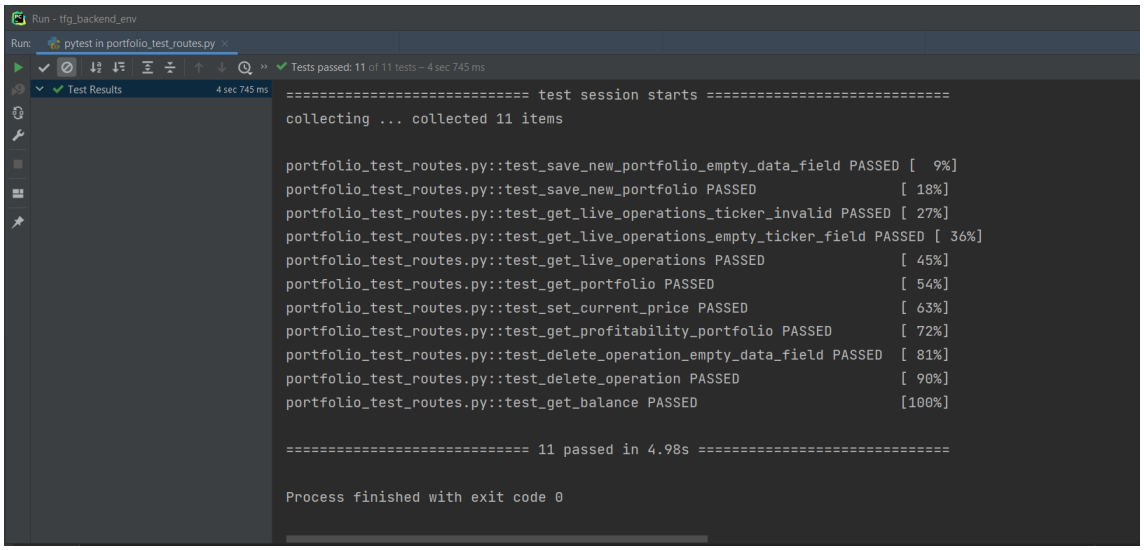

Figura 74: Pruebas unitarias - portfolio - capa de los endpoints

7.2.2.6. Price_test_routes.py

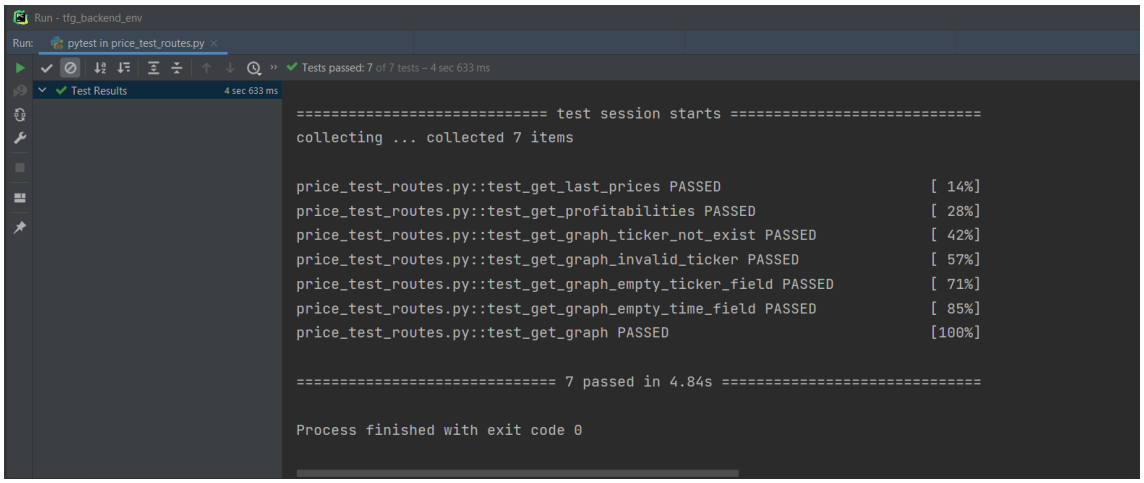

Figura 75: Pruebas unitarias - price - capa de los endpoints

7.2.2.7. Record_test_routes.py

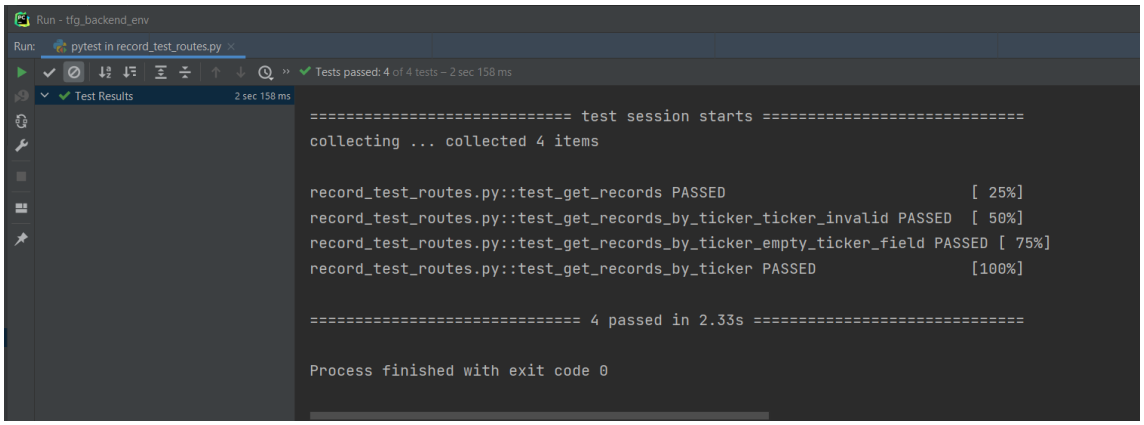

Figura 76: Pruebas unitarias - record - capa de los endpoints

7.2.3. Capa de la persistencia

7.2.3.1. Admin_test_pers.py

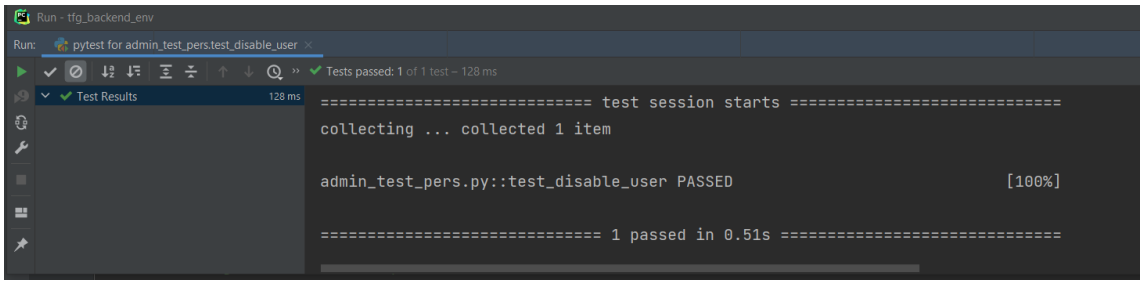

Figura 77: Pruebas unitarias - admin - capa de la persistencia

7.2.3.2. Auth_test_pers.py

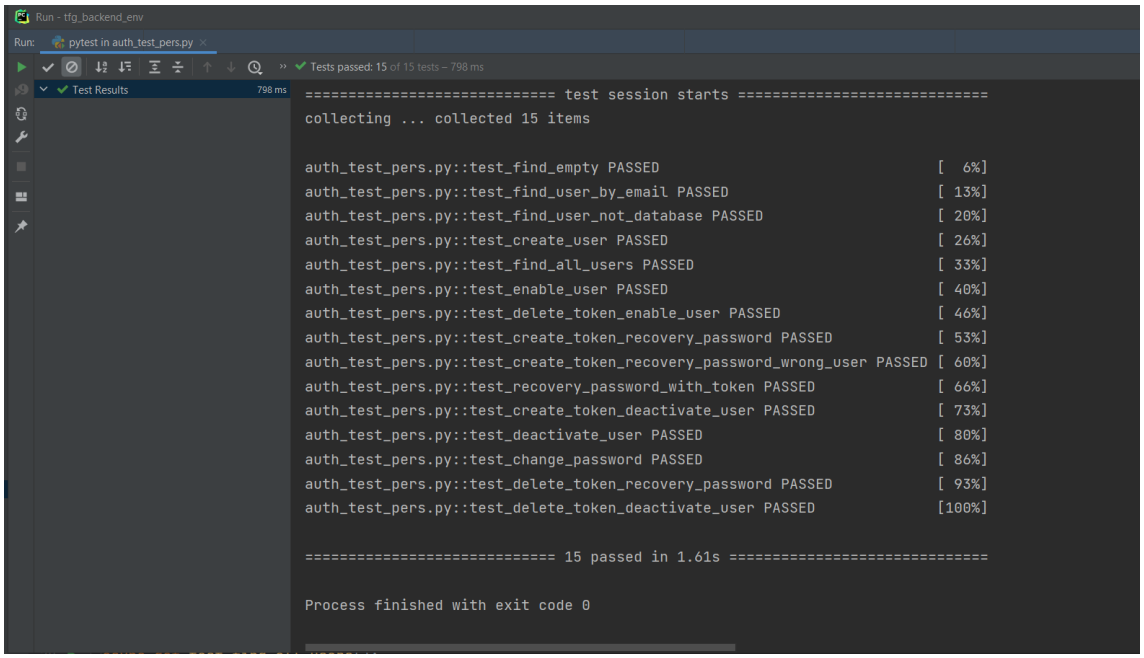

Figura 78: Pruebas unitarias - auth - capa de la persistencia

7.2.3.3. Index_test_pers.py

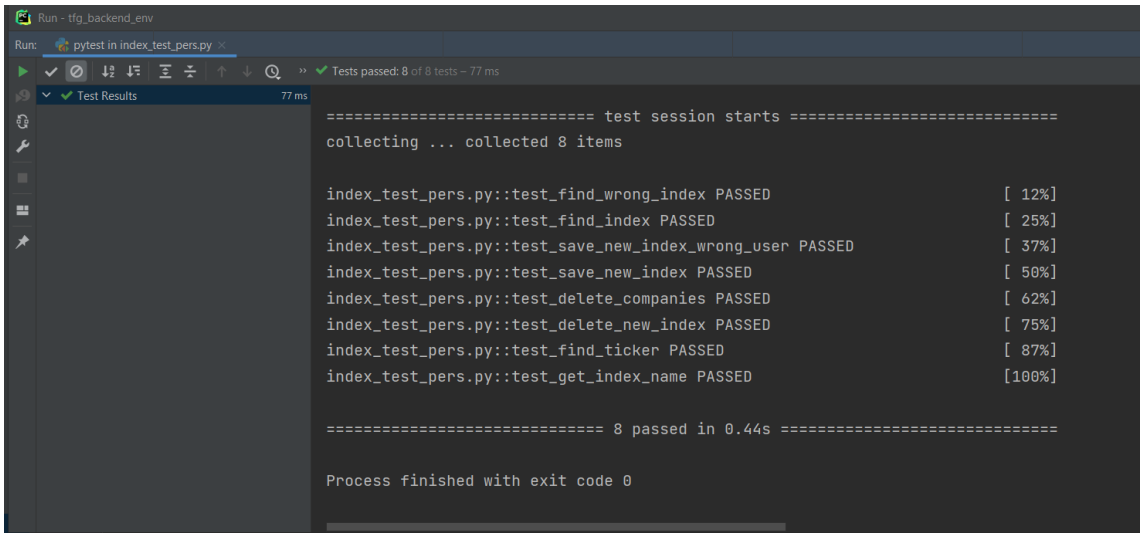

Figura 79: Pruebas unitarias - index - capa de la persistencia

7.2.3.4. News_test_pers.py

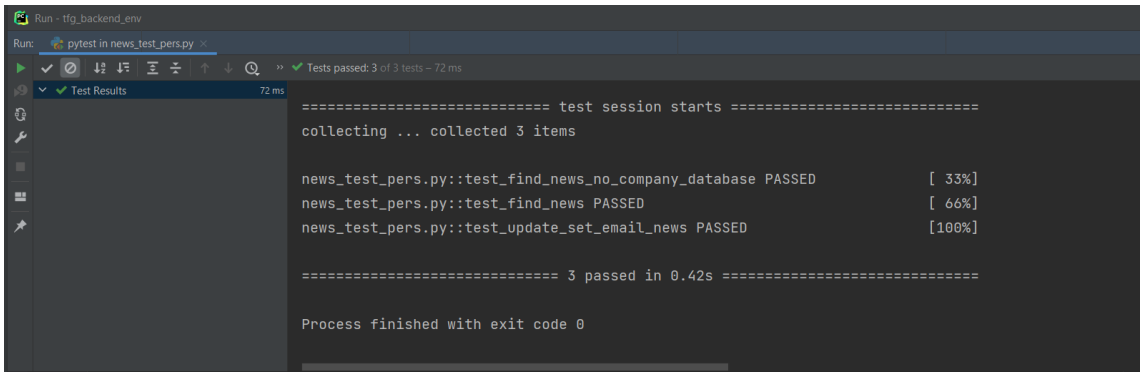

Figura 80: Pruebas unitarias - news - capa de la persistencia

7.2.3.5. Portfolio_test_pers.py

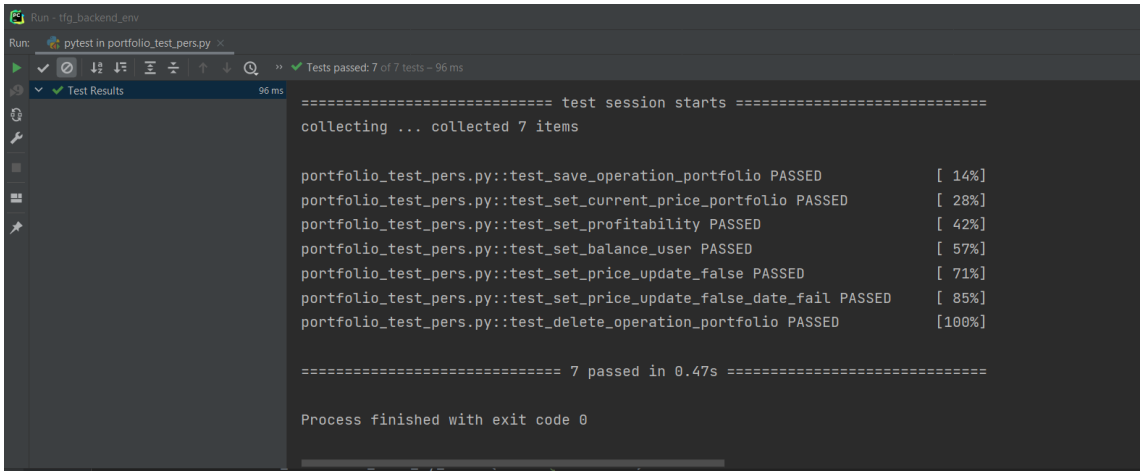

Figura 81: Pruebas unitarias - portfolio - capa de la persistencia

7.2.3.6. Record_test_pers.py

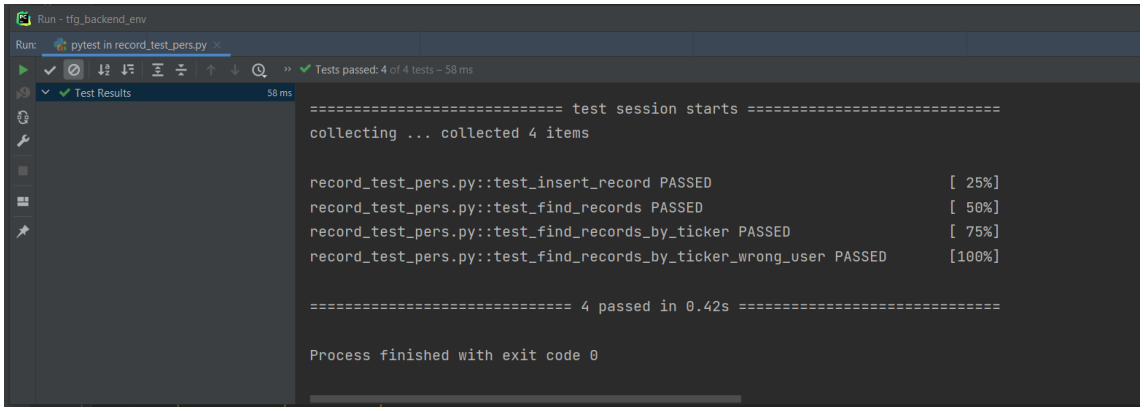

Figura 82: Pruebas unitarias - record - capa de la persistencia

7.2.4. Test Agregador

7.2.4.1. Test_persistence.py

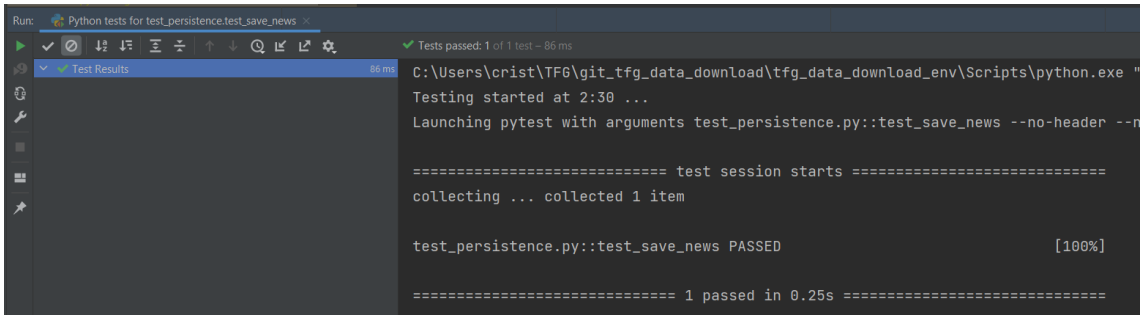

Figura 83: Pruebas unitarias - Agregador

8. Manuales del sistema

8.1. Manual de despliegue

Para montar el entorno de desarrollo de la aplicación es necesario dividir entre la base de datos, el backend y el frontend.

8.1.1. Base de datos

La base de datos MongoDB está corriendo en un servidor local mediante el uso de Docker.

Para esta labor se necesita tener instalado Docker y Docker-Compose, para ello usamos un contenedor editado por nosotros mismos y guardado en un archivo de llamado mongo.yml.

```yaml
version: '3.1'

services:

  mongo:
    image: mongo
    restart: always
    ports:
      - 27017:27017
    environment:
      MONGO_INITDB_ROOT_USERNAME: mongo
      MONGO_INITDB_ROOT_PASSWORD: mongo
    volumes:
      ~/programs/mongo/mongo-volume:/data/db
```

Figura 84: Captura del contenedor de docker mongo.yml

Una vez copiado este archivo en nuestro directorio del servidor, ejecutamos el siguiente comando para usar este contenedor y que empiece ya a funcionar:

```
docker-compose -f mongo.yml up -d
```

8.1.2. Backend

Para armar la parte del servidor necesitamos tener instalado el intérprete de Python, para este proyecto estamos usando la versión 3.8, lo cual es aconsejable no usar las versiones 3.9 o 3.10 las cuales ya han cambiado parte de la sintaxis que se ha usado en este proyecto.

Figura 85 Captura del logo de Python

Para ello ingresamos en la página oficial de Python y buscamos la versión adecuada para nosotros, cualquier versión sobre la 3.8.x.

El paso siguiente seria crear un entorno virtual en el cual podamos ejecutar el proyecto y cargar todas las librerías usadas en él. Para ello usaremos el entorno virtual "venv" que es muy simple de usar y viene de serie con el intérprete. Usaremos este comando en la consola de comandos:

```
py -3 -m venv TFG_entorno_virtual
```

Una vez creado, copiaremos la aplicación del servidor en él y desde consola activaremos el entorno virtual usando este comando:

```
Scripts\activate
```

Y usaremos el comando siguiente para instalar todas las librerías en el entorno virtual:

```
pip install -r requirements.txt
```

Una vez instaladas todas las librerías, se ingresará por consola de comandos a la carpeta de la aplicación y se ejecutará el siguiente comando para hacer correr el servidor:

```
uvicorn backend.main:app –host 0.0.0.0 –port 80
```

8.1.3. Frontend

La aplicación por la parte del cliente está usando Angular. Para hacerla funcionar se necesita instalar primero "NodeJs" desde su página oficial:

https://nodejs.org/es/

Se instalará el gestor de paquetes de node usando el siguiente comando:

```
npm install -g npm@latest
```

Se instalará Angular usando el siguiente comando:

```
npm install -g @angular/cli@latest
```

Una vez que se ha cargado nuestra aplicación de Angular, para probar que funciona, usamos el siguiente comando:

```
ng serve --open
```

Para poder desplegar nuestra aplicación web de Angular, se necesita crear los archivos que serán subidos al servidor. Para ello se usará el siguiente comando:

```
ng build
```

Se creará una nueva carpeta llamada “dist” con los ficheros compilados.

Antes de subir los ficheros al servidor, hay que tener instalado en el servidor web “Nginx”, una vez que se instale, se instalarán nuestros archivos a la carpeta:

```
/var/www/html
```

Se iniciará el servidor web de “nginx” con el comando:

```
systemctl start nginx
```

9. Conclusiones y ampliaciones

9.1. Conclusiones

Desarrollar con Python lo he encontrado muy cómodo, siempre existen diferentes formas de programar con este lenguaje. Crear el backend con FastApi ha sido un acierto.

Con los requisitos funcionales no ha habido problemas a la hora de implementarlos, pero de los no funcionales el único requisito con el que he tenido dudas y por eso el tiempo tan grande que tiene, es el de tiempo de respuesta. Al tirar de una librería de terceros también hecha en Python, cuantos más datos necesitásemos más tiempo podría tardar.

Me ha gustado mucho el hecho de trabajar con una base de datos como MongoDB, la cual es más desordenada y te da más libertad con los posibles cambios que tuvieras. Para este proyecto estaba seguro de que no tendría sentido usar una base de datos SQL y ha sido un gran acierto haber usado una NoSql como MongoDB.

Haber usado Angular para el lado del cliente también ha sido un buen acierto, me ha gustado mucho la manera de programar todas las vistas usando componentes, incluso componentes dentro de otros componentes. Y todo lo relacionado con la programación reactiva.

Estoy muy orgulloso del trabajo realizado ya que he podido aprender nuevas tecnologías además de poner en práctica todo lo aprendido durante la carrera.

9.2. Ampliaciones

Al ser un prototipo esta aplicación, se podrían desarrollar muchas ampliaciones, pero destacamos estas:

- Añadir el uso de una buena API de pago con el que tener mucha más y mejor información sobre los mercados.
- En la página principal añadiríamos un buscador de empresas de forma que el usuario pudiera hacer la búsqueda manual y se le pudiera sugerir que empresa quiere elegir.
- En la parte de información de la empresa, se podría añadir pestañas para navegar y poder desarrollar más información sobre la empresa en cuestión.

- Añadiríamos un stock screener (seleccionador de acciones con diferentes especificaciones) en una pantalla nueva, esto solo es posible con una API nueva.
- En el índice principal añadiríamos más información sobre la empresa como dividendos repartidos o en qué sector opera, etc.

10. Referencias Bibliográficas

- Aquí se muestra documentación y ejemplos del framework Fastapi.

  “Página oficial de Fastapi”

  https://fastapi.tiangolo.com/

- Aquí se muestra documentación y ejemplos del framework Angular.

  “Página oficial de Angular”

  https://angular.io/docs

- Aquí se muestra documentación y ejemplos de MongoDB.

  “Página oficial de la base de datos MongoDB”

  https://www.mongodb.com/docs/

- Aquí se muestran ejemplos usados en la aplicación.

  “Página oficial de la librería Angular Material”

  https://material.angular.io/guide/getting-started

- Aquí se muestran ejemplos que se han usado para el menú.

  “Página oficial de Bootstrap”

  https://getbootstrap.com/docs/5.2/getting-started/introduction/

- Aquí se muestra la documentación del controlador usado con MongoDB.

  “Página oficial de Motor”

  https://motor.readthedocs.io/en/stable/

11. Anexos

11.1. Contenido entregado

11.1.1. Aplicación desplegada

Se ha desplegado la aplicación a través de esta dirección url:

http://agpa.duckdns.org

Se han habilitado tres usuarios:

Email: invitado1@uno.es - contraseña: uno

Email: invitado2@dos.es - contraseña: dos

Email: admin@admin.es – contraseña: admin

También se puede registrar nuevos usuarios en la aplicación, usando direcciones de correo válidas y revisando el nuevo correo en la carpeta Spam.

11.1.2. Backend

La aplicación por parte del framework Fastapi tiene esta estructura de carpetas:

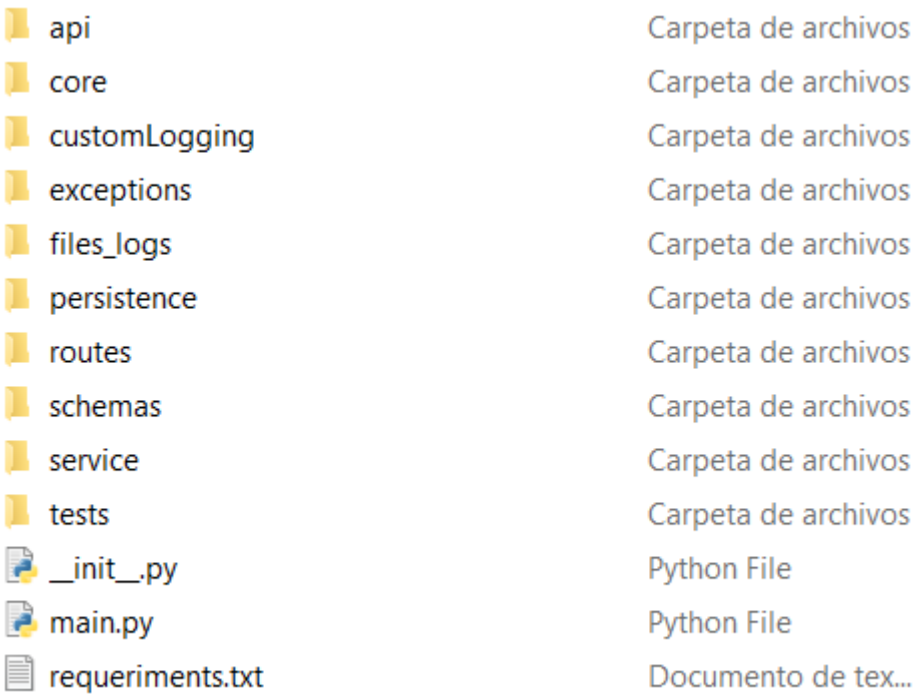

Figura 86: Arquitectura de capetas importantes del backend

Api: Contiene los ficheros relacionados con la descarga de precios.

Core: Contiene los ficheros relacionados con la configuración de la aplicación.

CustomLogging: Contiene el fichero de configuración para crear los logs.

Exceptions: Contiene las excepciones creadas por el usuario en la aplicación.

Files_logs: Contiene los logs de la aplicación.

Persistence: Contiene los ficheros relacionados con la capa de la persistencia.

Routes: Contiene los ficheros relacionados con la capa de los endpoints.

Schemas: Contiene los ficheros de los modelos de Pydantic para controlar la información que entra y sale de la aplicación.

Service: Contiene los ficheros relacionados con la capa de servicio.

Tests: Contiene todos los test unitarios.

Requeriments.txt: Contiene los nombres y versión de todas las librerías usadas en la aplicación por parte del servidor.

11.1.3. Frontend

La aplicación por parte del framework Angular tiene esta arquitectura de carpetas:

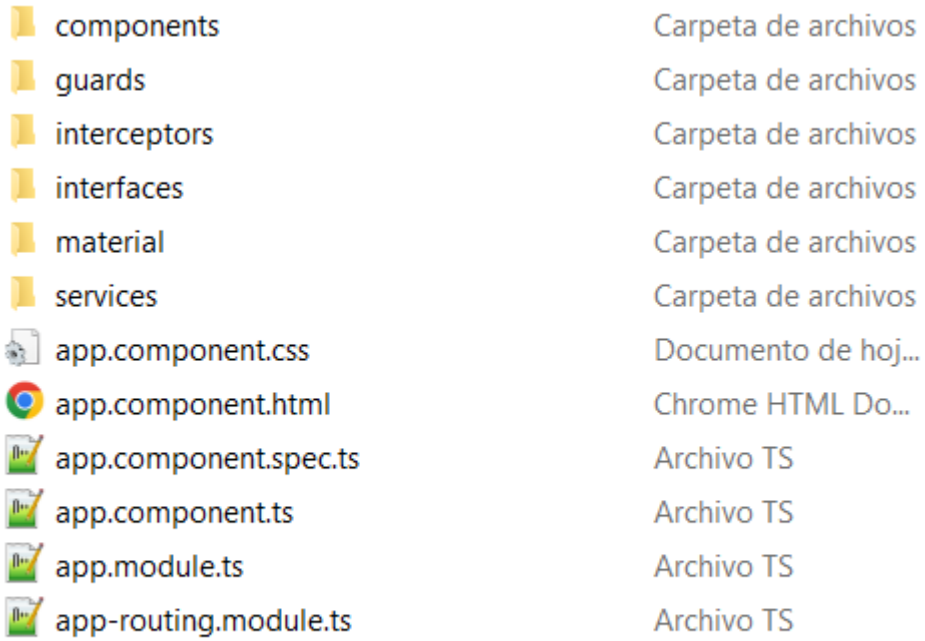

Figura 87: Arquitectura de carpetas importantes del frontend

Components: Contiene todos los ficheros relacionados con los componentes.

Guards: Contiene los ficheros personalizados que controlan los accesos a las distintas rutas mediante los roles.

Interceptors: Contiene el fichero que controla el acceso a las rutas mediante tokens.

Interfaces: Contiene los ficheros con los que se describen las estructuras de los datos.

Material: Contiene los ficheros usados por la librería de Angular-Material.

Services: Contiene los ficheros con los diferentes métodos que extraen o mandan información al servidor.

App-routing.module.ts: Contiene todas las rutas usadas por el cliente.

11.1.4. Agregador

La aplicación por parte del agregador de noticias tiene esta arquitectura de carpetas:

Figura 88: Arquitectura de carpetas del agregador de noticias

Core: Contiene los ficheros de configuración de la aplicación de descarga.

Extractions: Contiene los ficheros con los métodos de extracción de noticias.

Persistence: Contiene los ficheros relacionados con la base de datos.

Service: Contiene los ficheros con la lógica del agregador.

Tests: Contiene los ficheros con los tests unitarios.

Newspapers.json: Las direcciones rss desde donde se bajan las noticias.

11.1.5. Código fuente

A continuación, se va a mostrar el código de algunos métodos para ir viendo como ha sido la implementación de una aplicación asíncrona en la parte del backend.

Record_route.py: es la capa mas externa de la aplicación, es con la que interactúa el cliente. A partir de los parámetros de los métodos se usa la inyección de dependencia para comprobar mediante el token JWT que el usuario existe en la aplicación y por tanto tiene acceso a ese método.

En la cabecera del método, el "response model" viene gestionado por la librería Pydantic, que viene integrada en Fastapi y especifica el contenido que sale de la aplicación.

11.1.6. Record_route.py

```
from fastapi import APIRouter, status, HTTPException
from fastapi_project.service.record_service import RecordService
from fastapi_project.service.impl.record_service_impl \
    import RecordServiceImpl
from fastapi_project.exceptions.rest_exception import *
from fastapi_project.schemas.index_schema import *
from fastapi_project.routes.routes_resources.aux_routes_methods \
    import *
from typing import List

router = APIRouter()


@router.get("/get-records", response_model=List[Data_of_portfolio],
            status_code=200)
async def get_records(current_user: User = Depends(get_current_user)):

    record_service: RecordService = RecordServiceImpl()
    try:
        return await
record_service.get_records_serv(current_user.email)

    except MongoRestError:
        raise
HTTPException(status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
                            detail='Internal error')

@router.get("/get-records-by-ticker/{ticker}",
            response_model=List[Data_of_portfolio], status_code=200)
async def get_records_by_ticker(ticker: str,
                                current_user: User =
Depends(get_current_user)):

    record_service: RecordService = RecordServiceImpl()
    try:
        return await record_service.\
            get_records_by_ticker_serv(current_user.email, ticker)

    except InvalidTickerRestError:
        raise
```

```
HTTPException(status_code=status.HTTP_422_UNPROCESSABLE_ENTITY,
                            detail=f'{ticker} does not valid')
    except MongoRestError:
        raise
HTTPException(status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
                            detail='Internal error')
```

Record_service.py: es la capa del dominio, aquí como en todas las capas, los métodos llevan las palabras reservadas "async/await" para ejecutar las corrutinas.

11.1.7. Record_service.py

```
from fastapi_project.persistence.record_gateway import \
    AbsRecordGateway
from fastapi_project.persistence.impl.record_persistence_impl \
    import RecordGatewayImpl
from fastapi_project.persistence.auth_gateway import \
    AbsAuthGateway
from fastapi_project.persistence.impl.auth_persistence_impl \
    import AuthGatewayImpl
from fastapi_project.exceptions.service_exception import *
from fastapi_project.exceptions.rest_exception import *
from fastapi_project.customLogging.class_logging \
    import custom_logging
from ..service_resources.checks_service_methods import *
from ..record_service import RecordService
from typing import List

class RecordServiceImpl(RecordService):

    def __init__(self):
        self.exe_record: AbsRecordGateway = RecordGatewayImpl()
        self.exe_auth: AbsAuthGateway = AuthGatewayImpl()

    async def get_records_serv(self, email: str) -> List[dict]:

        try:
            check_email(email)
            user = await self.exe_auth.find_user_by_email(email)
            if not user:
                raise NotFoundServiceError(f'{email} does not exist')
            records = await self.exe_record.find_records(email)
            return records

        except MongoServiceError as err:
            custom_logging.logger.error(repr(err))
            raise MongoRestError(err)
        except NotFoundServiceError as err:
            raise NotFoundRestError(err)
        except InvalidEmailServiceError as err:
            raise InvalidEmailRestError(err)


    async def get_records_by_ticker_serv(self, email: str, ticker:
str)\
            -> List[dict]:

        aux_ticker = ticker.upper()
        try:
```

```
            check_email(email)
            check_ticker(ticker)
            user = await self.exe_auth.find_user_by_email(email)
            if not user:
                raise NotFoundServiceError(f'{email} does not exist')
            records = await self.exe_record.\
                find_records_by_ticker(email, aux_ticker)
            return records

        except MongoServiceError as err:
            custom_logging.logger.error(repr(err))
            raise MongoRestError(err)
        except NotFoundServiceError as err:
            raise NotFoundRestError(err)
        except InvalidEmailServiceError as err:
            raise InvalidEmailRestError(err)
        except InvalidTickerServiceError as err:
            raise InvalidTickerRestError(err)
```

Record_persistence.py: es la capa de la persistencia, también aquí se usan las corrutinas para hacer las peticiones a la base de datos. Al usar un controlador como Motor nos obliga a cambiar un poco la forma de hacer consultas para permitir peticiones asíncronas.

11.1.8. Record_persistence.py

```
from ..record_gateway import AbsRecordGateway
from fastapi_project.core.mongodb import db
from fastapi_project.exceptions.persistence_exception import *
from pymongo import errors
from typing import List

class RecordGatewayImpl(AbsRecordGateway):

    async def find_records(self, email: str):

        try:
            output = []
            collection = db.get_collection('record')
            async for record in collection.\
                    find({'email': email}):
                output.append(record)
            return output

        except errors.PyMongoError as err:
            raise MongoServiceError(err)

    async def find_records_by_ticker(self, email: str,
                                     ticker: str):

        try:
            output = []
            collection = db.get_collection('record')
            async for record in collection.\
                    find({'email': email, 'ticker': ticker}):
                output.append(record)
            return output
```

```
        except errors.PyMongoError as err:
            raise MongoServiceError(err)

    async def insert_record(self, data: dict):

        try:
            collection = db.get_collection('record')
            _id = await collection.insert_one(data)
            return str(_id.inserted_id)

        except errors.PyMongoError as err:
            raise MongoServiceError(err)
```