Aplicación para análisis de inversión

Application for investment analysis

Trabajo de Fin de Grado
Curso 2019–2020

Autor
Alejandro Povedano Atienza

Director
Adrián Riesco Rodríguez

Grado en Ingeniería Informática

Facultad de Informática

Universidad Complutense de Madrid

Índice

Índice de Figuras

Agradecimientos

Antes de comenzar este trabajo, me gustaría dedicar unas palabras a aquellas personas que me han servido de apoyo y de inspiración en la realización de este presente trabajo.

Quisiera dar las gracias a mis padres y a mi hermana, porque sin el apoyo que me han dado estos años de carrera, no podría permitirme escribir estas líneas.

En segundo lugar, me gustaría agradecer a mi tutor Adrián Riesco Rodríguez la dedicación en este proyecto y por ayudarme a alcanzar los objetivos que me había propuesto. Siempre me has proporcionado los recursos necesarios y consejo cuando lo necesitaba.

Por último, pero no por ello menos importante, gracias a mis amigos y a mis compañeros de trabajo por el apoyo moral que me han dado durante la realización del proyecto. En numerosas ocasiones pensaba que la envergadura de este trabajo era demasiado grande para el tiempo del que disponía, pero sus palabras me sirvieron para superar los obstáculos del camino.

A todos ellos, muchas gracias.

Resumen

La volatilidad en el mercado de valores es definida como un índice que informa sobre las variaciones y movimientos que puede sufrir el valor de los activos cotizados en bolsa. Este índice posee gran importancia dentro del mercado financiero, pues es el que determina el riesgo de una inversión económica. Cuanta más alta sea la volatilidad, mayor será la oscilación de valor de las acciones y, por lo tanto, será más arriesgado que la inversión realizada sea beneficiosa.

Los inversores buscan la mejor estrategia a la hora de invertir para aumentar el número de activos en su posesión. Es por esto que atienden a la volatilidad para encontrar la mayor rentabilidad de una compra o venta de acciones con el menor riesgo posible dado un periodo de tiempo.

Por esta razón se creó este proyecto: para proporcionar a los inversores una herramienta que obtenga en tiempo real las variaciones del valor las acciones de las empresas y poder plantearse o no, realizar una inversión. Para que esta herramienta estuviera al alcance de la mayor de usuarios posible, se decidió crear la aplicación en un entorno web. De esta manera, se puede acceder al contenido y hacer uso de sus servicios siempre y cuando se disponga de un ordenador o un dispositivo móvil con navegador y acceso a Internet (multiplataforma). La aplicación permite la visualización de datos financieros e históricos de las acciones de empresas representados en diferentes formatos de gráficos. Además, cada usuario registrado dentro de la web cuenta con un área privada, donde puede visualizar de manera rápida las acciones de las empresas que le interesan, así como los beneficios o pérdidas que se han registrado en el momento de la adquisición de los activos financieros.

Palabras clave

Aplicación web, datos financieros, volatilidad, gráficas, accesibilidad, Servicios web, acciones de empresas.

Abstract

Volatility in the stock market is defined as an index that reports on the variations and movements that the value of listed assets may undergo. This index has great importance within the financial market, because it determines the risk of an economic investment. The higher the volatility, the greater the fluctuation in the value of the shares and, therefore, the riskier the investment made will be beneficial.

Investors look for the best investment strategy to increase the number of assets in their possession. This is why they are keeping an eye on volatility to find the highest levels of investment profitability on a purchase or sale of shares with the least possible risk given a period of time.

This is the reason why this project was created: to provide investors with a tool to obtain real-time changes in the value of companies' shares and to be able to consider whether or not to make an investment. In order to make this tool available to as many users as possible, it was decided to create this application in a web environment. In this way, you can access content and use its services as long as you have a computer or a mobile device with a browser and Internet access (multiplatform). The application allows the visualization of financial and historical data of the shares of companies represented in different graphic formats. In addition, each user registered on the web has a private area, where they can quickly view the actions of the companies that interest them, as well as the benefits or losses that were recorded at the time of acquisition of the financial assets.

Keywords

Web application, financial data, volatility, graphs, accessibility, web services, company stocks

Sección 1

Introducción

En esta sección se expondrá la motivación de la realización de este presente trabajo, así como los objetivos que se pretenden cumplir. Además, se listarán las secciones que componen esta memoria, junto con una breve descripción de cada una de ellas.

1.1. Motivación

La volatilidad en el mercado de valores requiere de una continua supervisión de la evolución de los activos financieros que compongan la cartera que mantenga un particular o una empresa. Esta supervisión tiene una doble finalidad: valorar el progreso de los valores en los que actualmente se haya invertido y la expectativa de inversión de futuro en otros activos. Dada la volatilidad variable de los activos cotizados que generalmente caracteriza a la bolsa y especialmente dada la inestabilidad económica actual derivada de la crisis sanitaria, que indudablemente afecta también a los valores cotizados en bolsa, se requiere de herramientas informáticas que proporcionen en tiempo real las variaciones de los índices bursátiles.

Por lo tanto, la motivación de este proyecto es precisamente el desarrollo de un entorno que permita la visualización y el seguimiento de los valores de diferentes acciones, como a continuación más ampliamente se expondrá.

1.2. Objetivo

El objetivo principal de este trabajo consiste en el desarrollo de una aplicación web que permita visualizar los datos financieros e históricos de las acciones de empresas en diferentes formatos de gráficos. Dicha visualización contará con diferentes opciones y filtros que podrán ser aplicados sobre la información representada, que permitirán al usuario consultar los datos deseados de una manera rápida y eficaz.

Esta aplicación también tiene como objetivo proporcionar a los usuarios registrados, un registro de las acciones adquiridas, permitiéndoles observar la evolución del valor de dichas acciones.

1.3. Estructura del documento

Con el fin de que la memoria tuviera una estructura y organización coherentes, se ha dividido los puntos fundamentales identificados durante el desarrollo del proyecto en secciones. A continuación, se listan y se resumen el contenido de dichas secciones:

- **Sección 1:** en esta sección redactada tanto en inglés como en español, se describen puntos tales como la motivación y los objetivos que se desean cumplir a través del desarrollo y la implementación de la aplicación. También se describe la estructura que va a seguir el presente documento, dividido en secciones y subsecciones.

- **Sección 2:** en esta sección se listan y se detallan las tecnologías que han sido utilizadas para llevar a cabo el desarrollo. Se hablará sobre sus orígenes, así como sus principales características. Se mencionará las herramientas que han sido empleadas para el uso de cada tecnología, junto con una explicación de los servicios web, haciendo especial hincapié en los servicios REST, los cuales son el núcleo principal de la funcionalidad de la aplicación web.

- **Sección 3:** en esta sección se expondrá las iteraciones necesarias para conseguir un diseño coherente y funcional que cumpla con los objetivos propuestos. Estas iteraciones están acompañadas de plantillas mockups, que sirven para tener un diseño preliminar de la aplicación que se desea construir. Además, se exponen los Estándares Web W3C y los niveles de accesibilidad web que han sido tenidos en cuenta en la creación de estos diseños.

- **Sección 4:** en esta sección se describe la arquitectura de la aplicación, separada claramente en dos partes: *Front-End* y *Back-End*. Se realizará un recorrido por las diferentes pantallas de la aplicación, explicando su funcionalidad y propósito. Por último, se describe los servicios que se emplean para realizar la funcionalidad principal desde un punto de vista técnico.

- **Sección 5:** en esta sección redactada tanto en inglés como en español, se exponen las conclusiones que se han presentado tras haber terminado el desarrollo de la aplicación, así como los puntos identificados durante este desarrollo que permiten mejorar y escalar la aplicación. A continuación, se adjuntan las direcciones de los repositorios correspondiente a la parte Front y Back alojados en la plataforma GitHub:

  - https://github.com/Alepoved/TFG_FRONT
  - https://github.com/Alepoved/TFG_BACK

Cada repositorio contiene instrucciones de como ejecutar la aplicación y consideraciones técnicas básicas de cada proyecto. El uso del código está bajo licencia de código abierto por lo que su uso, distribución y modificación son públicos.

Section 1

Introduction

This section will explain the motivation for carrying out this work, as well as the objectives that are intended to be accomplished. Furthermore, the sections that make up this paper will be listed, along with a brief description of each one.

1.1. Motivation

The volatility in the stock market requires continuous monitoring of the evolution of the financial assets that make up the portfolio held by an individual or a company. This supervision has a dual purpose: to assess the progress of the assets in which he/she has currently invested and the expectation of future investment in other assets. Given the variable volatility of listed assets that generally characterizes the stock market and especially given the current economic instability derived from the health crisis, which undoubtedly also affects listed stocks, computer tools are required to provide variations in real time of the stock indices.

Therefore, the motivation of this project is precisely the development of an environment that allows the visualization and monitoring of the values of different actions, as will be explained more fully below.

1.2. Objective

This degree thesis' main objective is the development of a web application that allows users to visualize the financial data and historical companies' stocks values using different types of charts. Options and filters will be applied to the graph's visualization, allowing the user to see the desired data rapidly and effectively.

This application also aims to provide registered users with a record of the shares acquired, allowing them to observe the evolution of the value of said shares.

1.3. Document structure

In order for the memory to have a coherent structure and organization, the fundamental points identified during the development of the project have been divided into sections. The contents of these sections are listed and summarized below:

- **Section 1:** this section written in English and Spanish, describes points such as the motivation and the objectives that you want to achieve through the development and implementation of the application. The structure that this document will follow is also described, divided into sections and subsections.

- **Section 2:** this section lists and details the technologies that have been used to carry out the development. Its origins will be discussed, as well as its main characteristics. The tools that have been used for the use of each technology will be mentioned, along with an explanation of web services, with special emphasis on REST services, which are the main core of the functionality of the web application.

- **Section 3:** this section will expose the iterations necessary to achieve a coherent and functional design that meets the proposed objectives. These iterations are accompanied by mockup templates, which serve to have a preliminary design of the application to be built. In addition, the W3C Web Standards and the levels of web accessibility that have been taken into account in the creation of these designs are exposed.

- **Section 4:** this section describes the architecture of the application, clearly separated into two parts: Front-End and Back-End. A tour of the different screens of the application will be made, explaining its functionality and purpose. Finally, the services that are used to perform the main functionality from a technical point of view are described.

- **Section 5:** this section presents the conclusions that have been presented after completing the development of the application, as well as the points identified during this development that allow the application to be improved and scaled. The addresses of the repositories corresponding to the Front and Back part hosted on the GitHub platform are attached below:

  - https://github.com/Alepoved/TFG_FRONT
  - https://github.com/Alepoved/TFG_BACK

Each repository has instructions on how to run the application and basic technical consideration for each project. The use of the code is under an open source license, so its use, distribution and modification are public.

Sección 2

Tecnologías utilizadas

En esta sección se nombrarán y se detallarán las tecnologías que han sido empleadas para el desarrollo y la implementación de este trabajo. Se comenzará por listar las tecnologías, exponiendo sus principales características, así como sus orígenes, arquitectura y evolución de las mismas.

2.1. Angular

Angular (comúnmente llamado *Angular 2+* o *Angular 2*) es un *framework* para aplicaciones web desarrollado en *TypeScript*, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de *Modelo-Vista-Controlador (MVC)*, en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

La primera versión de este *framework* fue lanzada el 14 de septiembre de 2016 y su última versión estable es la 8.2.11, siendo publicada el 15 de octubre de 2019.

El desarrollo e implementación del código en *Angular* se realiza mediante el uso de *TypeScript*: un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases. Se define superconjunto como un lenguaje escrito y, por lo tanto, modificado sobre otro lenguaje. *TypeScript* está basado y claramente influenciado por lenguajes de programación como *Java, C++* y *Javascript*. Esto permite identificar y obtener las características más beneficiosas sobre estos lenguajes y migrarlas a un lenguaje único que pueda explotar dichas características.

Una de las características más relevantes sobre *Angular* es que se basa en la creación de componentes. Estos componentes una vez creados, permiten ser declarados y reutilizados, ya que cada componente debería funcionar de manera aislada. Un componente puede llamar a otros componentes y así hacer uso de sus funcionalidades, por lo que sus propiedades pueden ser usadas para el enlace o compartición de datos.

En un componente básico de *Angular*, se identifican la siguiente estructura:

- **Template:** Permite definir la vista de un componente. La vista está construida usando el lenguaje HTML, permitiendo que se puedan agregar expresiones propias de *Angular* para simplificar y agregar lógica al código HTML. Un ejemplo claro de estas expresiones sería la inclusión de condiciones del tipo "if-else" en la renderización de un fragmento de código HTML.

El template también permite invocar a otros componentes previamente creados e interactuar con ellos, por ejemplo, el paso de datos de un componente a otro.

- **Clase:** permite añadir metadatos a un componente para poder declarar e identificar el componente. Estos metadatos permiten relacionar y enlazar el template, la clase y los estilos, que son las tres partes básicas por las que se define un componente en *Angular*.

La clase permite crear funciones que contendrán toda la lógica que puede desarrollar un componente. Se pueden crear decoradores de tipo *@Input* y *@Output* para poder declarar variables de entrada y salida de datos.

- **Hoja de estilos:** permite almacenar el diseño y el formato de representación del contenido. Esto estilos se aplican al template para poder modificar el color, tamaño, fuentes, bordes, posición, etc.

Existe una referencia en la clase del componente a esta hoja de estilos para que el formato pueda ser aplicado al código HTML situado en la template.

Angular destaca por la carga rápida debido a su enrutador de componentes, permitiendo la creación de vistas de interfaz de manera simple y potente. Además, se caracteriza por el uso de una herramienta llamada Angular CLI, que se basa en el uso de comandos que permite crear, modificar y eliminar componentes.

2.1.1. Orígenes de Angular

Existe una confusión entre Angular y AngularJS debido a que los nombres son muy similares. AngularJS es un *framework* de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página (Single-Page Applications), mientras que Angular es un *framework* de TypeScript reescrito desde cero por el equipo que desarrolló AngularJS.

Cuando se desarrolló AngularJS (Angular 1.x), no se pensó en el soporte para móviles. Se diseñó para crear apps Web de tipo SPA, con enlazado a datos bidireccional. Por el contrario, Angular si está pensado para un desarrollo orientado a móviles. Existen números recursos, como por ejemplo bibliotecas del tipo NativeScript (ver Figura 2.1), que lo que permiten es facilitar y proveer un desarrollo mucho más ágil y sencillo para móviles.

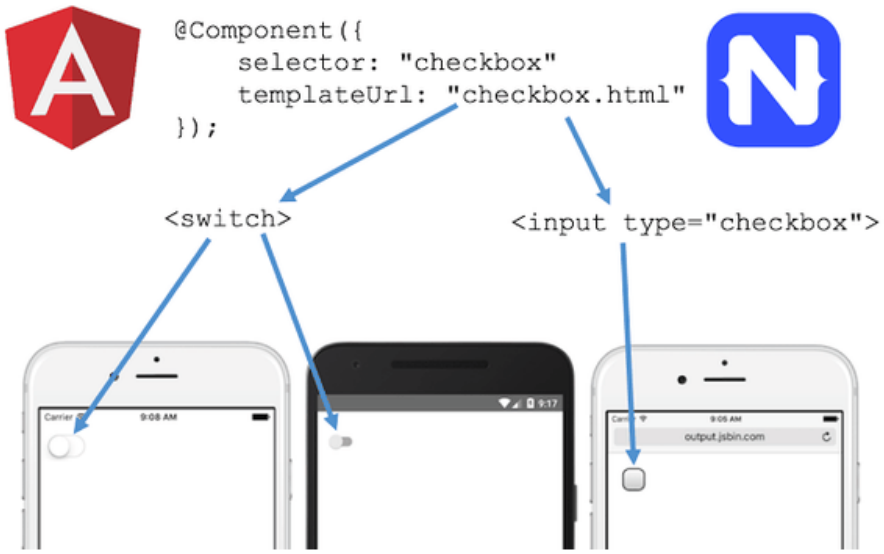


Figura 2.1: Ejemplo de uso de la biblioteca NativeScript

El cambio de JavaScript ofrecido por AngularJS a TypeScript ECMAScript 6 (ES6), TypeScript ES5 o Dart ofrecido por Angular, hace que esta última opción sea más apropiada para la creación de un proyecto robusto y eficiente. Otra de las razones por las que Angular es mejor candidato para proyectos medianos o grandes, es debido a su estructura de componentes y plantillas, que permite mantener estos proyectos de manera sencilla gracias a su organización basada en la modularidad.

2.1.2. Arquitectura de Angular

La arquitectura de Angular se basa en el uso de módulos denominados NgModules. Estos módulos contienen un conjunto de componentes y/o servicios de una aplicación (ver Figura 2.2). La organización de estos responde a una estructura de árbol: existe un módulo raíz del que parten otros módulos hijos, enlazándose así unos módulos con otros.

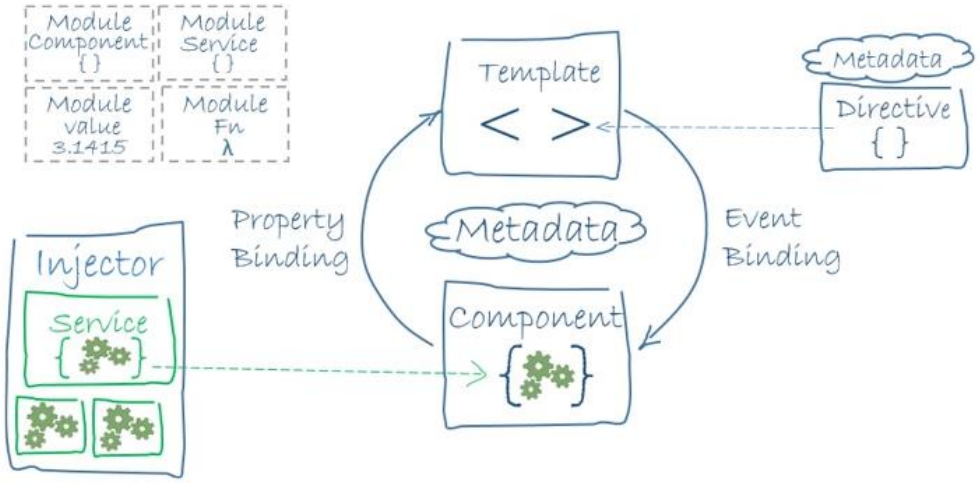


Figura 2.2: Esquema de arquitectura Angular.
Fuente https://sg.com.mx/revista/56/angular

Existen 3 tipos principales de módulos dentro de una aplicación basada en Angular:

- **Módulos de páginas:** los componentes creados en Angular tienen una visibilidad definida. Cuando se genera el componente mediante el uso de la herramienta Angular CLI, se declara en un módulo contenedor y esto permite que dicho componente puede ser usado dentro de este módulo, debido a su visibilidad dentro de este (la declaración se recoge en la propiedad *declarations:[ ]*). Si se desea usar fuera de este módulo en concreto, deberá ser exportado mediante la propiedad *exports:[ ]*. Por lo tanto, estos módulos permiten organizar el contenido de la aplicación.

- **Módulos de componentes:** estos módulos contienen una serie de componentes que pueden ser reutilizados en otras partes de la aplicación o incluso en otros proyectos, mediante las propiedades de importación y exportación. Los componentes en Angular se caracterizan por ser capaces de comunicarse con el componente o *template* padre que los está conteniendo (ver Figura 2.3).

- **Módulos de servicios:** se trata de módulos que contienen los servicios necesarios para el funcionamiento correcto de la aplicación. La mayoría de estos servicios son globales, es decir, los componentes pueden realizar uso de ellos siempre que se importe en su declaración el servicio que se desea consumir. Además, estos módulos contenedores de servicios se cargan una sola vez en el AppModule, debido a su alcance global. Se define AppModule como el módulo raíz que contiene la declaración del reto de módulos. Además, proporciona el mecanismo necesario para que la aplicación arranque.

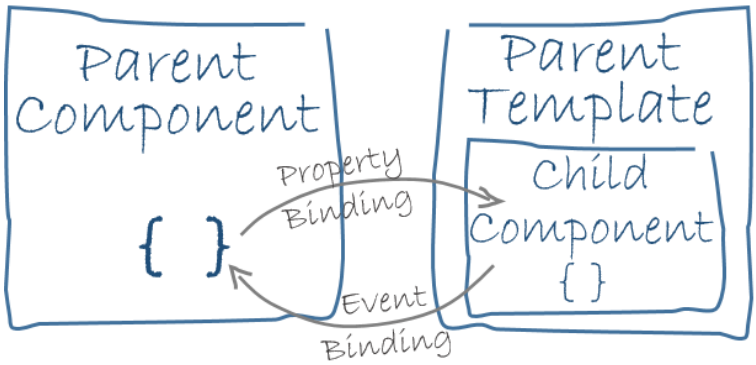


Figura 2.3: Esquema de relación entre componentes en Angular. Fuente https://v2.angular.io/docs/ts/latest/guide/architecture.html

2.2. Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft y tiene soporte para las plataformas de Windows, Linux y MacOS. Fue lanzado el 29 de abril de 2015 y al tratarse de un software gratuito y de código abierto, su código fuente fue público en la plataforma de GitHub el 18 de noviembre de 2015.

Esta herramienta posee las características básicas que podemos esperar de un editor de texto: colores dependiendo de la sintaxis, marcas de errores y advertencias, números de línea, etc. Sin embargo, el verdadero motivo por el cual esta herramienta destaca sobre otras es por su sistema de extensiones. Permite agregar extensiones que ayudan a facilitar el trabajo dependiendo del lenguaje en el que se esté desarrollando. Un ejemplo de extensiones son IntelliSense, que permite identificar y autocompletar código de manera inteligente (funciones, variables, etc) o Built-in-Git, que permite una integración con la plataforma de GitHub para poder subir el código a los repositorios.

2.3. Servicios Web

2.3.1. Servicios Web HTTP

Los servicios web son una tecnología que utiliza un conjunto de protocolos y estándares que sirven para poder intercambiar información entre aplicaciones. Esta tecnología permite realizar el intercambio de datos independientemente de la plataforma o del lenguaje de programación empleados para el envío o recepción de estos datos. Por lo tanto, un servicio web se podría definir como un canal de comunicación e interoperabilidad entre máquinas conectadas en una red.

En todo servicio web existen 3 componentes fundamentales: un proveedor del servicio web, un cliente que solicita el consumo de este servicio web ofrecido por el proveedor y el publicador.

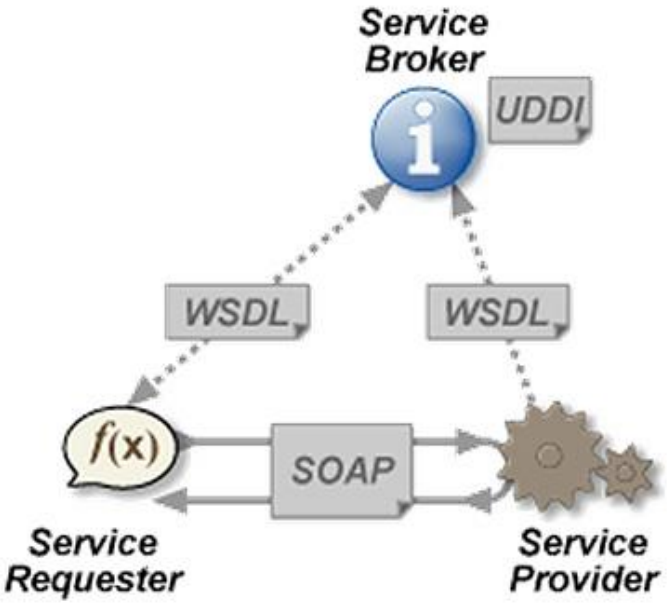


Figura 2.4: Esquema de arquitectura de Servicios Web SOAP.
Fuente: https://es.wikipedia.org/wiki/Servicio_web

El flujo de interacción entre estas 3 partes es la siguiente: el cliente desea realizar una operación y para ello tiene que comunicarse con el publicador que es quien descubre e identifica quién es el proveedor del servicio. Una vez identificado se recoge la petición de la operación que el cliente quiere realizar. El proveedor genera una respuesta correspondiente a la petición del cliente (solicitud de información enviando a su vez datos relevantes para que esta solicitud pueda ser procesada) y esta respuesta es enviada al cliente. Si no se ha producido ningún error a la hora de procesar y generar la respuesta, esta contiene los datos solicitados por el cliente inicialmente.

Cuando se devuelve una respuesta a un cliente que ha realizado una petición, se adjunta un código a esta respuesta. Este código forma parte de un diccionario de códigos HTTP que permiten identificar el estado de esa respuesta que se está devolviendo al cliente.

La nomenclatura que utiliza el diccionario de códigos de respuesta sigue un estándar globalmente aceptado: el primer digito del código de respuesta permite identificar la clase de respuesta. Existen 5 clases de respuesta: respuestas informativas, peticiones correctas, redirecciones, errores de cliente y errores de servidor.

Debido a que la comunicación entre máquinas cuyas características pueden diferir en gran medida debido a sus especificaciones técnicas, los lenguajes de programación o bases de datos empleados o incluso sus sistemas operativos, se ha optado por emplear un lenguaje escrito en archivos de texto plano para el intercambio de datos entre estas máquinas. Los lenguajes más populares y usados para este intercambio de datos son XML y JSON.

XML (Lenguaje de Marcado Extensible), está basado en el uso de etiquetas. Es un metalenguaje que proviene del lenguaje SGML. Se caracteriza por tener un formato estructurado y fácil de comprender. Al presentar este tipo de formato, es muy sencillo realizar validaciones de sintaxis y además permite definir estructuras de datos complejas y reutilizables.

JSON (Notación de Objeto de Javascript), es un subconjunto de la notación literal de objetos de JavaScript. Permite el uso de los siguientes tipos de datos: números, cadenas, booleanos, nulos, arrays y objetos. La principal ventaja que ofrece JSON es su alta velocidad de procesamiento de los datos. Sin embargo, su sintaxis puede llevar a confusión y a una mala interpretación de los datos a simple vista por parte del desarrollador.

Existen diferentes protocolos que permiten hacer uso de los servicios web. Uno de los primeros fue SOAP (Simple Object Access Protocol) que define que la manera que tienen de comunicarse dos máquinas es mediante el uso de XML. Posteriormente apareció el protocolo REST, el cual se caracteriza porque la comunicación entre maquinas se realiza usando el propio protocolo HTTP.

2.3.2. Servicios Web REST

REST (La transferencia de estado representacional) es un estilo de arquitectura que permite conectar maquinas entre si basándose en el protocolo de comunicación HTTP. Este concepto surgió en el año 2000, cuando Roy Fielding, uno de los autores que colaboró en la especificación del protocolo HTTP, presentó REST en una tesis doctoral sobre la web (Architectural Styles and the Design of Network-based Software Architectures).

A partir de ese momento, REST revolucionó la ingeniería del software ya que supuso un cambio relevante en el desarrollo de servicios de aplicaciones, permitiendo la comunicación entre un cliente y un servidor para poder obtener, procesar, generar o enviar datos en todos los formatos disponibles, como, por ejemplo, JSON o XML.

Las peticiones del cliente son captadas por el servidor y mediante el uso de servicios se generan y envían los datos requeridos en de petición al cliente (ver Figura 2.5). Estos servicios son ajenos al cliente: las peticiones se registran y se procesan en el servidor y todas las peticiones son independientes del resto. Esto quiere decir que REST se trata de un protocolo cliente/servidor sin estado, es decir, la petición se puede ejecutar sin necesidad de recordar el historial de peticiones anteriores. La mayor parte de las empresas como Google, Netflix, Twitter, Facebook, LinkedIn usan REST. Parte de sus servicios son públicos y otros desarrolladores pueden consumirlo e integrarlo en otros desarrollos, como, por ejemplo, los mapas de Google.

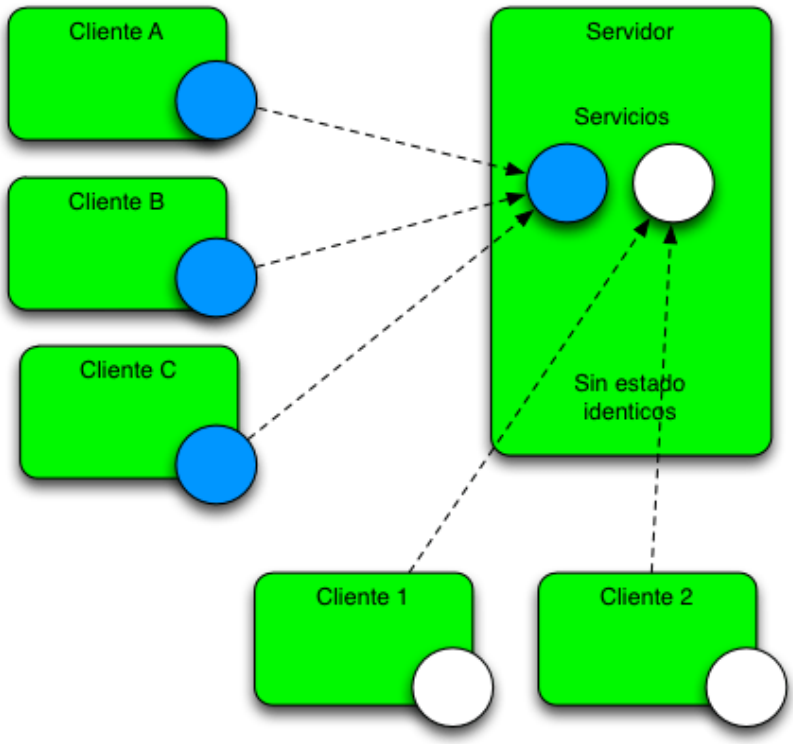


Figura 2.5: Ejemplo de consumición de un servicio REST.

Fuente https://www.arquitecturajava.com/servicios-rest/

Es perfectamente viable cachear los servicios desarrollados en REST. De esta manera, la primera petición realizada al servicio queda cacheada y si en un futuro esa misma petición es ejecutada, el servidor puede recurrir a esos datos cacheados para ejecutarla más rápidamente (ver Figura 2.6).

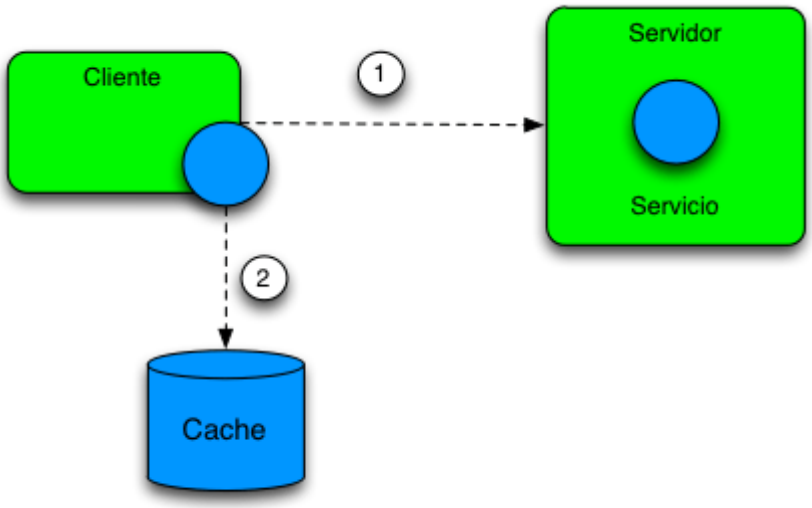


Figura 2.6: Uso de cache en un servicio REST.
Fuente https://www.arquitecturajava.com/servicios-rest/

Cuando navegamos por una página web, vemos que las URLs van cambiando dependiendo de la página, sección o documento que estemos visualizando en dicha página. REST utiliza este mismo concepto mediante el uso de URIs (Uniform Resource Identifier): nos permiten declarar un recurso dentro del servidor que ejecuta una tarea específica.

Las URIs deben seguir un formato básico aceptado por toda la comunidad:

- Deben tener un nombre único, puesto que cada recurso debería realizar una acción diferente al resto de recursos existentes.
- Deben seguir una estructura jerárquica lógica. Si deseamos obtener la información de una imagen cuyo identificador es 45 del cliente con identificador 20, la URI no debería tener el siguiente formato: /imágenes/45/cliente/20. El formato correcto sería: /cliente/20/imágenes/45.
- El nombre de las URIs no debe implicar una acción y es por esto que se evita el uso de verbos en la declaración del recurso

Las operaciones más comunes que se implementa a través de URIs son las que nos permiten realizar el clásico CRUD de objetos (Create, Read, Update y Delete). Estas operaciones se implementan mediante el uso de POST (crear), GET (consultar y listar), PUT (editar) y DELETE (eliminar).

2.3.3. ¿Qué es Alpha Vantage?

Alpha Vantage es una empresa que proporciona gratuitamente datos financieros de empresas a los usuarios que se registran en su portal. Estos datos los proporcionan a través de su REST API, siendo solamente necesario una Key única que conceden a los usuarios registrados. Al emplear una licencia gratuita de esta API para el desarrollo de este proyecto, existen una serie de limitaciones que se detallarán en la sección 4.4 Limitaciones de la aplicación.

Se trata de una alternativa excelente al famoso Yahoo Finance API (actualmente inactiva y cerrada), que de igual manera que Alpha Vantage, proporcionaba los datos históricos y en tiempo real de las acciones de empresas en formato csv y json.

En su documentación se puede encontrar las configuraciones y parámetros necesarios para obtener dichos datos. Además, se puede visualizar el valor de las criptomonedas dependiendo del mercado especificado.

2.4. Spring Boot

2.4.1. Aspectos básicos de Spring Boot

Spring Boot es una tecnología que facilita y ahorra el trabajo de configurar aplicaciones basadas en Spring. Spring es un *framework* para el desarrollo de aplicaciones escrito inicialmente por Rod Johnson y lanzado en junio del 2003. Pronto se convirtió en el *framework* más popular de Java empresarial por sus características revolucionarias, como la posibilidad de crear código de alto rendimiento y reutilizable.

La principal desventaja que presenta Spring es su compleja y tediosa configuración: selección de archivos jars con Maven, despliegue en el servidor, etc. Spring Boot nació para solucionar este problema, pues simplifica los pasos de la configuración y permite centrarse únicamente en el desarrollo de la aplicación.

Spring Boot ofrece los siguientes beneficios:

- Esta organizado de forma modular. Uno de los módulos que integra es un complejo módulo de autoconfiguración de la aplicación, para que el desarrollador no tenga que realizar esta configuración.
- Permite su ejecución en modo *Stand-alone*, aunque también cuenta con la posibilidad de ejecutar aplicaciones web con el uso de servidores webs integrados, como es el caso de TomCat.
- Al crear y elegir el tipo de proyecto, automáticamente se agregan las dependencias necesarias para el correcto funcionamiento de la aplicación. Además, cuenta con la posibilidad de agregar dependencias externas.
- Cuenta con *plugins* desarrollados por la comunidad que facilitan aún más tareas de desarrollo.

2.5. Eclipse

Se trata de un entorno de desarrollo multiplataforma que hace uso de diversas herramientas de programación de código abierto. Eclipse fue creado inicialmente por IBM, hasta que la Fundación Eclipse tomó el relevo en el desarrollo y evolución de esta herramienta. Cuenta con un editor de texto y con un analizador sintáctico. Además, incorpora la posibilidad de realizar pruebas unitarias con Junit junto con un control de versiones del código y un amplio abanico de *plugins*.

El desarrollo en Eclipse se basa en proyectos: un conjunto de carpetas y ficheros estructurados, donde nos podemos encontrar archivos de código fuente, ficheros de configuración, documentación, etc. Sin duda, el lenguaje de programación por excelencia en este entorno es Java, siendo el 92,66% de las líneas de código escritas en su editor de texto.

2.6. MySQL

2.6.1. Aspectos básicos de MySQL

MySQL es un sistema de gestión de bases de datos relacional, originalmente desarrollado por MySQL AB, refactorizada por MicroSystems en 2008 y finalmente adquirida por Oracle Corporation en 2010 (ver Figura 2.7). Este sistema de gestión cuenta con una doble licencia: código abierto (licencia pública general) y versión comercial (gestionada por Oracle).

MySQL presenta las siguientes características principales:

- **Arquitectura basada en Cliente-Servidor:** esta característica permite a cada cliente realizar consultas a la misma base de datos para leer, modificar, crear o eliminar registros. Esta comunicación que se realiza de manera individual y diferenciada, permite que tenga un menor impacto en el rendimiento al realizar estas consultas simultáneas.
- **Implementación de transacciones:** se trata de un conjunto de órdenes que están relacionadas entre sí, pues tiene que ejecutarse conjuntamente, sin posibilidad de división para garantizar la integridad de los datos almacenados.
- **Implementación de Triggers:** permite crear “disparadores” que se ejecutan cada cierto tiempo o al cumplirse una condición especifica. Estos Triggers se usan para automatizar ciertas tareas, como la actualización de registros.

Figura 2.7: Evolución de MySQL.
Fuente: https://openwebinars.net/blog/que-es-mysql/

Sección 3

Diseño de la página web

En esta sección se explicará el motivo por el cual se crearon las pantallas de la página web, así como las diferentes opciones que se muestran en cada una de ellas. Existen algunas pantallas que tienen un acceso privado, únicas por cada usuario, y hay otras que son accesibles por cualquier usuario.

3.1. Diseño y funcionalidades

3.1.1. Estándares Web W3C

Alrededor de año 2000, la construcción y mantenimiento de páginas web era un proceso caótico y nada fácil de realizar. Esto era debido a que una página web se puede visualizar en diferentes dispositivos, navegadores web o sistemas operativos, lo que provocaba que elementos en la página web no funcionaran correctamente, que algunos contenidos no se visualizaran, el tamaño de las fuentes cambie sin ningún motivo, etc.

Para solventar estos problemas, se decidió establecer y aplicar un conjunto de estándares, patrones y normas que los diseñadores, programadores y editores de páginas web debían cumplir si querían que su web estuviera exenta de errores según la plataforma de visualización. El precursor de esta iniciativa para que la web alcanzara su máximo potencial, permitiendo que una web sea accesible por todos, fue el World Wide Web Consortium (W3C). El W3C es una comunidad internacional que pretende fomentar el uso de reglas y estándares en la web, cuyo fundador fue Tim Berners-Lee, uno de los padres de la web.

La creación de un estándar Web tiene que contemplar y tener en cuenta diferentes puntos de vista de las tecnologías empleadas para su desarrollo, generando de esta manera, un estándar de calidad que asegure que un contenido está disponible y fácil de usar por todos los usuarios. En este proceso de creación de un estándar, intervienen más de 400 organizaciones nacionales o internacionales, expertos en software y hardware, universidades, etc.

Los estándares web, hacen referencia directa a las diferentes partes del código que se usan para generar una web. Estas partes son códigos escritos en HTML, CSS, XML y JavaScript. El código de cada una de estas partes, está regido por diferentes estándares, pues cada uno tiene sus características.

Estos estándares se han tenido en cuenta a la hora de desarrollar la aplicación web, ya que uno de los objetivos planteados inicialmente era que la web fuera accesible y compatible para la mayoría de los usuarios. Además, el hecho de aplicar estas normas ha permitido que el tiempo de desarrollo y mantenimiento se reduzca, puesto que, si es necesario introducir un cambio o agregar una nueva funcionalidad, es mucho más sencillo porque el código sigue un patrón lógico.

3.1.2. Accesibilidad Web

Otro de los puntos importantes cuando se habla de estándares de calidad de la web, es la accesibilidad dentro de la web. La accesibilidad tiene como objetivo conseguir que la mayoría de personas puedan utilizar y navegar en una página web, con independencia de sus capacidades personales, su formación o de su conocimiento técnico sobre informática.

Por lo tanto, una web accesible se define como una página que muestra contenidos y éstos, pueden ser comprendidos, visualizados y con la posibilidad de interacción por la mayoría de personas. En este grupo también se incluyen las personas que padezcan algún tipo de discapacidad física o intelectual.

Se tienen que desarrollar mecanismos para que personas con problemas auditivos, visuales, físicos o cognitivos puedan interactuar con una web sin ningún tipo de impedimentos para que todos los usuarios tengan un acceso igualitario y con las mismas posibilidades de interacción.

Existen diferentes estándares de calidad para clasificar la accesibilidad de una web denominados niveles de conformidad. Estos niveles son A, AA y AAA, siendo esta última el nivel más alto de calidad en cuanto a la accesibilidad se refiere. La aplicación web se ha desarrollado, en la medida de lo posible, para que cumpla con el nivel de accesibilidad mínimo (nivel A).

3.1.3. Funcionalidades básicas de la aplicación

Las funcionalidades básicas que deberían estar presente en la web son las siguientes:

- Visualización de los datos financieros e históricos de las acciones de una empresa, representados en graficas con las que el usuario puede interactuar.
- Buscar empresas por nombre o por símbolo de la empresa para poder visualizar sus datos.
- Login y registro de usuarios dentro de la aplicación.
- Acceder a un área privada donde el usuario podrá visualizar sus datos personales.
- Agregar, eliminar acciones de una empresa y poder modificar el número de acciones que el usuario posee.
- Posibilidad de comparar las acciones de 2 empresas en el mismo marco de tiempo para contrastar su evolución.
- Posibilidad de filtrar la información financiera de las empresas, así como la posibilidad de descargar las gráficas y los datos que contienen en diferentes formatos.

3.2. Modelado de la interfaz

3.2.1. Visión del diseño mediante mockups

En esta subsección se realizarán iteraciones por la aplicación a través de unos diseños preliminares llamados mockups. Para la realización de estos mockups, se han tenido en cuenta los objetivos y las funcionalidades descritas en los puntos 1.2 Objetivo y 3.1.3 Funcionalidades básicas de la aplicación.

El primer diseño de la pantalla principal de la página web contiene una tabla que permite visualizar datos en función de la pestaña en la que el usuario se encuentre situado. Como podemos ver en la Figura 3.1, el usuario estaría consultando el valor de los principales índices bursátiles de la bolsa. Estos índices estarían clasificados por continentes y tendría una sección de divisas y criptodivisas. Por ejemplo, se mostrarían los valores del Ibex35 (índice de Bolsas y Mercados Españoles), FTSE MIB (índice del Mercados Italianos), Nikkei 225 (índice de Mercados Japoneses), etc. Al estar situado en la pestaña “Vista Global”, aparecen los principales índices globales.

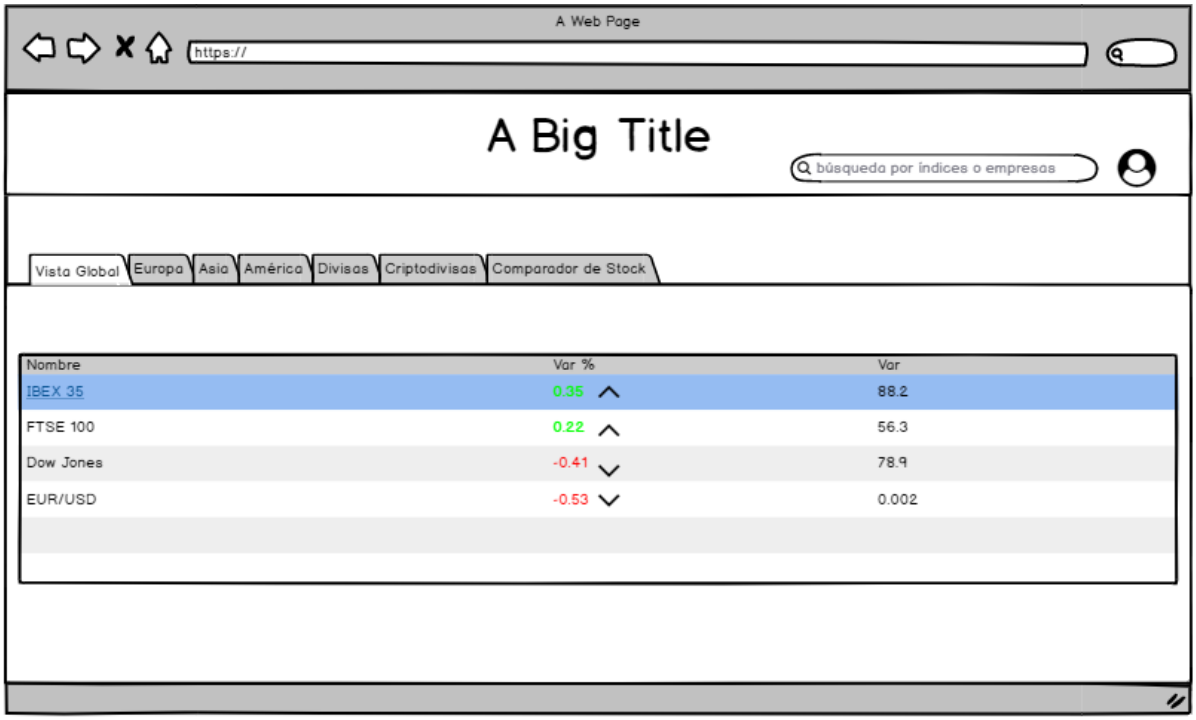
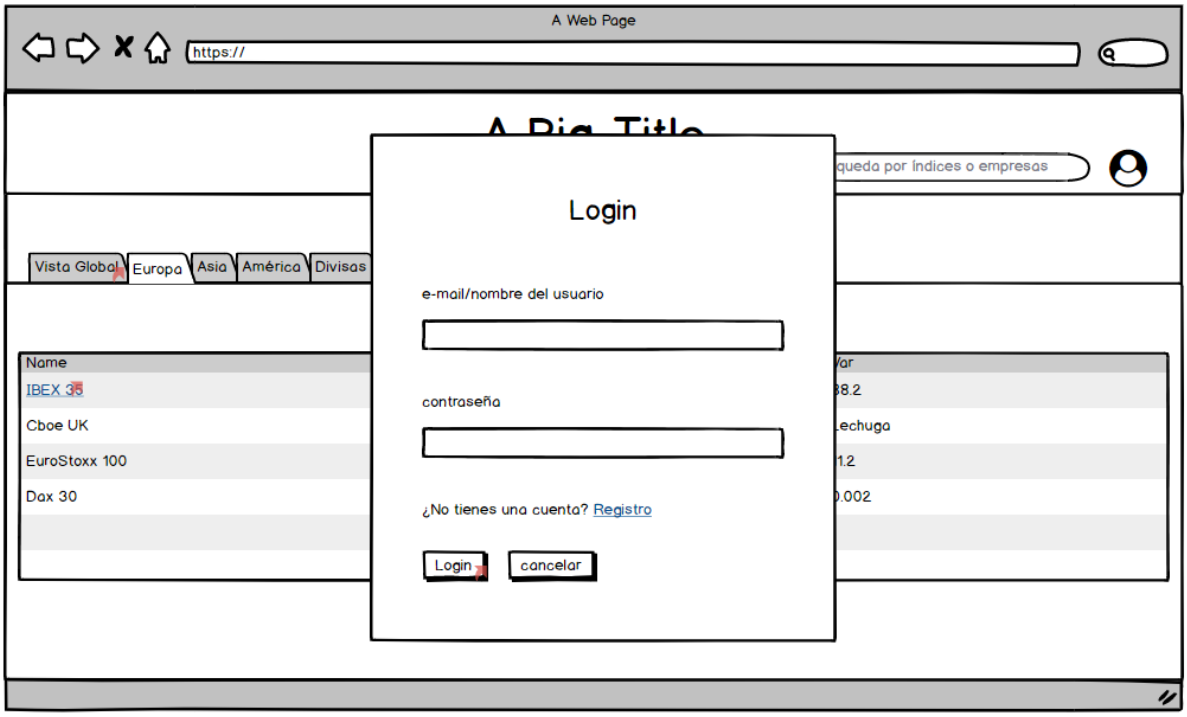


Figura 3.1: Pantalla de inicio de la web.

Si el usuario se sitúa en la pestaña Europa, el contenido de la tabla cambia y se muestran los índices bursátiles junto con sus respectivos valores en tiempo real. Como se puede observar en la Figura 3.2, las pestañas de los países están situadas junto a la pestaña de “Vista Global”. Cuando el usuario itere sobre las pestañas, no será necesario un refresco de la página, pues solo cambia el contenido que se muestra en la tabla

Figura 3.2: Navegación entre los principales índices bursátiles por país.

Al tratarse de una página web en la que usuarios podían consultar información pública (datos financieros) y otros en cambio información privada (datos de acciones adquiridas), era necesario la creación de una pantalla de *login* y otra pantalla de registro. A través de estas dos pantallas, los usuarios pueden registrarse dentro de la aplicación y volver a loguearse posteriormente introduciendo las credenciales indicadas en el proceso de alta (ver Figura 3.3).



.

Figura 3.3: Pantalla de *Login*.

Al identificarse como un usuario registrado, aparecerá un área privada denominada *Mi cartera* (ver Figura 3.4). En esta área privada, se podrá consultar el valor de las acciones de todas las empresas que posea el usuario. Esta información se representará de manera sencilla de entender de un solo vistazo en formato tabla.

En esta área privada, el usuario podrá gestionar las acciones de las empresas de las que dispone. Podrá agregar y eliminar empresas, así como modificar el número de acciones que ha adquirido de dichas empresas mediante controles situados en cada registro de la tabla. Junto con esos controles, se mostrará el valor actualizado de las acciones.

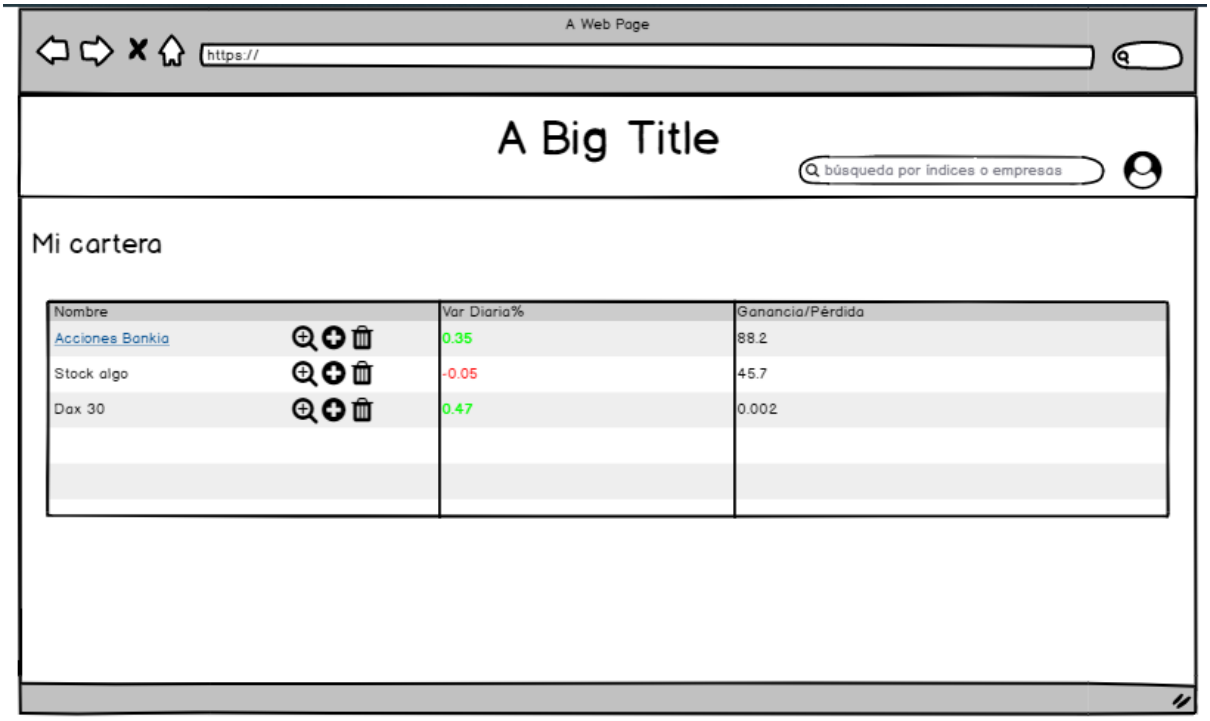


Figura 3.4: Área privada del usuario.

Otra de las pantallas de la aplicación es la visualización de gráficos. Se trata de una de las pantallas principales de la aplicación, pues es la que permite mostrar los datos financieros e históricos tanto en gráficas y en tablas. Además, contará con opciones como cambiar el tipo de gráfica, elegir el rango de tiempo para visualizar los datos o incluso con la posibilidad de comparar acciones de otra empresa dada la misma franja de tiempo (ver Figura 3.5 y Figura 3.6).

Los controles de filtrado o de formato de representación de las gráficas deberían ser fáciles de usar e intuitivos. Por ello, se optó por un diseño simple con uso de pocos colores en los demás elementos, para que estos controles resaltaran y fueran fáciles de localizar.

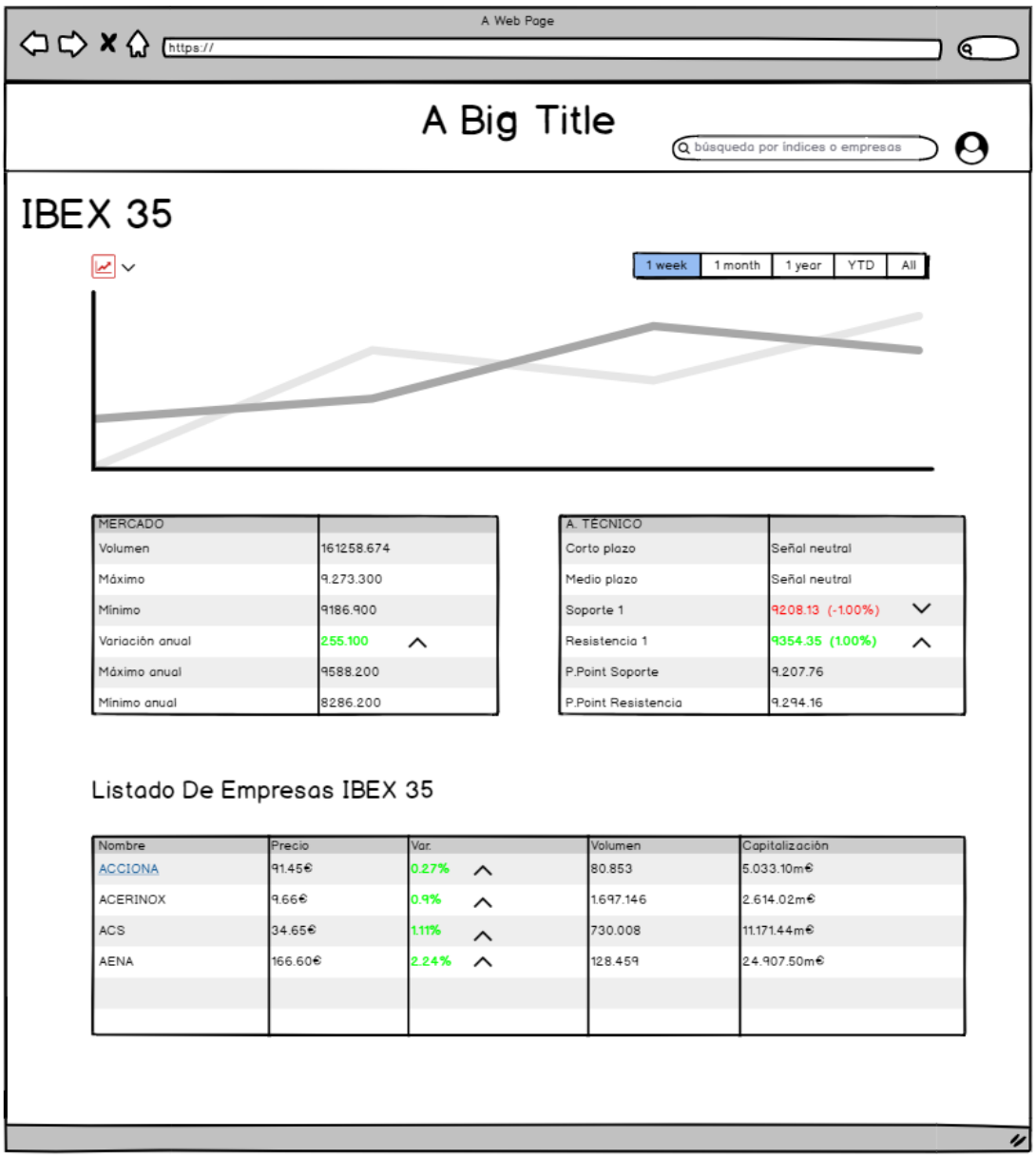


Figura 3.5: Visualización de graficas representando datos financieros e históricos.

En la Figura 3.6 se puede observar la comparación de datos entre empresas de manera gráfica y mediante el uso de tablas. Esta grafica comparativa contaría con las mismas opciones de filtrado que las gráficas estándares.

Se decidió que debería existir un menú simple situado en la parte superior de la pantalla, por el cual el usuario pudiera realizar acciones habituales: buscar empresas para visualizar datos o acciones de autenticación dentro de la web (login, registro, cerrar sesión). Este menú debería aparecer en el mayor número de pantallas posible.

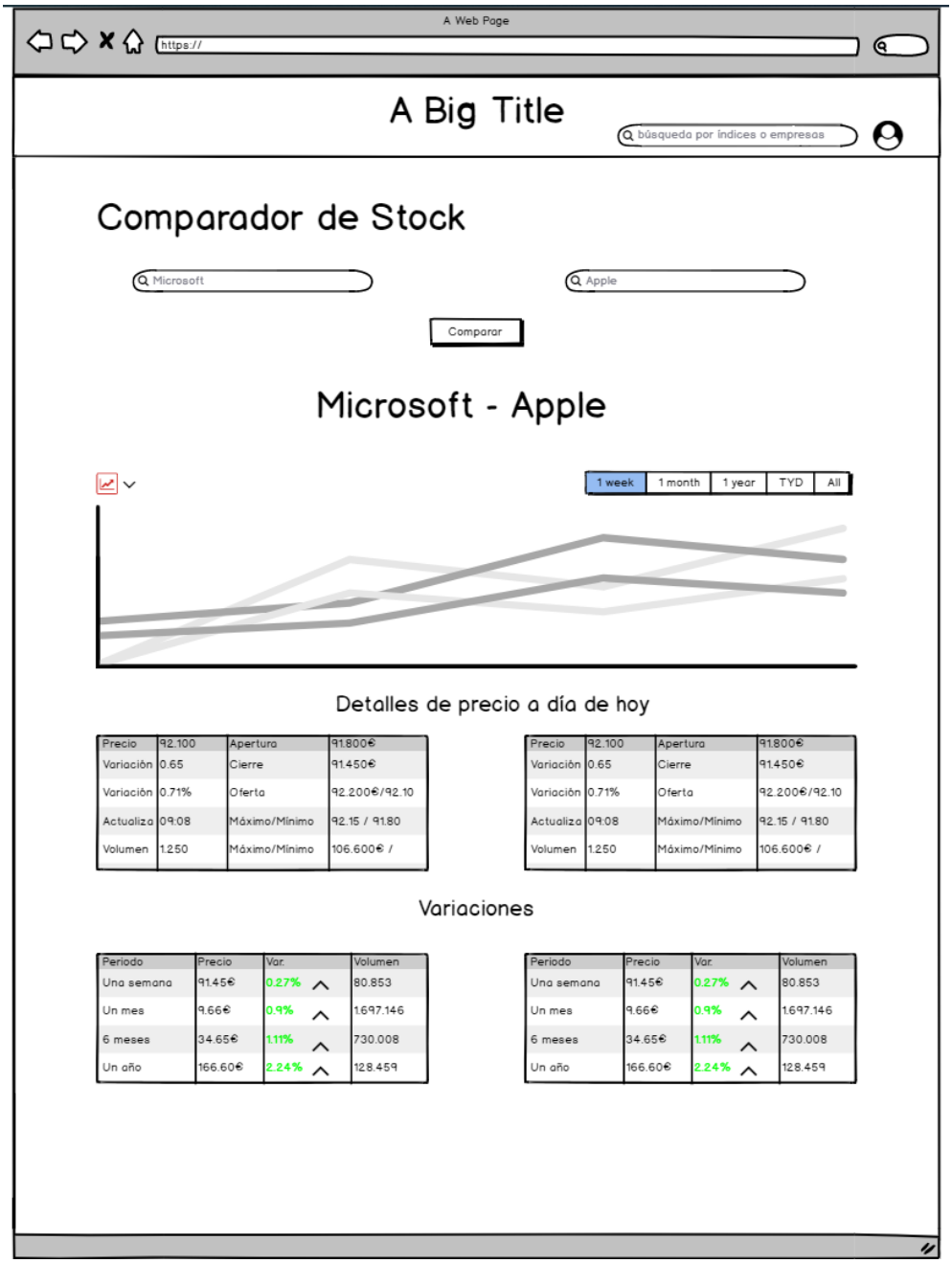


Figura 3.6: Comparación de datos financieros entre empresas.

Sección 4

Implementación

En esta sección se detallará la arquitectura de la aplicación, explicando desde un punto de vista técnico, cómo interactúan el Front-End con el Back-End y viceversa. Además, se realizará un recorrido por las diferentes pantallas que ofrece la web, listando las posibles acciones que existen en estas pantallas y cómo son procesadas. Posteriormente, se enumerarán argumentos de por qué se han seleccionado las tecnologías empleadas para el desarrollo, indicando sus beneficios y sus posibles alternativas.

4.1. Descripción de la página web

Como se ha comentado en otros puntos del documento, la aplicación web tiene como objetivo mostrar datos financieros e históricos de las empresas a través del uso de gráficas que cuentan con filtros sobre estos datos. Esta información se emplea para visualizar las ganancias o pérdidas de un usuario que tenga acciones de una o varias de empresas, calculando estos beneficios o pérdidas con datos actualizados.

Esta página web se ha desarrollado teniendo en cuenta los estándares y normas especificados por W3C, tanto para el código HTML como para el código CSS. Además, esta web cumple con el nivel A de accesibilidad, por lo que se pueden visualizar los contenidos en otros formatos como móviles o tablets, ofreciendo una versión *responsive* de la aplicación.

4.2. Arquitectura de la aplicación

La parte Front-End de esta aplicación web está desarrollada con Angular versión 7.2.15 y la parte de Back-End está desarrollada con Spring Boot. Ambas partes se comunican mediante el uso del protocolo HTTP, haciendo uso de una API REST. El Back-End recibe peticiones generadas por el Front-End y estas peticiones son procesadas, devolviendo una respuesta en el proceso. Las peticiones son enviadas y recibidas en formato JSON, para el intercambio de datos entre el cliente y el servidor.

Algunas peticiones pueden contener datos más comprometidos que otros. Por ejemplo, no es lo mismo realizar una petición para buscar los datos de una empresa que obtener los datos de las acciones de un usuario. Por ello, se ha desarrollado un mecanismo que permita realizar estas acciones más comprometidas con mayor seguridad. Esto ha sido posible gracias a la creación de un token de autenticación que se genera y guarda el usuario en su cliente cuando se completa satisfactoriamente el proceso de login.

El token usa el formato JSON Web Tokens (JWT), el cual se caracteriza por ser una secuencia de números y letras separadas por tres puntos:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI1NGE4Y2U2MThlOTFiMGIxMzY2NWUyZjkiLCJpYXQiOiIxNDI0MTgwNDg0IiwiZXhwIjoiMTQyNTM5MDE0MiJ9.yk4nouUteW54F1HbWtgg1wJxeDjqDA_8AhUPyjE5K0U

Como se ha mencionado previamente, este token es almacenado en el lado del cliente en una sesión. Cuando el usuario realiza una acción, genera una petición al servidor. Si esta acción es considerada como comprometida, la petición resultante hará uso del token para validar su autenticidad. Para ello, la cadena de caracteres es incorporada en la cabecera de la petición y enviada y validada por el Back-End. Si el token es válido y no esta expirado (los tokens tienen una caducidad de 24 horas), la petición será procesada.

Otras peticiones, necesitan hacer uso de una base de datos para guardar información que será necesaria en un futuro. Por eso, Spring Boot está conectada a una BBDD relacional que usa MySQL. Un ejemplo de petición que requiere consulta a la base de datos es obtener las acciones y el número de ellas que posee un usuario en concreto.

El Front-End desarrollado en Angular sigue la siguiente estructura:

- **Components:** en esta carpeta se encuentran todos los componentes que se han empleado para la construcción de la página. Cada componente tiene su carpeta identificada por un nombre, en la que se alojan 3 archivos:

- **Archivo CSS:** se define los estilos de las clases que emplea el componente
- **Archivo HTML:** se define la vista del componente.
- **Archivo TypeScript:** se define la lógica que puede ejecutar el componente.

- **Models:** contiene todos los modelos empleados a lo largo del desarrollo de la aplicación. Cada modelo tiene definidos unos atributos y el tipo de dato de dichos atributos.
- **Services:** contiene todos los servicios que conectan las acciones que ocurren con la vista, con la API REST. Por lo tanto, estos se encargan de generar las peticiones y recoger la respuesta del servidor.
- **Views:** carpeta que contiene las diferentes vistas de la aplicación. Estas vistas se caracterizar por el uso de 2 o más componentes y cada vista tiene asociada una url accesible dentro de la web. Estas rutas se encuentran definidas en el archivo app-routing.module situado en la raíz del proyecto.

En cuanto al Back-End desarrollado en Spring Boot, se ha seguido la siguiente estructura de paquetes:

- **Main:** contiene la lógica de arranque de la aplicación.
- **Controllers:** contienen todos los controladores de la aplicación. Son los encargados de recibir las peticiones generadas en el Front-End y llamar a los DataServices. Cuando los dataServices hayan procesado y generado una respuesta, los controladores son los encargados de devolver dicha respuesta.
- **Repositories:** contienen todos los repositorios que se conectan a la BBDD para realizar consultas sobre esta.
- **DataServices:** contienen la lógica que se debe llevar a cabo dada una entrada de datos entregada por los controladores. Una vez generado un resultado dada esa entrada de datos, este resultado es entregado a los controladores.
- **VO:** contiene diferentes modelos que se emplean a lo largo del desarrollo. Cada modelo tiene sus atributos definidos.
- **VO.Config:** contiene la lógica de generación y validación del token de autenticación JWT.
- **VO.Request:** contiene todos los modelos de tipo *request*.
- **VO.Response:** contiene todos los modelos de tipo *response*.

4.2.1 Funcionalidades y recorrido por la aplicación

Al entrar en la web, la primera pantalla que visualiza el usuario es la pantalla *Home*. En esta pantalla. Se muestra una barra de búsqueda en la que podrá escribir los símbolos o el nombre de la empresa, para posteriormente visualizar sus datos. Junto a esta barra de búsqueda, existirá una pequeña descripción de la web y contará con accesos directos de login o registro (ver Figura 4.1).

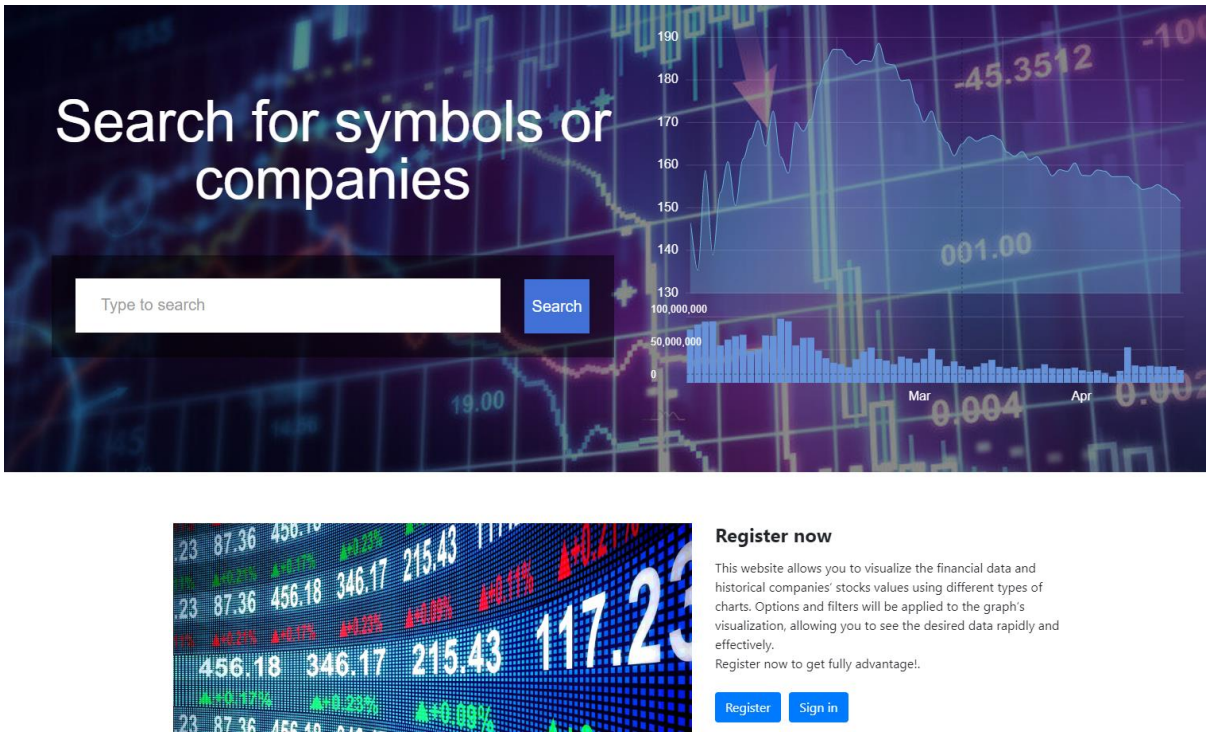


Figura 4.1: Pantalla *Home*.

Otra de las pantallas principales de la web es la de la visualización de datos en graficas (ver Figura 4.2). A esta pantalla se puede acceder tras haber introducido el nombre o símbolo de una empresa. Lo primero que el usuario visualiza es la gráfica en formato de líneas. Por defecto, esta grafica está representada en este formato, pero el usuario puede cambiarlo en cualquier momento a través del desplegable "*Select a chart*", donde aparecerán otras opciones de representación. También está por defecto el rango de tiempo de un día "*Intraday*", por lo que la información está referenciando a los datos que se han generado en las últimas 24 horas. Al igual que en las opciones de representación de gráficas, el rango de tiempo también se puede cambiar mediante el desplegable "*Select a TimeSeries*" (ver Figura 4.3), donde aparecerán otras alternativas, tales como "*Daily*" y "*Monthly*".

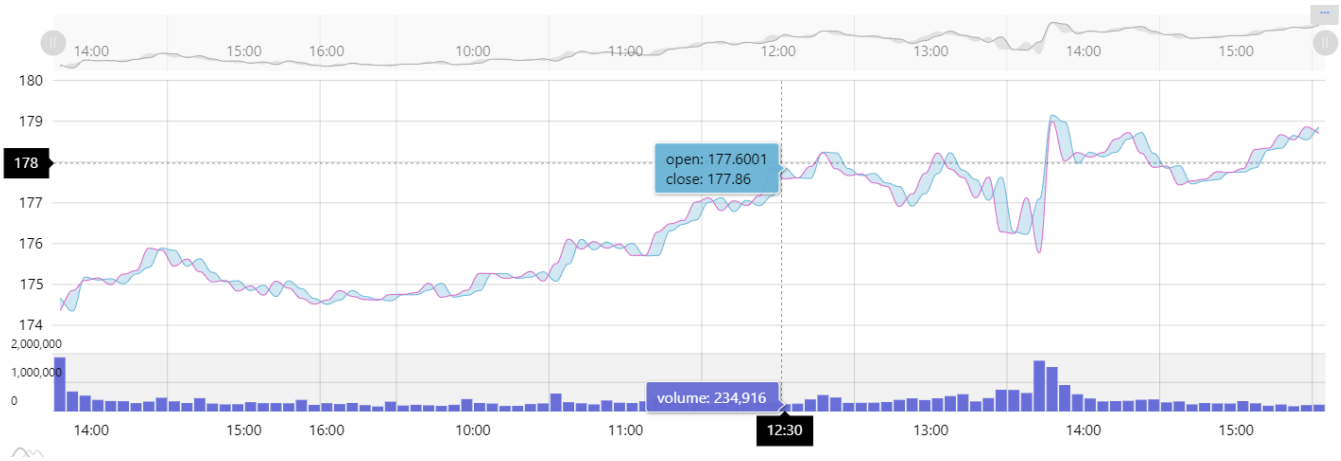


Figura 4.2: Gráfica de datos financieros.

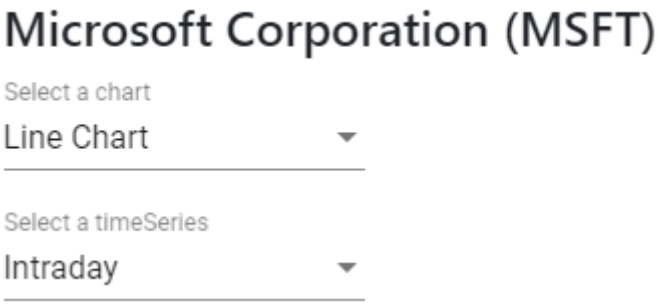


Figura 4.3: Controles de tipo de gráfica y rango de tiempo.

Las gráficas son representadas mediante una biblioteca de JavaScript llamada amCharts 4. La biblioteca permite representar un conjunto de datos de entrada de forma gráfica, haciendo uso de diagramas de barras, circulares, etc. Puede ser usada en cualquier tecnología compatible con JavaScript/TypeScript, por lo que puede ser empleada en entornos como Angular, React, Vue.js, etc.

El usuario puede interactuar con el grafico de varias maneras. Si sitúa el ratón sobre este, aparecerán representados los datos de las acciones de esa franja de tiempo (open, close, volumen, etc). Se puede realizar zoom en el grafico si el usuario hace clic en un punto y arrastra el ratón hasta otra zona, que indicará el final del zoom realizado. Otra manera de realizar zoom es a través del control situado en la parte superior de la gráfica. A parte de indicarnos los rangos de tiempo de los datos que se están visualizando, permite acotar estos rangos mediante dos controles claramente definidos (ver Figura 4.4). Para acotar estos rangos, el usuario deberá hacer clic en uno de los controles y arrastrarlo hasta llegar a la franja de tiempo deseada.

Al hacer zoom, aparecerá el siguiente control en forma de botón,  situado en la parte superior izquierda que servirá para reestablecer el zoom por defecto. Para las opciones con un rango de tiempo amplio, se ha introducido un selector en forma de calendario para facilitar al usuario esta tarea de zoom (ver Figura 4.5).

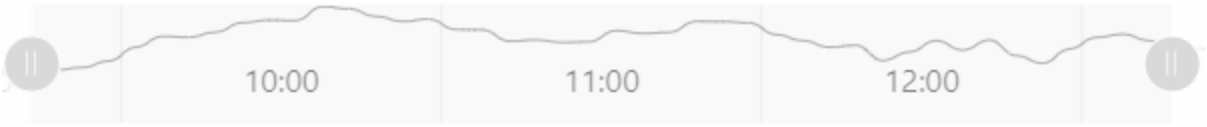


Figura 4.4: Control del rango de tiempo.

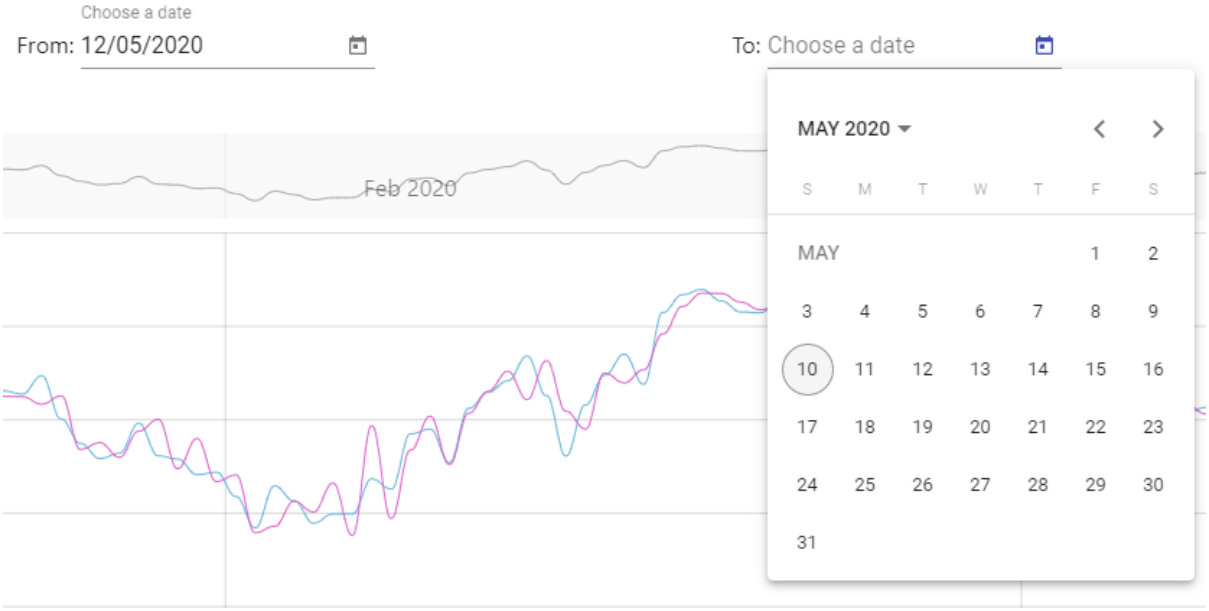
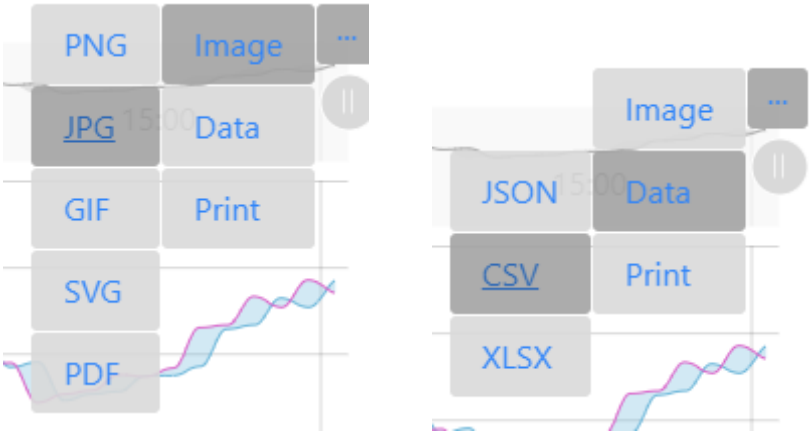


Figura 4.5: Control de rango de tiempo en formato calendario.

La grafica cuenta con varias opciones adicionales al pulsar en el botón (ver Figura 4.6a y Figura 4.6b):

- Se puede crear y exportar una imagen de la gráfica actual en diferentes formatos: PNG, JPG, SVG, PDF, etc.
- Se puede extraer y exportar los datos que contiene la gráfica en los siguientes formatos: JSON, CSV, XLSX (formato Excel).

- Se puede imprimir la gráfica.



(a) (b)

Figura 4.6: Opciones de exportación de datos de la gráfica.

En la parte superior de la pantalla, se puede observar un pequeño menú que contiene el símbolo de una lupa  . Esta lupa permite buscar por nombre o símbolo una empresa. Al igual que en la barra de búsqueda del *Home*, esta búsqueda cuenta con la función de autocompletar a medida que el usuario va introduciendo caracteres (ver Figura 4.7).

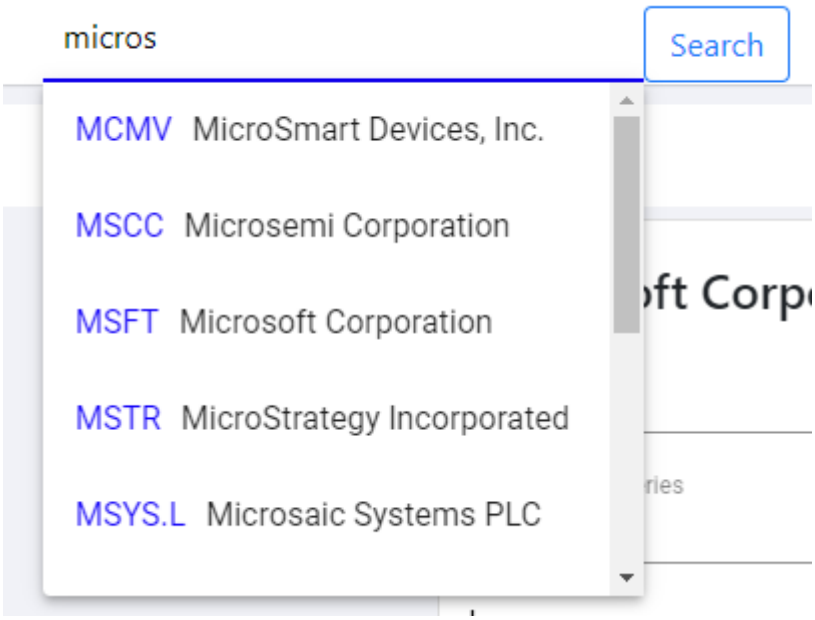


Figura 4.7: Función de autocompletado de empresas.

Otros elementos que se encuentran dentro de este menú son accesos directos de *login* y registro. Tanto el *login* como el registro se encuentran en pantallas diferentes. En la pantalla de *login*, el usuario deberá introducir las credenciales. Estas credenciales son el email y la contraseña. Si la validación es correcta, el usuario podrá acceder a la parte privada *Mi cartera*. Al realizarse esta acción de *login*, el servidor genera un token único y que cuenta con una fecha de caducidad, que se almacena en una sesión en el cliente. Este token se emplea para realizar peticiones al servidor, cuando el origen de estas llamadas contiene información comprometida. Un ejemplo de una petición que usa este token de autenticación seria la consulta de la lista de acciones de las empresas que tiene asociadas. Otras peticiones como la visualización de datos de una empresa, no hacen uso de este token, puesto que es información pública.

Por último, en el menú existe el componente *Migas de pan,* que permiten mostrar al usuario el nivel de la página en la que se encuentra, es decir, indicar las secciones padres y la sección actual en la que se encuentra navegando (ver Figura 4.8).

Figura 4.8: Componentes *Migas de pan*.

En las opciones de la gráfica, el usuario podrá hacer clic en el icono  para poder introducir el nombre o símbolo de la empresa que desea comparar con la empresa que se visualiza actualmente. Las opciones de comparación de gráficas, cuentan con las mismas opciones que la gráfica individual (selección rango de tiempo, zoom de datos, exportar datos, etc).

En la pantalla *Mi cartera*, el usuario puede visualizar la información de empresas y las acciones que dispone, mostrando el valor de esas acciones y el beneficio que ha obtenido de ellas. El número de estas acciones se pueden cambiar a través de controles de números situados en una columna de la tabla (ver Figura 4.9).

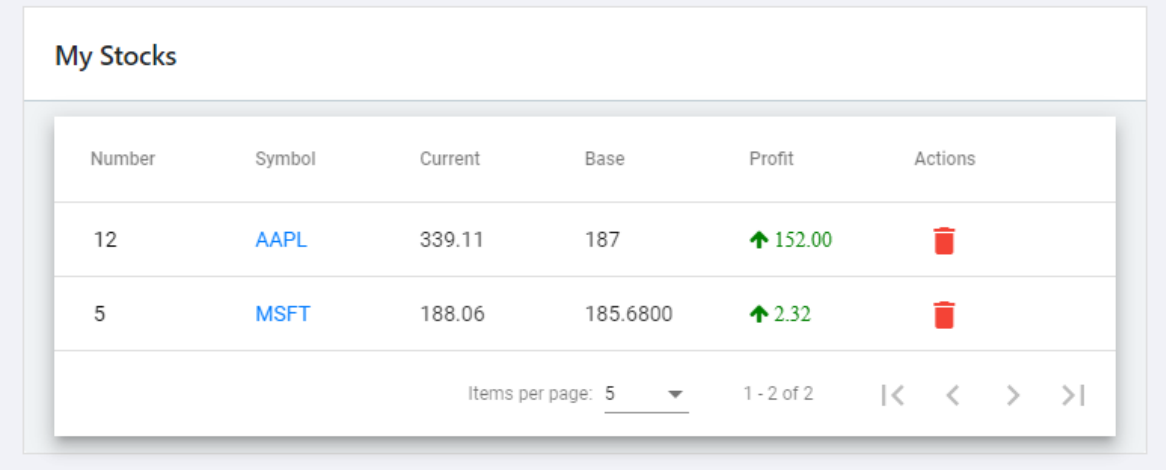


Figura 4.9: Lista de acciones de empresas del usuario.

En esta pantalla aparecen los graficos de lineas de los simbolos de las empresas asociadas al usuario. Esto permite una visualizacion rapida de las empresas de las que dispone acciones sin necesidad de buscar cada empresa de manera individual (ver Figura 4.10). Se generaran tantos graficos de lineas como empresas tenga almacenadas el usuario.

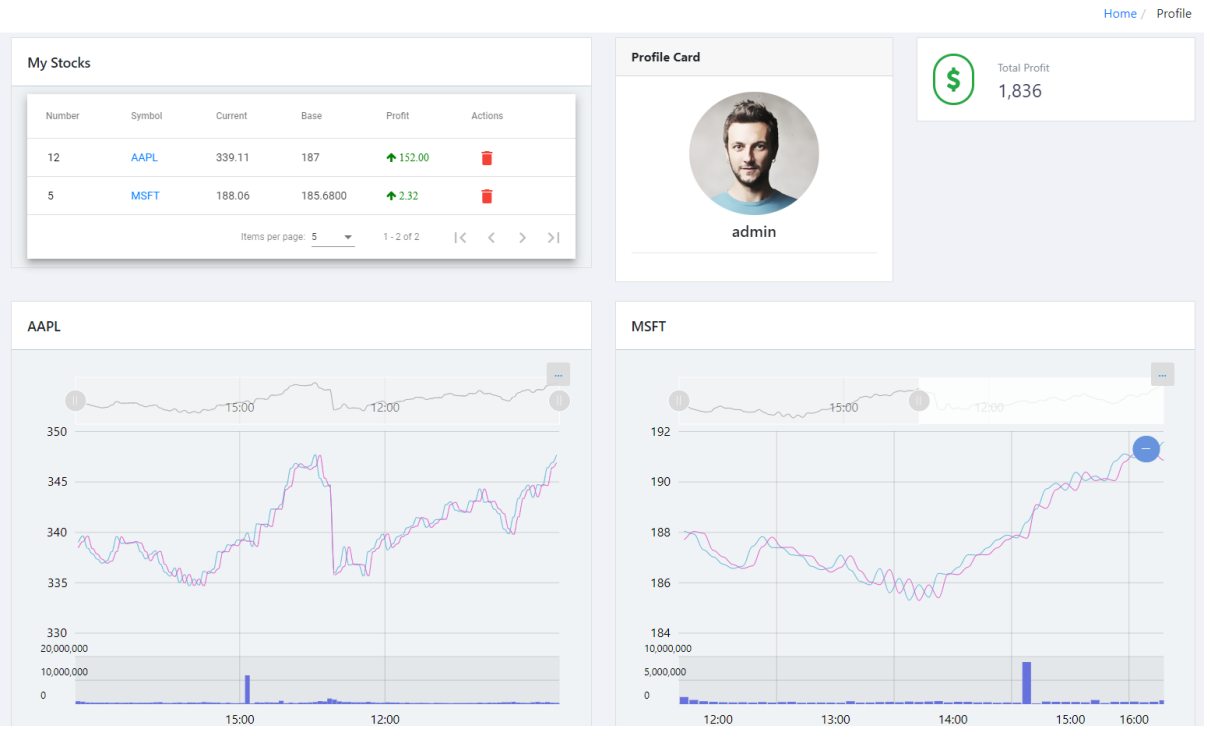


Figura 4.10: Pantalla *Mi cartera*.

4.2.2. Servicios web externos

Como se ha mencionado en el punto 3.2.3 del presente documento, la aplicación usa una API REST llamada Alpha Vantage para consumir datos sobre las acciones de empresas. A parte de poder consumir estos datos, también ofrece la posibilidad de consultar información relevante sobre las criptomonedas (Bitcoin, Ethereum, etc), indicadores técnicos y rendimiento por sectores (Sanidad, Servicios de comunicación, Industrial, etc.).

La funcionalidad de obtener los históricos de las acciones, permite pasarle una serie de parámetros (siendo algunos opcionales y otros obligatorios) para obtener un resultado concreto. Los parámetros obligatorios para generar la petición son los siguientes:

- ***Function:*** permite seleccionar el rango de tiempo para la obtención de los datos. Estas opciones de rango pueden ser diarias, por semanas, por meses, etc.
- ***Symbol:*** símbolo de la empresa que se desea consultar. Cada empresa está identificada por un símbolo único e inequívoco. Por ejemplo, Microsoft Corporation se identifica por el símbolo MSFT y el símbolo de Apple es AAPL.
- ***Interval:*** intervalo de diferencia entre un dato y otro, es decir, el tiempo expresado en minutos que debe separar un valor de una acción y otra en un rango de tiempo definido. Las opciones pueden variar entre 1, 5, 15, 30 y 60 minutos.
- ***Api Key:*** clave única asociada a un usuario que permite realizar la petición.

Los parámetros opcionales para la consulta de datos son los siguientes:

- ***Output Size:*** existen dos opciones dentro de este parámetro: compact y full. Por defecto la petición se realiza con el valor compact, pues la opción full permite obtener todos los datos de la petición generada. Por el contrario, la opción compact devuelve los últimos 100 datos del histórico, siendo esta opción más eficiente para reducir el tamaño de los datos que se tienen que enviar.
- ***DataType:*** permite seleccionar el formato con el que se quiere obtener y recoger los datos generados al llamar a la API. por defecto esta opción está fijada al formato json, aunque existe la posibilidad de enviar los datos en formato csv (archivo en formato Excel cuyos datos están separados por comas)

La configuración de los parámetros anteriores debe seguir un formato específico. Este formato sigue la misma nomenclatura que los query parameters, es decir, cada parámetro se identifica por un nombre y un valor asociado. Los parámetros están separados por el carácter &. Un ejemplo de una llamada a la API sería el siguiente:

https://www.alphavantage.co/query?**function**=TIME_SERIES_INTRADAY&**symbol**=IBM&**interval**=5min&**outputsize**=full&**apikey**=demo

Respuesta que devuelve la llamada:


```
        {
            "Meta Data": {
                "1. Information": "Intraday (5min) open, high, low, close prices and volume",
                "2. Symbol": "IBM",
                "3. Last Refreshed": "2020-04-30 16:00:00",
                "4. Interval": "5min",
                "5. Output Size": "Full size",
                "6. Time Zone": "US/Eastern"
            },
            "Time Series (5min)": {
                "2020-04-30 16:00:00": {
                    "1. open": "126.1100",
                    "2. high": "126.1350",
                    "3. low": "125.7700",
                    "4. close": "125.8000",
                    "5. volume": "205326"
                },
                "2020-04-30 15:55:00": {
                    "1. open": "125.6600",
                    "2. high": "126.1600",
                    "3. low": "125.4000",
                    "4. close": "126.0800",
                    "5. volume": "162565"
```


4.2.3. Estructura de la base de datos

En la base de datos se pueden encontrar las siguientes tablas:

- ***FavouriteCompanies***: las columnas que contiene esta tabla son symbol, email y company_shares. Relacionan a un usuario con las acciones de una empresa.
- ***Symbols***: las columnas que contiene esta tabla son ticker y name. Cada registro identifica a una empresa por su símbolo y nombre únicos.
- ***Users***: las columnas que contiene esta tabla son userid, username, password, email, address, is_active, last_name1, last_name2, phone y rol.

Los registros que almacena la tabla *Symbols*, se consultan y se extraen en la funcionalidad de búsqueda autocompletada de empresas, tanto por su nombre como por su símbolo inequívoco.

Los datos alojados en la tabla *Users* son utilizados para las tareas de login y registro dentro de la web. Estos datos están relacionados con los registros de la tabla

FavouriteCompanies, que, mediante la vinculación del email del usuario de ambas tablas, se puede saber que empresas son las que el usuario tiene guardadas.

4.3. Elección de tecnologías y posibles alternativas

En esta sección se explicará brevemente por qué se han usado las tecnologías listadas en la sección 3 de este documento, para el desarrollo de esta aplicación web, así como sus posibles alternativas:

- **Angular:** se trata de uno de los *frameworks* para el desarrollo del Front-End más famosos por su arquitectura en forma de componentes y el uso del lenguaje TypeScript. Esto permite que la app sea fácilmente escalable mediante evolutivos. Otras buenas alternativas para el desarrollo Front son React (biblioteca de JavaScript de código abierto) o Vue (marco de JavaScript de código abierto basado en el Modelo-Vista-Controlador).
- **Spring Boot:** uno de los *frameworks* más utilizados en el mundo de Java, cuya popularidad ha ido creciendo de manera exponencial tras su salida. Su filosofía es que el desarrollador pierda el menor tiempo posible en la configuración del Back-End. Otras alternativas son Hibernate (permite el acceso a BBDD de alto nivel) o Struts 2 (facilita el desarrollo de aplicaciones web basadas en Java EE).
- **Alpha Vantage:** API gratuita sobre datos financieros que ofrece diversas configuraciones para la consumición de estos datos. Mediante el uso de parámetros, se puede configurar fácilmente la petición que se desea realizar. Con el cierre de Yahoo Finance API, Alpha Vantage se ha convertido en su sucesor por excelencia, pues ofrece un servicio rápido, eficiente y cuenta con una gran comunidad. Las mejores alternativas a esta API son World Trading Data,Quandl, Intrinio, etc.
- **MySQL:** se trata de la base de datos de código abierto más popular del mundo, pues permite la gestión de los datos almacenados en tablas de manera relacional. Otras alternativas a MySQL son MariaDB y Postgre SQL.

4.4. Limitaciones de la aplicación

La aplicación cuenta con una serie de limitaciones que se han ido presentando a medida que se realizaba el desarrollo y la implementación de la misma. La principal limitante es debida a las llamadas realizadas a la API Alpha Vantage. Cuando se produce la consumición de datos, la petición que se genera para consumirlos tiene que tener especificada obligatoriamente un parámetro llamado Api Key. Esta clave se le proporciona de manera totalmente gratuita a cada usuario para su libre uso. El impedimento de esta clave es que tiene un máximo de llamadas por minuto, siendo 5 el número máximo de llamadas. Por lo tanto, si un usuario que esté navegando realiza más de 5 llamadas en el mismo minuto, la aplicación mostrará un mensaje por pantalla indicando que no ha sido posible mostrar los datos, puesto que la respuesta de la API es nula por haber excedido ese límite.

Para mitigar esta limitación, se utilizan 3 claves distintas obtenidas mediante diferentes usuarios para diferentes secciones de la app: una de ellas se utiliza para obtener los datos que se deberán mostrar en una gráfica, otra se emplea para las comparaciones de datos de diferentes empresas que se muestran en otra gráfica, y la última se emplea para obtener el valor de las acciones que el usuario tiene compradas de diferentes empresas. Con estas funcionalidades con claves separadas, reduce la posibilidad de que un usuario no reciba respuesta debido a que se ha superado el límite por minuto. Alpha Vantage proporciona diferentes planes que permiten que una clave aumente sus prestaciones. De esta manera, se podría eliminar casi por completo este problema a la hora de usar la web.

Otro de las grandes limitantes que han surgido en el desarrollo es la búsqueda de empresas, que permite posteriormente la visualización de sus datos. Actualmente, esta búsqueda cuenta con una opción de autocompletado de palabras para facilitar el acceso al usuario: a medida que se teclea un símbolo o el nombre de una empresa, la búsqueda irá mostrando las opciones que se adecúen a los caracteres introducidos. Estas sugerencias anteriormente se explotaban de un recurso que proporcionaba Alpha Vantage. Esta opción se descartó puesto que cada carácter introducido tenía que hacer una llamada a la API para mostrar las opciones sugeridas. Como existe la limitación de 5 llamadas por minuto, en el momento que se tecleen 5 caracteres, la API no devolverá ninguna sugerencia pasado ese tiempo.

Para solventar esta incidencia, se obtuvo un registro de empresas identificadas con sus símbolos y se volcaron a la base de datos. De esta manera, cuando el usuario teclea un carácter nuevo, se realiza una petición y se obtienen las sugerencias relacionadas, pero estas sugerencias se obtienen de la tabla que contiene todos los registros de las empresas en lugar de consumir una petición a la API. Por lo tanto, el usuario puede teclear la empresa o símbolo deseado sin que exista posibilidad de devolución de datos nulo. La desventaja de esta implementación es que, al tratarse de 96679 registros de empresas y símbolos, puede que alguna no esté contemplada en Alpha Vantage, pues si realizamos una petición de una empresa cuyo símbolo no esté

registrado dentro de la API, nos devolverá una respuesta con un mensaje indicando que no se ha encontrado dicho símbolo.

Otro inconveniente que presenta este límite de 5 llamadas por minuto es cuando se quiere cargar el valor de las acciones de las empresas que el usuario tiene guardas. Si tiene más de 5 empresas, el valor de las acciones de la sexta empresa no se obtendrá. Este problema debería resolverse al contratar alguno de los planes de pago de la API, aumentando el número de consultas por minuto.

Sección 5

Conclusiones y Trabajo Futuro

En esta última sección, se describirán las conclusiones obtenidas y se listarán una serie de mejoras que se pueden implementar.

7.1. Conclusiones

Partiendo de la base de querer crear una aplicación, la cual permita al usuario valorar el progreso de los valores en los que actualmente se ha invertido en bolsa, se realizó un estudio para aprender todo lo referente a las acciones, cotización, volatilidad del mercado de valores, índices bursátiles, etc. En este proceso de aprendizaje se llevó a cabo a través de investigación y con ayuda del tutor Adrián Riesco Rodríguez.

Una vez familiarizado con los términos y características fundamentales del mercado de valores, se inició el desarrollo de la aplicación con los objetivos a cumplir en mente. Debido a que el mundo de las finanzas se caracteriza por una alta complejidad y sumado a la salida de un miembro del equipo en mitad del desarrollo, se decidió limitar el alcance de la aplicación a algo que fuera asumible por un solo desarrollador, pero a la vez proporcionando una funcionalidad completa y terminada.

Uno de los motivos por los cuales se decidió crear esta aplicación en versión web, fue porque la mayoría de personas pueden acceder a su contenido y usar sus servicios siendo solamente necesario un ordenador o un dispositivo móvil con navegador y acceso a Internet (multiplataforma). Además, como regla general, las aplicaciones web son muy sencillas de usar, pues cualquier persona con conocimientos básicos de informática puede trabajar con ellas. Por estos motivos, se prestó mucha atención a que la web cumpliera los Estándares Web W3C y que contara con un estándar de calidad de accesibilidad web.

7.2. Trabajo futuro

La aplicación web presentada es funcional y completamente operativa, y además sirve de base para algunas mejoras que pueden ser implementadas en futuros evolutivos de desarrollo:

- Contratar con Alpha Vantage un plan de pago para obtener una clave Premium. Se podrá realizar 120, 300, 600 o incluso 1200 llamadas por minuto y sin limitación diaria de llamadas.

- Implementar una comparación múltiple entre datos de empresas. Actualmente está implementada únicamente la comparación entre 2 empresas.

- Añadir más opciones de visualización de gráficas, es decir, más formatos de representación de los datos. Actualmente sólo se encuentra disponible la gráfica de líneas y de velas japonés (candlestick).

- Mejorar la web añadiendo nuevas paginas por las que navegar, sacando más provecho al componente de “migas de pan” ya creado.

- Mejorar la parte privada del usuario. Actualmente sólo se muestra la información privada en forma de tabla y en gráficas pequeñas, pero se podría añadir un historial de beneficios/perdidas por cada usuario.

- Permitir modificar la información de perfil del usuario (cambio de teléfono, dirección, etc.), así como añadir la posibilidad de subidas de imágenes de perfil.

- Distinguir a tipos de usuarios mediante el uso de roles. Cada rol estaría identificado con unos permisos y puede realizar acciones que afecten a ciertos usuarios.

Section 5

Conclusions and Future Work

7.1. Conclusions

Starting from the basis of wanting to create an application, which allows the user to assess the progress of the securities in which they have currently invested in the stock market, a study was conducted to learn everything related to the shares, price, volatility of the market of securities, stock indices, etc. In this learning process it was carried out through research and with the help of the tutor Adrián Riesco Rodríguez.

Once familiar with the fundamental terms and characteristics of the stock market, the development of the application began with the goals to be met in mind. Because the world of finance is characterized by high complexity and added to the departure of a team member in the middle of development, it was decided to limit the scope of the application to something that was acceptable by a single developer, but at the same time, providing a complete and a finished functionality.

One of the reasons why it was decided to create this application in a web version was because most people can access its content and use its services with only a computer or a mobile device with a browser and Internet access (multiplatform) being necessary. Also, as a rule of thumb, web applications are very easy to use, since anyone with basic computer skills can work with them. For these reasons, great attention was paid to ensuring that the website complied with the W3C Web Standards and that it had a quality standard for web accessibility.

7.2. Future work

The web application presented above is functional and can be used as the foundation for some improvements that can be implemented in future developments:

- Contract with Alpha Vantage a payment plan to obtain a Premium key. It will be possible to make 120, 300, 600 or even 1200 calls per minute and without daily call limitation.

- Implement a multiple comparison between company data. Currently, only the comparison between 2 companies is implemented.

- Add more graph display options, that is, more data representation formats. Currently only the Japanese candlestick chart and line graph is available.

- Improve the website by adding new pages to navigate through, taking more advantage of the "breadcrumbs" component already created.

- Improve the user's private part. Currently only private information is shown in table form and in small graphs, but a profit / loss history for each user could be added.

- Allow modifying the user's profile information (change of phone, address, etc.), as well as adding the possibility of uploading profile images.

- Distinguish types of users through the use of roles. Each role would be identified with permissions and can perform actions that affect certain users.

Bibliografía

Greg Shields. (2018). *Financial Statements: The Ultimate Guide to Financial Statements Analysis for Business Owners and Investors*, CreateSpace Independent Publishing Platform. CreateSpace Independent Publishing

«Retail Investor .org : How To Pick a Stock Investing Strategy That Suits You - Investor Education». Disponible en http://www.retailinvestor.org/next.html (en inglés). Consultado el 22 de febrero de 2020.

«Introduction To Dividends: Investing In Dividend Stocks». Investopedia. Disponible en https://www.investopedia.com/dividend-stocks-4689744 (en inglés). Consultado el 22 de febrero de 2020.

E., Kieso, Donald; D., Warfield, Terry (2007). *Intermediate accounting* (edición nº 12). p. 1320. [Consulta: 30 de noviembre de 2019].

Paul Pignataro. (2013). *Financial Modeling and Evaluation. A Practical Guide to Investmet Banking and Private Equity.* Wiley.

Nate Murray, Felipe Coury, Ari Lener, Carlos Taborda (2018). *Ng-book. The Complete Book of Angular.* CreateSpace Independent Publishing Platform (edicion n º 5). [Consulta: 15 de deciembre de 2019].

Cecilio Álvarez Caules (2013). Arquitectura Java: Blog sobre Java EE. Disponible en https://www.arquitecturajava.com/servicios-rest/. Consultado el 20 de febrero de 2020

Jorge Cano (2018). SG Buzz: Angular: Mucho más que un framework. Disponible en https://sg.com.mx/revista/56/angular. Consultado el 1 de marzo de 2018

Nacho Blanco (2018). Openwebminars: Qué es Angular: Origen y evolución. Disponible en https://openwebinars.net/blog/que-es-angular/. Consultado el 8 de marzo de 2020

Ángel Robledano (2018). Openwebminars: Qué es MySQL: Características y ventajas. Disponible en https://openwebinars.net/blog/que-es-mysql/. Consultado el 8 de marzo de 2020

Vidales Rubí, Leonel (2003). *Glosario de términos financieros.* Mexicali (México): Plaza y Valdés. pagina. 15. [Consulta: 30 de noviembre 2019].

José María Arribas Macho (2013). *Sociología del consumo e investigación de mercados. Una guía didáctica*. UNED.

Michael McMillan, Jerald E. Pinto, Wendy L. Pirie, Gerhard Van de Venter (2011). *Investments. Principles of Portfolio and Equity Analysis*. (edicion nº 1). John Wiley and Sons Ltd.

Kenneth R. Ferris, Kirk L. Tennant, Scott I. Jerris (1992). *How to Understand Financial Statements. A Nontechnical Guide for Financial Analysts, Managers and Executives*. CCH

Sunil Parameswaran (2011*). Fundamentals of Financial Instruments: An Introduction to Stocks, Bonds, Foreign Exchange, and Derivatives.* (edicion nº 1). Wiley.

Dr. George A. Manning, PhD, CFE, EA (2010). *Financial Investigation and Forensic Accounting* (edicion nº 3). Routledge

John D Finnerty (1985). *Corporate Financial Analysis: A Comprehensive Guide To Real-world Approaches For Financial Managers*. McGraw-Hill.

Mora Enguídanos, Araceli (2008). *Diccionario de Contabilidad, Auditoría y Control de Gestión*. Madrid: Ecobook. p. 12. [Consulta: 30 de noviembre 2015].

Colaboradores de Wikipedia, "*Angular (framework)*" Wikipedia, La enciclopedia libre, https://es.wikipedia.org/wiki/Angular_(framework)
(versión consultada: 25 de mayo de 2020).

Colaboradores de Wikipedia, "*Eclipse (software)*" Wikipedia, La enciclopedia libre, https://es.wikipedia.org/wiki/Eclipse_(software)
[versión consultada: 21 de marzo de 2020].

Colaboradores de Wikipedia, "*MySQL*" Wikipedia, La enciclopedia libre, https://es.wikipedia.org/wiki/MySQL
[versión consultada: 24 de mayo de 2020].

Colaboradores de Wikipedia, "*Servicio web*" Wikipedia, La enciclopedia libre, https://es.wikipedia.org/wiki/Servicio_web
[versión consultada: 22 de marzo de 2020].