UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO DE CIENCIAS E INGENIERÍAS

Aplicación web para la visualización y análisis del cambio en el precio global de combustibles fósiles y su efecto en la valoración de empresas del sector energético renovable y no renovable

César Alfonso Jiménez Morales

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación

Quito, 18 de Diciembre de 2021

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ
COLEGIO DE CIENCIAS E INGENIERÍAS

HOJA DE CALIFICACIÓN
DE TRABAJO DE TITULACIÓN

Aplicación web para la visualización y análisis del cambio en el precio global de combustibles fósiles y su efecto en la valoración de empresas del sector energético renovable y no renovable

César Alfonso Jiménez Morales

Nombre del profesor, Título académico	**Daniel Fellig Goldvechmiedt, MSc. in Computer Science**

Quito, 18 de Diciembre de 2021

© DERECHOS DE AUTOR

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en la Ley Orgánica de Educación Superior del Ecuador.

Nombres y apellidos: César Alfonso Jiménez Morales

Código: 00200784

Cédula de identidad: 1724547771

Lugar y fecha: Quito, 18 de Diciembre de 2021

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

La aplicación presentada en este documento busca ofrecer una plataforma web en la cual se pueda visualizar mediante gráficas y métricas estadísticas la afectación directa del precio de combustibles fósiles sobre distintas compañías del sector energético renovable y no renovable. Para este propósito, se utilizará la valoración en la bolsa de las compañías, en conjunto con el precio de distintos tipos de combustibles fósiles sobre un período de tiempo dado. Para la realización del sistema se diseñó una base de datos relacional utilizando el sistema de gestión MySQL además de un módulo que actualice esta información todos los días. Los datos almacenados en este sistema serán consumidos por un servidor diseñado a partir del framework para Python3, Flask. Este servidor atiende las peticiones del front-end en donde se implementa Vue.js para ofrecer al usuario una página web del tipo ‘single-page application’.

Palabras clave: Combustibles fósiles, Desarrollo web, Vue.js, Python, MySQL, Bolsa de valores, Energía, Compañías.

ABSTRACT

The web application presented in this document seeks to offer a platform in which by the use of plots and statistic metrics shows the price of different types of fossil fuels and its direct impact on various companies in the renewable and non-renewable energy sector. For this purpose, the market valuation of the companies and the 'future' prices of different types of fossil fuels over a time period will be used. For the development of the system, a relational database model was designed using the MySQL database management service as well as a module that updates the information on a daily basis. The data stored will be consumed by a server developed using the framework for Python3, Flask. This server attends petitions from the front-end application, implemented using Vue.js to give the user a 'single-page application' web page type.

Key words: Fossil fuels, Web development, Vue.js, Python, MySQL, Stock market, Energy, Companies.

TABLA DE CONTENIDO

Introducción 10
Objetivo General……..11
Objetivos Específicos……...11
Desarrollo del Tema.....13
Análisis de requerimientos……..13
Casos de uso……14
Diseño de la aplicación……19
Implementación……23
Caso de estudio (Prueba del sistema)...……34
Conclusiones.....38
Conlcusiones……38
Trabajo futuro……...38
Referencias bibliográficas 41

ÍNDICE DE TABLAS

Tabla 1. Empresas seleccionadas del sector no renovable…….20

Tabla 2. Empresas seleccionadas del sector renovable…….20

Tabla 3. Tecnologías utilizadas y sus respectivas versiones……..23

ÍNDICE DE FIGURAS

Figura 1. Diagrama de casos de uso para el actor "User"……14

Figura 2. Selección de fechas por medio de un calendario……15

Figura 3. Alerta por selección errónea del usuario……16

Figura 4. Selección de la compañía del sector Renovable……16

Figura 5. Selección del tipo de combustible fósil……17

Figura 6. Selección de predicción……18

Figura 7. Ventana de más información para Media y Desviación estándar……18

Figura 8. Esquema del sistema implementado para la aplicación……19

Figura 9. Esquema del funcionamiento de WSGI……22

Figura 10. Paradigma reactivo en Vue.js……23

Figura 11. Diagrama de entidad-relación para el modelo de datos de las compañías……24

Figura 12. Diagrama de entidad-relación para el modelo de datos de combustibles fósiles.25

Figura 13. Respuesta del API de AlphaVantage para Royal Dutch Shell……26

Figura 14. Código en Python3 para la conexión y recuperación de datos de la instancia MySQL……27

Figura 15. Código para aplicación Flask. respuesta en formato JSON……29

Figura 16. Esquema sobre el flujo de datos asincrónicos……30

Figura 17. Resultados de la aplicación web con West Texas Intermediate……36

Figura 18. Resultados de la aplicación web con Brent Crude Oil……36

Figura 19. Visualización de la aplicación en un dispositivo móvil……39

INTRODUCCIÓN

Los combustibles fósiles son la fuente de energía que ha sostenido e impulsado el desarrollo humano del último siglo. Estos son recursos finitos y conforme la ciencia avanza se van creando nuevas formas de obtener estos y otros tipos de fuentes de energía junto con más y mejores maneras de aprovecharlas. De modo que, desde 1950 hasta la actualidad se ha utilizado más energía (aprox. 22 zetajoules) que en los 11,700 años previos (aprox. 14.6 zetajoules). Debido en mayor medida a la combustión de combustibles fósiles extraídos alrededor del mundo (Syvitski et al., 2020).

Como lo explica (Anable et al., 2011), en la actualidad el uso de energía por parte de nuestra especie y especialmente en países desarrollados determina nuestra capacidad para movilizarnos, producir y consumir la variedad de productos disponibles en el mercado, mantener una temperatura cómoda en nuestros hogares, etc. Es así como se puede deducir, que nuestro estilo de vida es codependiente a la extracción de estos recursos naturales. Por lo tanto, la industria de la energía es posiblemente la más importante del mundo y en mayor o menor medida todas estas empresas que la componen dependen del valor y la disponibilidad de estos recursos explotados alrededor del mundo.

La industria de la energía se refiere al conjunto de empresas que se dedican a producir, comercializar y almacenar energía, sea esta proveniente de combustibles fósiles o de distintas fuentes, como es el caso de la energía renovable. En general, las empresas más importantes que se dedican a esto son gigantes conglomerados multinacionales que cuentan con sus activos financieros cotizándose en la bolsa de

valores. Es así como el precio de sus acciones está sujeto a la ley de oferta y demanda (Fuad et al., 2021).

Considerando la relevancia de los combustibles fósiles en el modelo de negocio de estas empresas, sea explotándolos, aprovechándolos u ofreciendo alternativas a los mismos. Tener una herramienta en donde se pueda visualizar el precio de los distintos combustibles fósiles junto con la valoración de empresas claves de distintos sectores de la industria energética sería útil para introducir al público en general cómo evolucionan y se interrelacionan estas piezas que mueven al mundo.

Objetivo General

Crear una web app en donde se pueda visualizar el precio de combustibles fósiles y su efecto en la valoración de empresas del sector energético en la bolsa de valores. Además de poder calcular una predicción de las valoraciones de estas empresas en función del precio de estos recursos naturales.

Objetivos Específicos

- Desarrollar un programa que a partir del uso de servicios ofrecidos por REST APIs y en conjunto con data-sets históricos, genere la base de datos necesaria para la presentación de la información en la web app.
- Establecer los modelos estadísticos más adecuados y relevantes tanto para el análisis, como para la predicción de las valoraciones e implementarlos en la aplicación.

- Presentar al usuario un diseño atractivo e intuitivo, orientado para que el público en general al cual le interese esta información, pueda utilizar la web app por su cuenta.

DESARROLLO DEL TEMA

Análisis de requerimientos

Con el objetivo de presentar una aplicación que cumpla con lo planteado en el apartado de introducción, es necesario presentar los requerimientos de la misma. Se proponen las siguientes funcionalidades para que el prototipo las cumpla:

- Capacidad de personalización por parte del usuario para escoger una compañía, combustible fósil y período de tiempo en específico sobre los cuales se realizará el análisis.
- Base de datos con toda la información financiera necesaria (precios de combustibles fósiles y acciones de compañías, con granularidad mínima de un día).
- Presentar gráficos del precio de las acciones y el tipo de combustible fósil sobre el tiempo. En el rango de fechas seleccionado por el usuario.
- Módulo que, dado un precio del combustible fósil ingresado por el usuario, realice una predicción del valor de las acciones de una compañía en base a un modelo estadístico.
- Cálculo de métricas clave para describir el comportamiento de los datos en el rango de tiempo seleccionado. Además, estas ayudarán al usuario a tener una idea de la fiabilidad que tiene el módulo de predicción sobre ese período de tiempo.

Con el objetivo de cumplir con estos requerimientos y brindar al usuario la experiencia más fluida posible, se propone una aplicación con un diseño del tipo ‘Dashboard Single-page Application’. De modo que el sitio web reaccione a cada interacción del usuario y en esa misma página vuelva a renderizar la información y gráficos automáticamente tras una nueva petición al ‘back-end’. Para este propósito se utilizaron varios lenguajes de programación de los cuales

destacan Python3 y JavaScript. A lo largo del documento se explicará más en detalle sobre el desarrollo del sistema y las diferentes tecnologías utilizadas.

Casos de uso

Los siguientes casos de uso presentados a continuación, describen la interacción de los usuarios con la aplicación web. Como se ha mencionado anteriormente, el sistema propuesto consiste en una ‘single-page application’ la cual no requiere un proceso de autenticación para su uso. El propósito de lo ya mencionado es brindar al usuario una experiencia fluida para “explorar” las distintas empresas y combustibles fósiles sin la barrera del registro.

La Figura 1 muestra el diagrama de casos de uso para el actor “User”. Esta entidad, representa a las personas que utilizarán el sitio web y por lo tanto demandan las distintas funcionalidades que el sistema debe ser capaz de cumplir, es decir: Seleccionar fechas, compañías y el combustible fósil específicos para el análisis. Además de pedir más información y hacer una predicción.

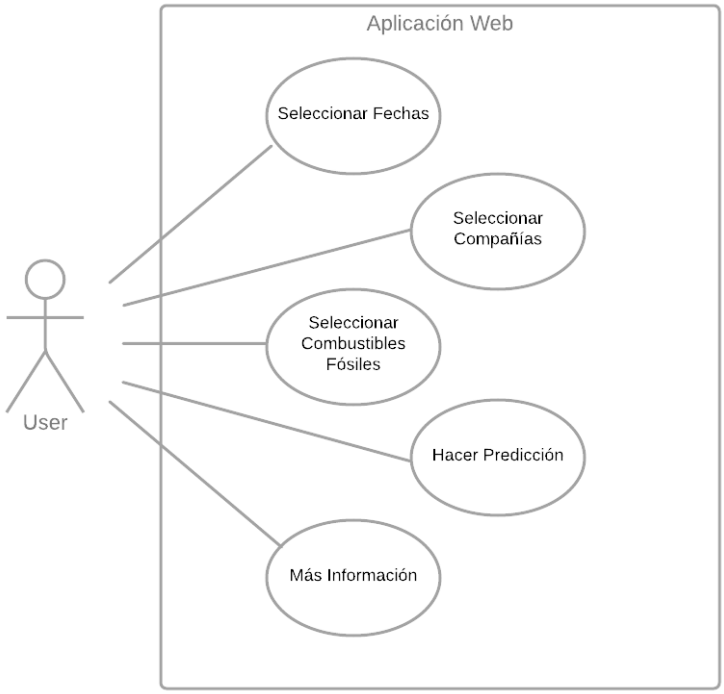


Figura 1. Diagrama de casos de uso para el actor "User".

Caso de Uso: *Seleccionar Fechas*

Curso Normal: Al utilizar la aplicación, el usuario tiene la capacidad de cambiar las fechas (desde y hasta) del rango de tiempo que se utilizará para mostrar los gráficos del valor de las acciones de la empresa y combustibles fósiles seleccionados, las métricas clave y el módulo de predicciones. En este caso el usuario escogerá un rango mínimo de un mes por medio de los dos componentes de 'calendario' como se puede observar en la Figura 2.

Figura 2. Selección de fechas por medio de un calendario.

Curso Alterno: Se puede dar el caso que un usuario seleccione una opción errónea, como por ejemplo escoger una fecha hasta la cual se realizará el análisis menor a la fecha desde la cual se realizará el análisis o un rango menor a 30 días. Entonces, en lugar de recibir esta información, la aplicación web despliega una alerta describiendo el error (Figura 3) y con los datos restablecidos a su estado original.

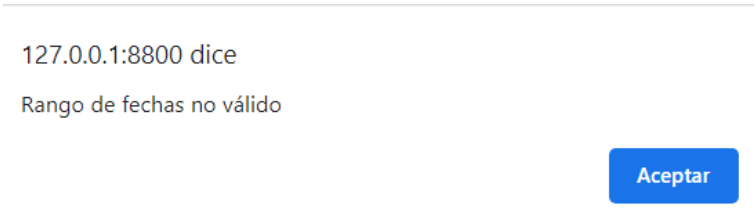


Figura 3. Alerta por selección errónea del usuario.

Caso de Uso: *Seleccionar Compañías*

Curso Normal: Para poder seleccionar una empresa en particular, la aplicación ofrece una lista de todas las compañías de energía presentes en la base de datos. Estas, están categorizadas de acuerdo a su área específica dentro de la industria energética. Cuando se hace clic sobre cualquiera de estas empresas los gráficos, métricas y el módulo de predicción se actualizarán acorde a lo que se ha seleccionado como se puede observar en la Figura 4.

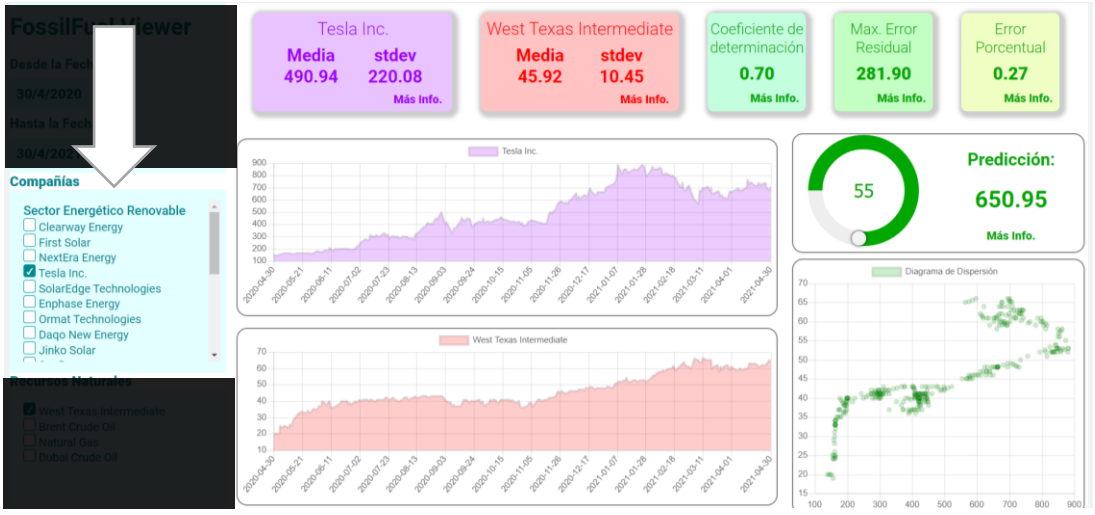


Figura 4. Selección de una compañía del sector Renovable.

Curso Alterno: Cabe la posibilidad de seleccionar una compañía cuya información en la base de datos no se encuentre disponible para el rango de tiempo seleccionado. Esto puede suceder especialmente para rangos mayores a diez años, en compañías que son relativamente nuevas en la bolsa de valores. Entonces, la aplicación web desplegará una alerta (al igual que

en la Figura 3) informando al usuario de lo ocurrido y cambiará automáticamente el rango de fechas a uno válido.

Caso de Uso: *Seleccionar Combustibles Fósiles*

Curso Normal: La aplicación web brinda la opción de escoger específicamente el combustible fósil sobre el cual se realizará el análisis. En este caso, al igual que en 'Seleccionar Compañías', la aplicación cuenta con una lista en la barra izquierda para seleccionar cualquiera de las opciones disponibles de combustibles fósiles. La Figura 5 muestra esta interacción en la aplicación.

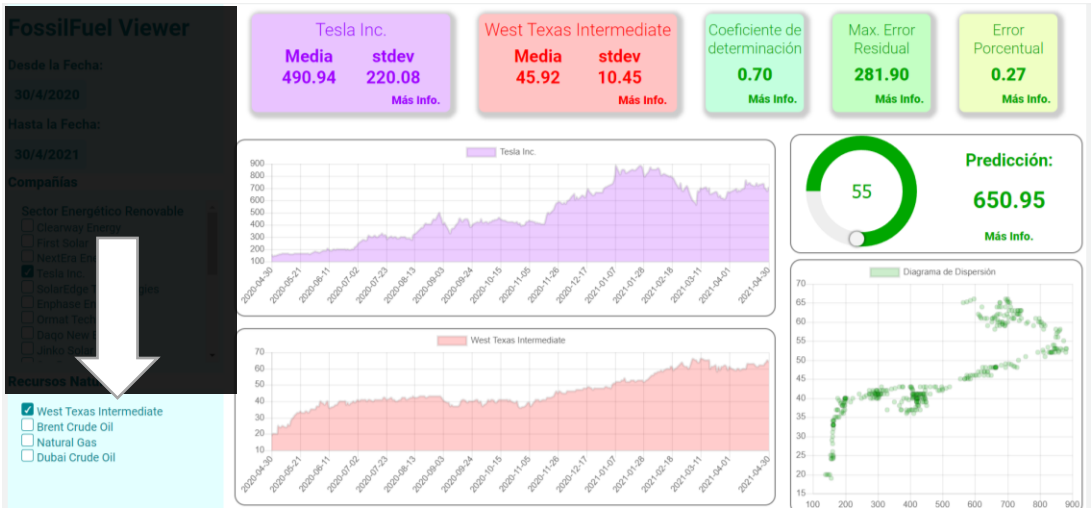


Figura 5. Selección del tipo de combustible fósil.

Caso de Uso: *Hacer Predicción*

Curso Normal: Cuando se han seleccionado atributos válidos y la página muestra los gráficos y métricas correspondientes, este módulo está listo para ser utilizado. El usuario puede seleccionar un valor de combustible fósil en el 'slider' circular a la izquierda del contenedor de predicción que se muestra en la Figura 6. Entonces, el valor de predicción para el valor de las acciones de la compañía seleccionada será actualizado.

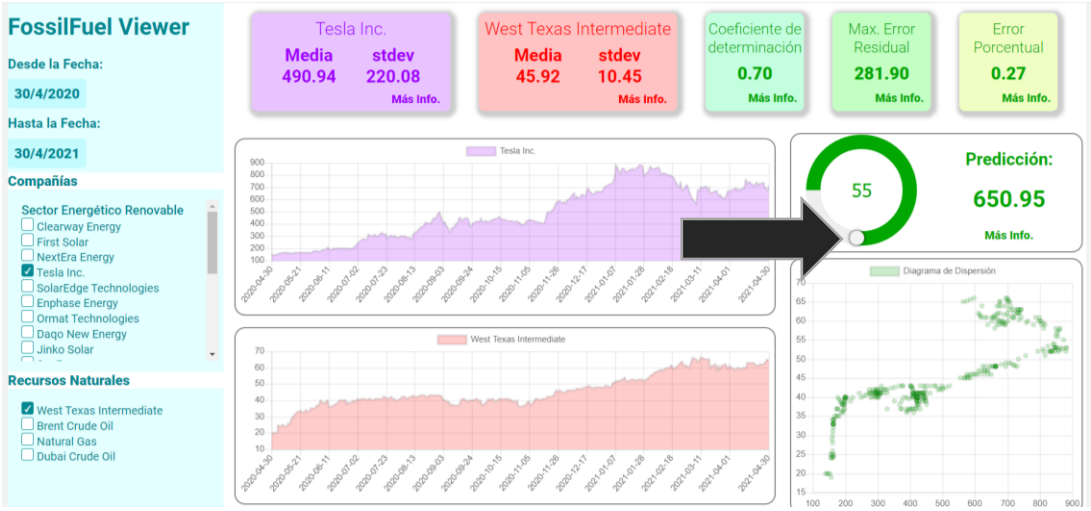


Figura 6. Selección de predicción.

Caso de Uso: *Más Información*

Curso Normal: el usuario selecciona esta opción dando clic sobre el botón de más información presente en cualquiera de las métricas o en el módulo de predicción. Se desprenderá una ventana emergente que contiene la descripción de la métrica/predicción seleccionada. De esta manera, la aplicación presenta una ayuda para que los usuarios que no estén familiarizados con estos conceptos estadísticos puedan interpretar los datos mostrados sin problemas. La figura 7 muestra un ejemplo de lo mencionado, para la media y desviación estándar.

Figura 7. Ventana de más información para Media y Desviación estándar

Diseño de la aplicación

Como se muestra en la Figura 8 la aplicación está diseñada de modo que existen 3 módulos independientes entre sí, los cuales son: 'Vue.js application', 'Flask Back-end' y 'MySQL database'. El flujo resumido de este sistema consiste en que el usuario interacciona con la aplicación Vue.js (Sitio web), automáticamente se genera una petición (URL con atributos específicos) a la aplicación de Flask. Una vez recibida la petición, se valida y mediante 'queries' hacia la base de datos, se extrae la información, se procesa y estos datos son enviados a la aplicación cliente en formato JSON.

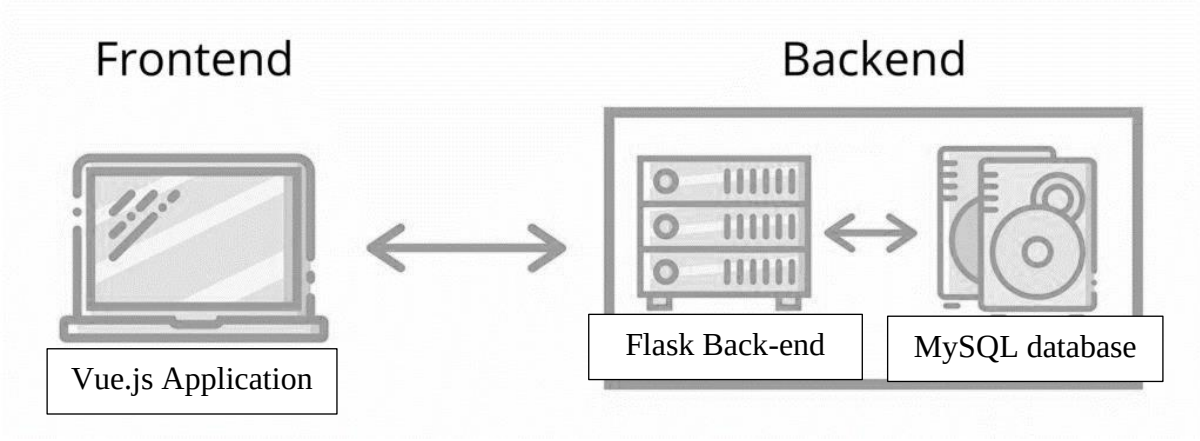


Figura 8. Esquema del sistema implementado para la aplicación.

La base sobre la cual se sustenta el funcionamiento de la aplicación son los datos que se encuentran guardados en la instancia en línea de MySQL con dos esquemas (uno para combustibles fósiles y otro para compañías). Para combustibles fósiles se ha planteado tener 4 tipos como son WTI, BRENT, DUBAI y Gas Natural. Para el segundo esquema se seleccionaron compañías importantes del sector energético renovable y no renovable cuyas acciones se cotizan en NYSE (New York Stock Exchange), NASDAQ (National Association

of Securities Dealers Automated Quotations) y LON (London Stock Exchange). Esta selección se muestra en la Tabla 1 y Tabla 2.

Nombre de la compañía.	**Ticker**
ExxonMobil	XOM
Schlumberger	SLB
Chevron Corporation	CVX
Phillips 66	PSX
Enterprise Products	EPD
Equinor	EQNR
British Petroleum	BP.LON
Total Energies	TTE
PetroChina	PTR
Royal Dutch Shell	RDS-A
ExxonMobil	XOM

Tabla 1. Empresas seleccionadas del sector no renovable.

Nombre de la compañía.	**Ticker**
Tesla Inc.	TSLA
Clearway Energy	CWEN/CWEN.A
First Solar	FSLR

NextEra Energy	NEE
SolarEdge Technologies	SEDG
Enphase Energy	ENPH
Ormat Technologies	ORA
Daqo New Energy	DQ
Jinko Solar	JKS
SunPower	SPWR
Canadian Solar	CSIQ

Tabla 2. Empresas seleccionadas del sector renovable.

Una vez terminado el modelo de datos, se desarrolló una aplicación utilizando el framework minimalista para desarrollo de aplicaciones web Flask, escrito en el lenguaje de programación Python3. El funcionamiento de esta aplicación consiste en brindar un servicio web del tipo REST API que, a través de un rango de fechas, una compañía y un combustible fósil dado, sea capaz de producir los datos listos para presentarse en la aplicación. Flask se basa en una especificación de WSGI (Web Server Gateway Interface). Esto permite a sus aplicaciones manejar miles de peticiones y que WSGI las resuelva y envíe la respuesta en forma de HTML al cliente que la solicitó, escalando de manera eficiente el tráfico web. La Figura 9 muestra en un esquema el manejo de peticiones y respuestas desde un cliente hacia el WSGI server.

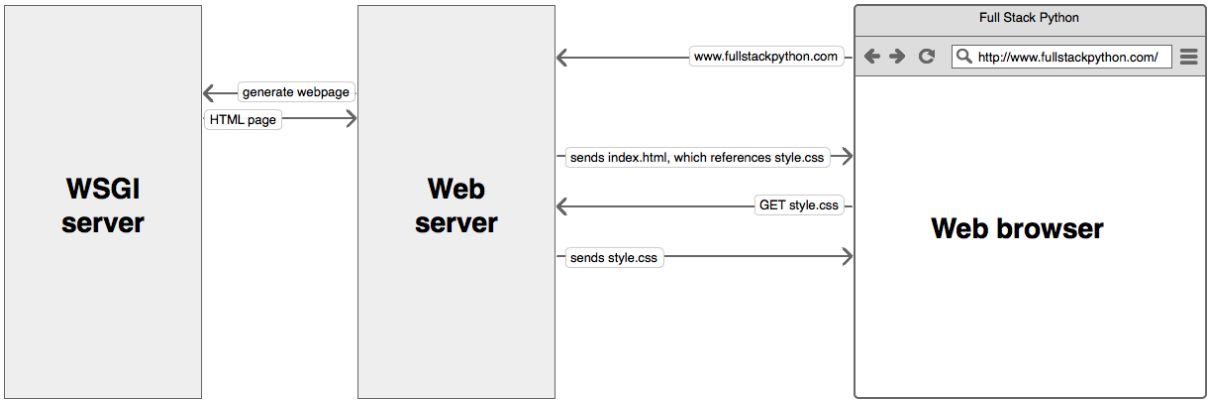


Figura 9. Esquema del funcionamiento de WSGI.

Nota. Adaptado de WSGI Servers [Ilustración], por Matt Makai, 2021, Full Stack Python (https://www.fullstackpython.com/wsgi-servers.html)

Con la base de datos implementada y la aplicación web del tipo REST API disponible. Es necesario una aplicación cliente que pueda consumir estos servicios y presentar la información a los usuarios. Como se había planteado anteriormente el diseño de la misma es del tipo ‘single-page application’ y por lo tanto es necesario un framework como Vue.js que permita manejar las interacciones de los usuarios de manera asíncrona sin la necesidad de redirección hacia otra página e incluso de recargar la misma. El funcionamiento de esta tecnología se basa en el paradigma de programación reactiva concerniente a la propagación de cambios a partir de distintos flujos de datos (ver Figura 10). Estos cambios luego son actualizados en la interfaz gráfica de manera automática y sin actualizar manualmente (clic en refrescar dentro del browser).

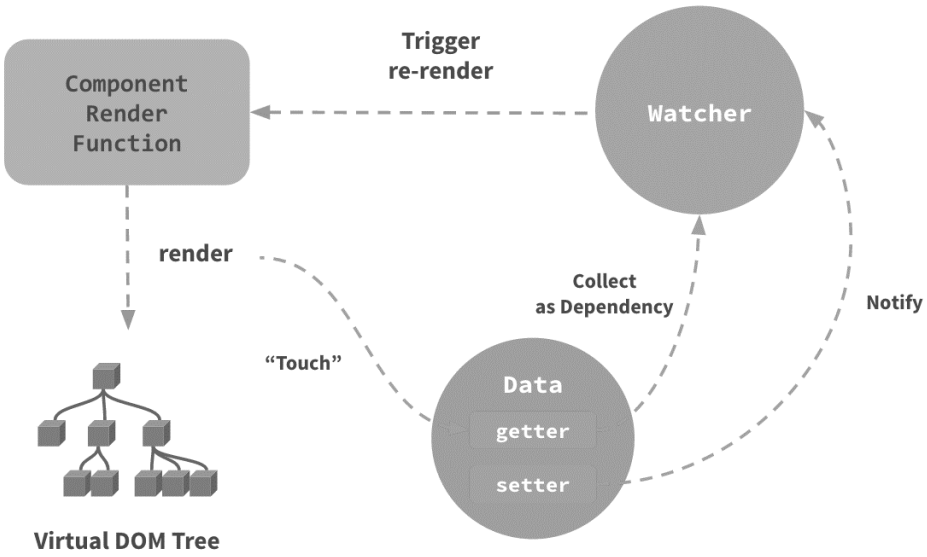


Figura 10. Paradigma reactivo en Vue.js.

Nota. Adaptado de 'Reactivity' [Ilustración], por Vuejs.org, s.f., Guía Vue.js (https://es.vuejs.org/v2/guide/reactivity.html)

Implementación

Herramientas

Para el desarrollo del sistema se utilizaron los ambientes de desarrollo PyCharm y Visual Studio Code, además de la herramienta para el diseño de bases de datos MySQL Workbench. A continuación, se presenta la Tabla 3 que muestra las principales tecnologías utilizadas y su versión.

Tecnología	**Versión**
Python	3.8
Vue.js	2
SciKit Learn	0.24.2

MySQL	8.0.26
Flask	2.0.1
Chart.js	2.9.4

Tabla 3. Tecnologías utilizadas y sus respectivas versiones.

Base de datos MySQL

Para la creación del modelo de datos se utilizó el sistema de administración de bases de datos MySQL. En este proyecto se planteó un esquema relacional de manera que la información almacenada estaría guardada, organizada y sea fácil y rápido obtener la información exacta necesaria de acuerdo a la petición de un usuario. Las Figuras 11 y 12 muestran los diagramas de entidad-relación o 'DER' de las dos bases de datos utilizadas para este proyecto. Se observan las distintas tablas y sus atributos de acuerdo al modelo relacional, de modo que toda la información necesaria para el funcionamiento de la aplicación se encuentra organizada utilizando la estructura de relaciones de los diagramas en cuestión.

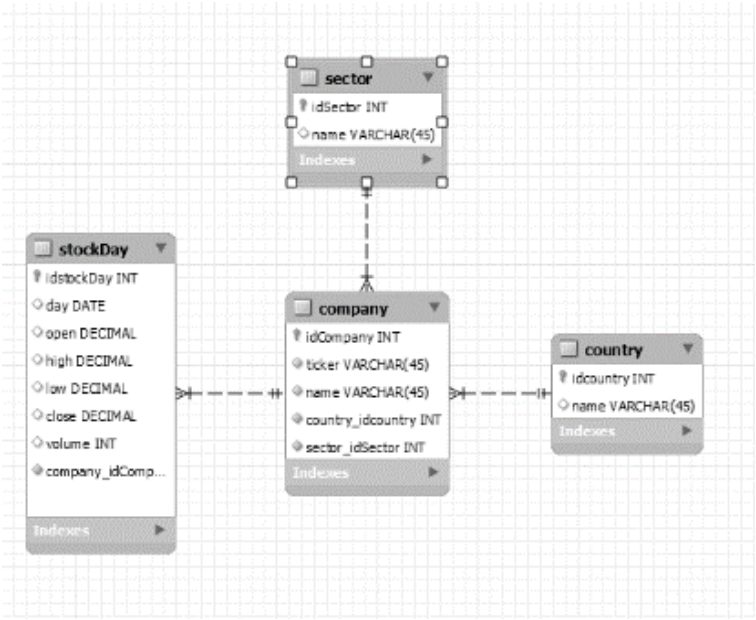


Figura 11. Diagrama de entidad-relación para el modelo de datos de las compañías.

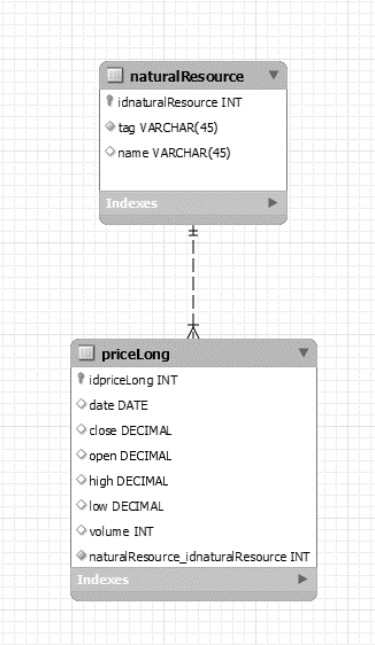


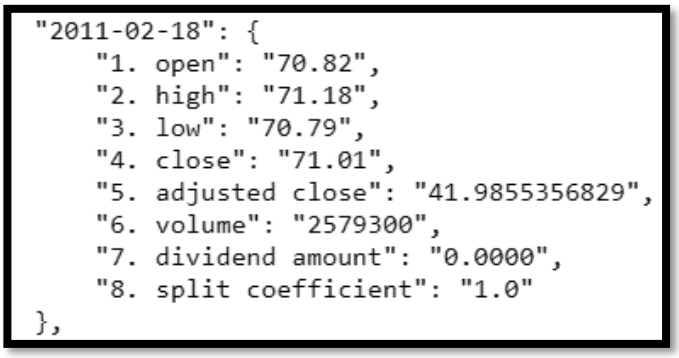
Figura 12. Diagrama de entidad-relación para el modelo de datos de combustibles fósiles.

Como se puede observar a partir de los 'DER' de las Figuras 9 y 10, el modelo de datos no es muy complejo y se almacena únicamente la información esencial en base a la cual se realizará el procesamiento necesario a partir de la petición del usuario. Para este proyecto se utilizaron 4 tipos de combustibles fósiles, WTI, BRENT, DUBAI y Gas natural.

El data-set utilizado para combustibles fósiles es el historial de precios futuros para recursos naturales de la plataforma en línea 'Investing.com'. Estos son datos denominados 'futures prices' y son derivados del precio en tiempo real 'spot prices' del mercado global. Aunque apuntan a ser lo más precisos posibles no dejan de ser referenciales de la actual cotización de mercado (compra inmediata). Sin embargo, para el propósito de este proyecto y basados en la granularidad mínima de un día, la información obtenida es utilizable.

En cuanto a los datos del precio de compañías, estos fueron obtenidos y son actualizados diariamente a través de la API pública AlphaVantage. A partir de esta se extraen los valores financieros de las compañías, pertenecientes a distintos mercados de valores.

Es importante recalcar que el valor utilizado para el propósito de esta aplicación es el ‘Adjusted closing price’ de cada día. Como lo explica (Hartill, 2021), el precio de ajustado de una acción toma en cuenta el pago de dividendos y los ‘stock splits’ (cuando se dividen las acciones con el objetivo de hacer más accesible la cotización de sus activos). Entonces, este valor ajustado nos brinda una mejor referencia para evaluar acciones sobre el tiempo. ‘Closing price’ y ‘Adjusted closing price’ NO siempre son iguales (ver Figura 13).



```
"2011-02-18": {
    "1. open": "70.82",
    "2. high": "71.18",
    "3. low": "70.79",
    "4. close": "71.01",
    "5. adjusted close": "41.9855356829",
    "6. volume": "2579300",
    "7. dividend amount": "0.0000",
    "8. split coefficient": "1.0"
},
```


Figura 13. Respuesta del API de AlphaVantage para Royal Dutch Shell

Aplicación Flask para el procesamiento de datos y cálculos estadísticos

En relación a lo explicado anteriormente, el diseño de la aplicación cuenta con un módulo intermediario entre la base de datos y la aplicación cliente. Este componente se trata en un programa desarrollado utilizando la tecnología de Flask y el lenguaje Python3. La tarea del mismo consiste en recibir las peticiones desde una aplicación cliente (enviadas utilizando el método GET del protocolo HTTP). Una vez listos los distintos atributos, la aplicación envía una respuesta en formato JSON conteniendo todos los datos necesarios para que la aplicación cliente renderice el nuevo estado.

Los cuatro atributos que necesita recibir la aplicación son: fecha inicial, fecha final, nombre de la compañía y nombre de un tipo de combustible fósil. Una vez recibidos los datos, una cadena de procesos comienza hasta formular la respuesta al cliente:

- Lo primero que hace la aplicación es solicitar mediante ‘queries’ a la base de datos los precios diarios de las compañías (‘adjusted close’) y de los combustibles fósiles (‘close’) desde las fechas de inicio hasta la fecha final. La figura muestra una captura del proceso para conectarse con la instancia de MySQL desde una aplicación Python y recibir el resultado en cuestión. La Figura 14 muestra una función que dado un ‘query’ (en el lenguaje específico de MySQL) y el nombre de la base de datos, retorna la respuesta como un diccionario de Python.

```
def getFromDatabase(query,dataB):
    result = None
    try:
        connectionObj=mysql.connector.connect(host='database-1.[redacted].us-east-2.rds.amazonaws.com',
                                              database=dataB,user=[redacted] password=[redacted])
        if connectionObj.is_connected():
            mySql_select_query = query
            cursor = connectionObj.cursor()
            cursor.execute(mySql_select_query)
            result = cursor.fetchall()
    except Error as e:
        print("Error while connecting to MySQL", e)
    finally:
        if connectionObj.is_connected():
            cursor.close()
            connectionObj.close()
        return result
```

Figura 14. Código en Python3 para la conexión y recuperación de datos de la instancia MySQL.

- Una vez recibidos los datos, es necesario procesarlos antes de aplicar cualquier modelo estadístico. En primer lugar, se verifica que todos los días en el rango de fechas seleccionado tengan un valor y de lo contrario se interpola para completar los datos faltantes, obteniendo dos listas (compañía y combustible fósil) con ‘n’ elementos, siendo ‘n’ la cantidad de días en el rango de tiempo.
- Con los datos procesados, se obtienen las dos listas completas de valores financieros para la línea del tiempo y estos son ingresadas a un diccionario de

Python. Es importante que cada elemento que sea ingresado al diccionario tenga una llave del tipo String y en caso de las listas que todos los valores sean del mismo tipo (float). Esto debido a que al final del proceso, este diccionario será convertido al formato JSON para ser transmitido como una respuesta HTTP.

- Una vez que ambas listas se encuentran en el diccionario, a partir de estas se calculan las métricas de promedio y desviación estándar. Estos resultados también son ingresados al mismo diccionario.
- Utilizando funciones de SCIKIT-LEARN se realiza una regresión lineal tomando como variable independiente el precio del tipo de combustible fósil seleccionado y como variable dependiente el precio de las acciones de la compañía. Este método de regresión busca encontrar un modelo lineal de la siguiente forma:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

Siendo Y la variable referente al precio de las acciones de la compañía y X el precio del combustible fósil. El término ϵ se refiere al error, mientras que β_0 y β_1 son los coeficientes que buscamos encontrar, tal que se minimice la suma de cuadrados de los residuos.

- Una vez obtenidos $\beta_0 + \beta_1$ se ingresan ambos coeficientes en el mismo diccionario donde están el resto de datos para ser enviados en la respuesta HTML. Con estos coeficientes, se puede realizar (desde la aplicación cliente) una predicción de la forma:

$$\hat{y} = \beta_0 + \beta_1 x + \epsilon$$

En donde $\hat{y}$ es una predicción del valor de las acciones de una compañía para cierto valor x del precio de un combustible fósil.

- Para finalizar la tarea de la aplicación Flask, se calculan las métricas: Coeficiente de Determinación, Máximo Error Residual y el Error Porcentual. Estos valores, también son ingresados al diccionario para ser presentados en la aplicación cliente.
- Una vez se completan estos pasos, la función “fetchData” del módulo análisis de la aplicación Flask retorna el diccionario. Este es enviado como respuesta al cliente tras ser reformateado como un objeto JSON, tal como se muestra en la Figura 15.

```
@app.route("/api/v1.0/petition")
def petition():
    req = analisis.fetchData(request.args.get('firstDate'),request.args.get('secondDate'),
                             request.args.get('name'),request.args.get('nameNR'))
    return jsonify(req)
```

Figura 15. Código para aplicación Flask. respuesta en formato JSON

Front-end, aplicación Vue.js

Para este apartado del sistema, se decidió utilizar el marco para desarrollo de aplicaciones web Vue.js. Esto debido a la facilidad que ofrece a la hora de implementar páginas web dinámicas y aplicar el paradigma de programación reactiva orientado a responder la interacción asincrónica que existe entre el sitio y los usuarios. Este marco o ‘framework’ implementa distintas tecnologías para ofrecer a los desarrolladores la funcionalidad que se necesita para una ‘Single-page application’. Se destacan por sobre las demás, las siguientes características de una aplicación Vue.js:

- Virtual DOM: Document Object Model (DOM) se refiere a una interface que estructura el código XML o HTML de forma que cada componente es representado como un objeto en una estructura de árbol. Virtual DOM consiste en crear una réplica del DOM

real con la intención de ser modificada. Una vez se han finalizado los cambios, la estructura de esta réplica es actualizada al DOM real. De esta manera se optimiza el costo de actualizar la página y es posible hacerlo más seguido sin sacrificar fluidez para el usuario.

- Components: El marco permite la creación de componentes reutilizables con la capacidad de poseer una plantilla HTML, estructura lógica (datos y métodos) y diseño CSS propios. Modificando los datos de inicialización es posible reutilizar el mismo componente en distintas partes del sitio web.
- V-Bind: Vue proporciona la directiva v-bind. Esta permite manipular o asignar valores a los atributos HTML. Entonces, si en la estructura lógica del sitio web (instancia de vue.js) se cambian las variables, los componentes HTML que están enlazados a estas se re-renderizarán automáticamente en función de los cambios.
- Watchers: En Vue.js se utilizan 'Watchers' con la intención de reaccionar de una manera personalizada al cambio en alguna variable. Se declaran métodos, los cuales se ejecutarán automáticamente ante cualquier cambio en la variable definida. Esta funcionalidad es importante para implementar el paradigma de programación reactiva, debido a que un evento de este tipo puede desencadenar una serie de flujos de datos (ver Figura 16) que no se puede representar solamente utilizando la directiva v-bind.

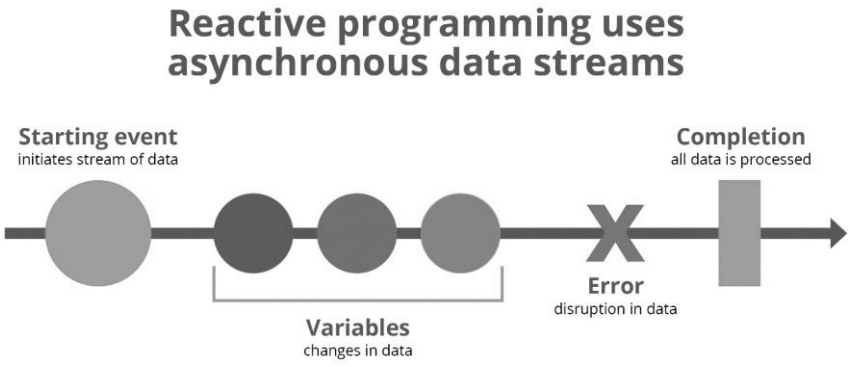


Figura 16. Esquema sobre el flujo de datos asincrónicos.

Nota. Adaptado de 'Reactive programming' [Ilustración], por Aayushi Sahu, 2019, What is Reactive Programming? (https://www.westagilelabs.com/blog/five-interesting-facts-about-reactive-programming-frameworks/)

En términos generales, el funcionamiento del front-end es el siguiente:

- Existen 5 medios por los cuales el usuario puede generar un 'Starting Event' (Ver Figura 16) que desencadene un nuevo flujo de datos hasta que se vuelva a renderizar alguno o varios elementos de la interfaz.
- En la barra izquierda de la aplicación, el usuario puede modificar la fecha de inicio, la fecha final, la empresa y el combustible fósil para el análisis. Cualquier cambio en alguno de estos valores desencadenará la siguiente secuencia de eventos:
    - Se verifica que las fechas sean válidas: rango (mínimo 2005, máximo 2021), lógica (que la fecha inicial sea anterior a la fecha final).
    - Para la empresa seleccionada, se valida que el sistema cuente con datos para el rango de fechas. Aplica especialmente para algunas compañías del sector renovable que tienen pocos años en el mercado de valores.
    - Con los datos validados se construye una cadena de caracteres con la estructura del tipo URL y los atributos necesarios para solicitar información al back-end como lo muestra la figura 13.
    - Con la URL se realiza una petición al back-end, quien devuelve en formato JSON todos los datos necesarios para actualizar la interfaz del front-end.
    - En este momento, se remplazan los valores de las variables asociadas a los 3 gráficos. Los nuevos datos que son ingresados son los que fueron recuperados tras consumir el servicio del back-end (ya procesados, sin

faltantes). Entonces, se actualizan las dos líneas del tiempo y el diagrama de dispersión.

- Los valores de métricas clave también son recibidos con el objeto JSON. Y son por lo tanto actualizados en sus componentes dentro de la interfaz.
- El módulo de predicción (slider) cambia su valor mínimo y máximo de acuerdo a los datos recibidos del precio de combustible fósil. Además, los coeficientes necesarios para el proceso de regresión no se muestran en la interfaz, pero ya están cargados como atributos para el método que realiza este cálculo.
- Todos estos pasos suceden automáticamente y una vez la nueva interfaz ha sido renderizada, el usuario puede volver a modificar los parámetros del análisis y este proceso se repetirá.

Elementos que conforman el análisis en la interfaz Single-page:

Los elementos ubicados en el bloque derecho de la aplicación son todos parte del análisis que realiza la aplicación en función de la entrada de datos especificada por el usuario. Estos se dividen en 3 tipos distintos de componentes los cuales se detallarán a continuación:

- Gráficas: se utilizan dos tipos de gráficas, un gráfico lineal y un diagrama de dispersión. Aunque, son dos tipos distintos de componentes, el concepto es el mismo para ambos. Se define un componente del tipo Vue.js que herede del componente específico de cada gráfico de vue-chartjs. Chart.js es una librería para la renderización de distintos tipos de gráficos en páginas web, vue-chartjs es un 'wrapper' de esta tecnología que se utiliza para extender la misma funcionalidad hacia componentes Vue.js.
- Métricas: En el caso de las métricas es más sencillo, debido a que lo único dinámico que presenta cada uno de los contenedores individuales es el valor calculado por el

back-end. Este valor es una variable de la instancia Vue.js principal la cual se ha hecho 'bind' con el componente HTML de cada métrica. Entonces, cualquier cambio detectado en esta variable se renderizará en la página nuevamente.

- Módulo de Predicción: Este componente cuenta con dos elementos, alineados horizontalmente sobre un contenedor. En primer lugar, está un 'slider' circular cuyo valor representa un posible precio del petróleo y puede ser modificado por el usuario. El número a la derecha sería la predicción resultante del método de mínimos cuadrados.

Despliegue en Amazon Web Services (AWS):

Amazon RDS: Este servicio que ofrece AWS permite montar una instancia de MySQL a la cual se puede acceder por medio de una dirección en línea. El desarrollo del sistema comenzó con la creación de la base de datos. Análogamente, el primer paso para realizar el despliegue al sistema en línea fue subir la misma en el servicio RDS. En primer lugar, se creó la instancia MySQL por medio del cliente web de AWS. Con la instancia creada se conectó a ella a través de MySQL Workbench utilizando la dirección web junto con el usuario y contraseña de administrador designados. En este momento, se generó un respaldo de la base de datos local hacia un script (.sql). Se ejecutó este script en la instancia de AWS desde el Workbench y de este modo se migraron todos los esquemas y datos para ser accesibles en línea.

Amazon Elastic Beanstalk: Para el back-end se utilizó este servicio de AWS que permite implementar y escalar aplicaciones web desarrolladas en Python, Docker y demás. La tecnología subyacente de la instancia de Elastic Beanstalk creada y configurada para este proyecto consta de una máquina virtual Linux y un sistema de almacenamiento asociado a esta. Es importante mencionar que el ambiente debe contar con todos los módulos y librerías necesarios para el funcionamiento del servidor Flask. Para este propósito, se subió junto con el código de la aplicación, un archivo con el detalle sobre los módulos y sus versiones para que

puedan ser instalados por el controlador del ambiente al crear la instancia de Elastic Beanstalk. Una vez montada la aplicación, es posible hacer las peticiones de datos a una URL pública en lugar de una local, como se hacía durante el desarrollo.

Amazon S3: El último componente de la aplicación web es el cliente (Vue application). AWS también cuenta con un servicio de hosting para aplicaciones web estáticas. En este caso el término estático se refiere a páginas HTML simples o como en este caso, con scripts pero que corren del lado del cliente y no necesitan procesamiento del lado del servidor. Entonces, este servicio es el que se utilizará a futuro para montar en línea el front-end.

Caso de Estudio (Prueba del sistema)

Una vez finalizados los componentes del sistema, se consideran muchas pruebas posibles. Cada diferente combinación de atributos resultará en una distinta petición de datos al back-end y por lo tanto un distinto análisis. Entonces, tomando en cuenta la variabilidad del problema y las prácticamente infinitas combinaciones de compañías, tipos de combustibles y rangos de fechas. Se decide escoger de manera arbitraria el siguiente caso, para observar el comportamiento de una empresa y distintos combustibles fósiles en un período de un año en donde se volvía a estabilizar la economía del petróleo que en abril de 2020 sufrió el precio más bajo del último siglo debido a la pandemia de Covid-19.

Caso de estudio: *Tesla, 30 de Abril de 2020 a 30 de Abril de 2021*

Para esta prueba, se selecciona la empresa Tesla en el rango de fechas mencionado. En primer lugar, con el tipo combustible fósil West Texas Intermediate y posteriormente con Brent Crude Oil. Para ambos casos, la aplicación en pocos segundos cumple con la funcionalidad esperada. Todos los gráficos son actualizados de acuerdo a la selección especificada, se muestran el valor de las acciones de la compañía en función del tiempo. También, el precio del combustible fósil en función del tiempo. Además, un diagrama de dispersión de las dos

variables. Con estos 3 gráficos es posible observar el cambio en el precio del combustible fósil en paralelo al cambio que tuvo Tesla en su valoración en bolsa y su correlación en función de la forma y distribución de datos mostrados por el diagrama de dispersión.

Las Figuras 15 y 16 muestran los resultados de la aplicación para los dos casos descritos anteriormente. En la parte superior de los resultados se pueden observar métricas clave para analizar el comportamiento de ambas variables y las últimas 3 que muestran la calidad que tiene el modelo de regresión en donde el factor determinante es qué tan relacionadas linealmente se encuentran estas dos variables sobre el tiempo. Evidentemente, no se puede garantizar que un usuario 'cualquiera' posea el conocimiento para interpretar estas variables. Por lo tanto, se pueden utilizar las opciones de más información y visualizar una explicación simple de cada una de las métricas presentadas.

Para ambos casos, el coeficiente de determinación muestra un resultado de 0.70. Este, es un valor bastante alto considerando que para que este sea 1 el modelo debería mostrar una correlación lineal perfecta (sea positiva o negativa). Entonces, con estos resultados en mente, un usuario cualquiera puede hacer pruebas con el módulo predictivo, cambiar las fechas, la compañía o el tipo de combustible fósil. En otras palabras, manipular a su gusto la aplicación e ir sacando conclusiones propias sobre el efecto directo que tienen distintos tipos de combustibles fósiles sobre la valoración en bolsa de las compañías de la industria energética.

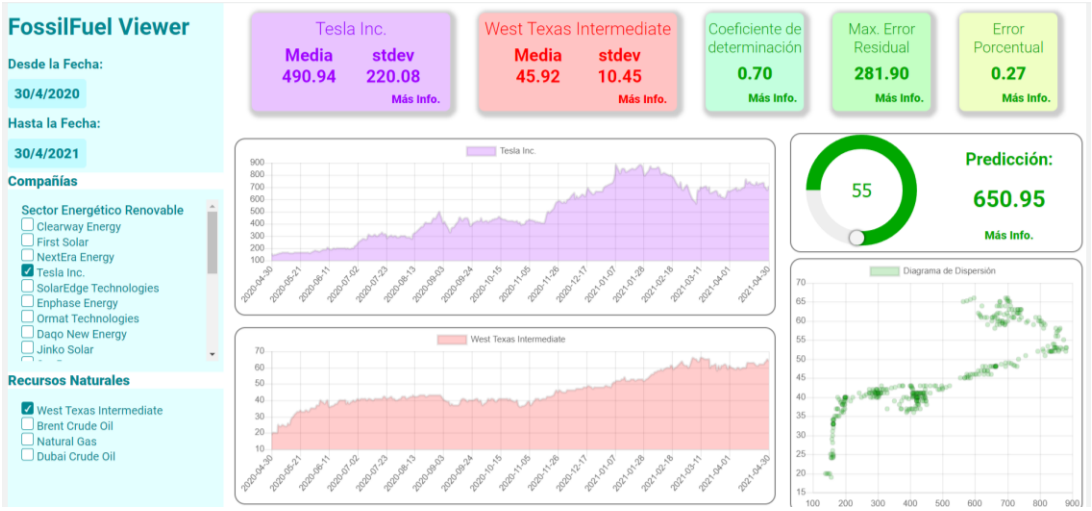


Figura 17. Resultados de la aplicación web con West Texas Intermediate.

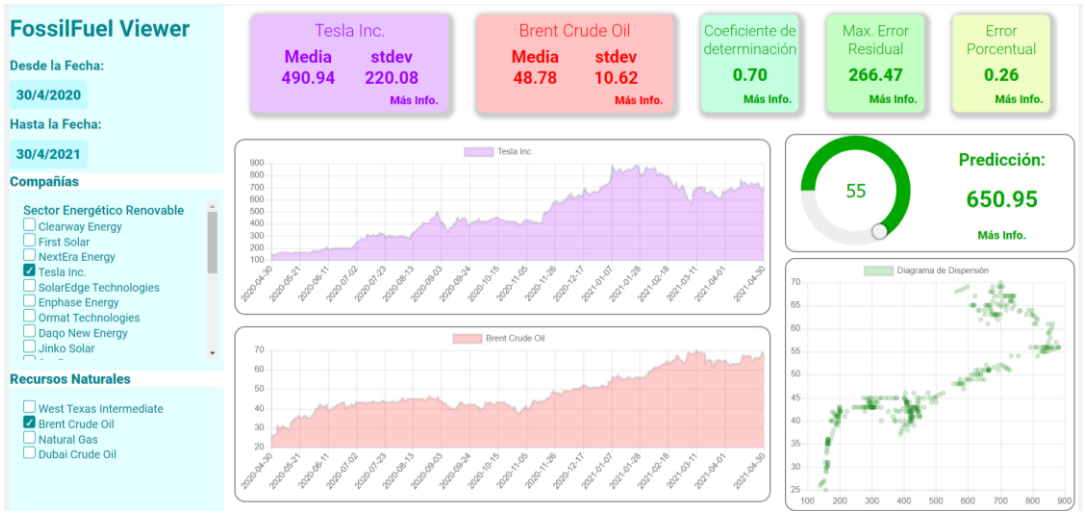


Figura 18. Resultados de la aplicación web con Brent Crude Oil.

Problemas conocidos:

Es posible que un usuario encuentre, para un período reciente, que el valor de cierta compañía está altamente relacionado a un tipo de combustibles fósiles (coeficiente de determinación cercano a 1.0). Entonces, esto puede incentivar a que se utilice la aplicación como una herramienta de inversión. Cabe la posibilidad que llegue a dar buenos resultados en casos especiales (Ej. compañía que consigue un contrato muy grande, en donde el precio del petróleo es determinante para sus utilidades) o incluso por coincidencia. Sin embargo, la

aplicación NO está diseñada para eso. Al menos, no para que sea la única referencia antes de tomar una decisión de financiera. Para mitigar este riesgo hacia los usuarios, se tiene una nota que menciona esto en el apartado de más información del módulo de predicción.

CONCLUSIONES

Conclusiones:

- Tras la prueba presentada con Tesla sobre el período de un año y observando los resultados mostrados a través del navegador. Se puede concluir que la aplicación cumple con los requerimientos planteados al inicio del proyecto.
- El uso de varias tecnologías, entre las cuales destacan Flask, Vue.js y MySQL fue satisfactorio. Los distintos componentes del sistema se conectan adecuadamente para brindar la funcionalidad esperada al usuario. Además, se realizó un despliegue exitoso, haciendo uso de los servicios ofrecidos por AWS.
- La planificación previa al desarrollo de cualquier software es de vital importancia para garantizar que el resultado del proyecto sea satisfactorio. En este caso, se demostró que la arquitectura propuesta desde un principio, donde se interconectan front-end, back-end y la base de datos fue correcta para cumplir con los requerimientos.
- Durante el desarrollo de la aplicación web se necesitó aplicar distintos contenidos teóricos abordados en la carrera de Ingeniería en Ciencias de la Computación. Las diferentes clases de: Programación, Redes, Bases de datos y Aplicaciones distribuidas formaron una sólida base de conocimiento para modelar e implementar este proyecto.

Trabajo futuro:

La aplicación presentada en este documento es todavía un prototipo y por lo tanto existen múltiples funcionalidades que pueden ser implementadas a futuro. Un ejemplo de esto sería brindar la capacidad al usuario de escoger el modelo estadístico sobre el cual medir la relación entre las dos variables. Sin embargo, al corto plazo la mejora más significativa para la aplicación sería compatibilizar su contenido para distintos dispositivos. De momento, además

de funcionar en el monitor de 1920x1080p donde se desarrolló, la aplicación también funciona en los dispositivos móviles ipad y ipad pro como se muestra en la Figura 17.

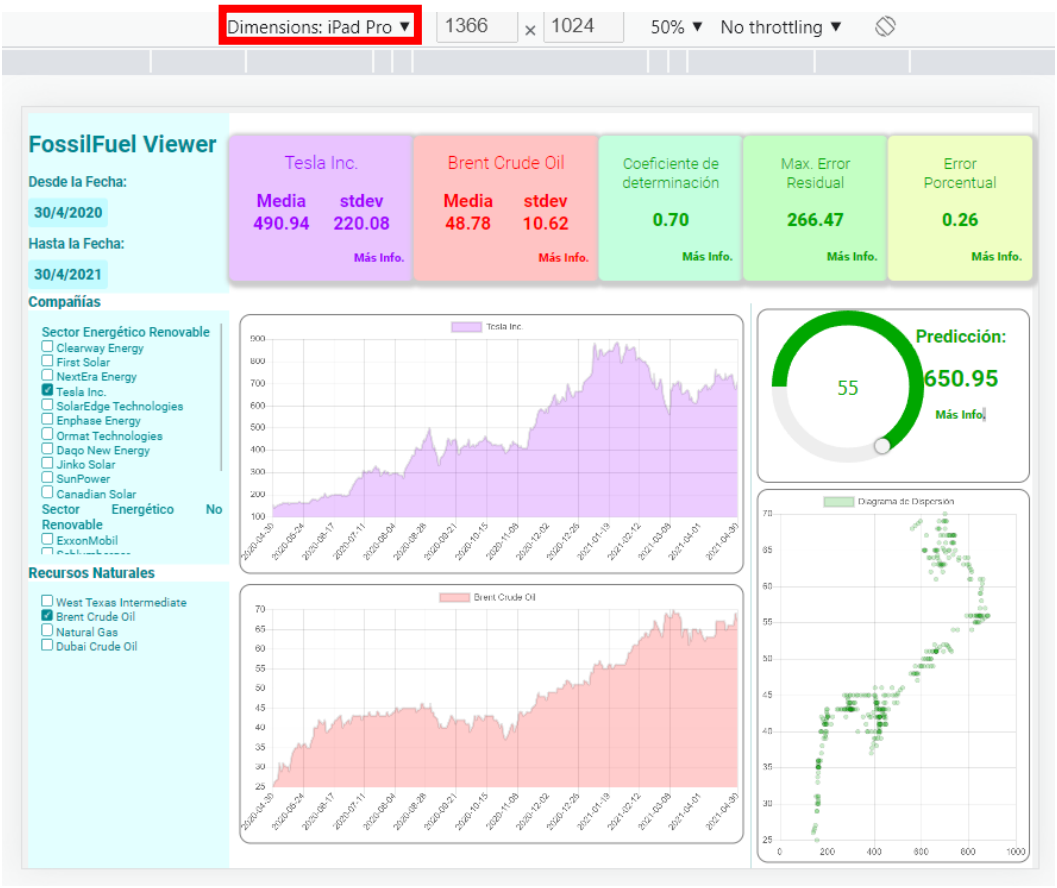


Figura 19. Visualización de la aplicación en el dispositivo móvil iPad Pro.

Aunque también funciona para otros dispositivos como iPad mini o teléfonos celulares, en estos casos la experiencia está lejos de ser la más óptima para el usuario. De momento, en lo que respecta al patrón de diseño 'responsive design' se ha implementado los contenedores de modo que ajusten su tamaño automáticamente en función de la pantalla en la que se encuentran. Pero esto no es suficiente para el caso de dispositivos cuyas proporciones del tamaño de pantalla difieren mucho a la de un monitor convencional. Entonces, como trabajo a futuro se plantea programar el front-end de la aplicación de modo que, dependiendo del tamaño y orientación de la pantalla cambie la distribución de los componentes. Además, en esta nueva distribución, que los componentes ajusten su tamaño acordemente. Una vez implementada, esta

mejora a la aplicación brindará una mejor experiencia a una gran parte del posible conjunto de usuarios a futuro.

REFERENCIAS BIBLIOGRÁFICAS

Anable, J., Brand, C., Eyre, N., Layberry, R., Bergman, N., Strachan, N., Fawcett, T. & Tran. (2011). Lifestyle and Energy Consumption. Recuperado en el 26 de Septiembre de 2021, de https://ukerc.rl.ac.uk/UCAT/PUBLICATIONS/Energy_2050_-_WG1_Energy_Demand_Lifestyle_and_Energy_Consumption.pdf

Fuad, F., Ghozali, I., Kharis, A., Nurcahyono, N & Sukesti, F. (2021). Factors Affecting the Stock Price: The Role of Firm Performance. Recuperado en el 26 de Septiembre de 2021, de https://www.koreascience.or.kr/article/JAKO202104142178581.pdf

Hartill, R. (2021). What Is Adjusted Closing Price? Recuperado en el 15 de Noviembre de 2021, de https://www.thebalance.com/what-is-adjusted-closing-price-5190242

Syvitski, J., Waters, C.N., Day, J. et al. (2020). Extraordinary human energy consumption and resultant geological impacts beginning around 1950 CE initiated the proposed Anthropocene Epoch. Recuperado en el 26 de Septiembre de 2021, de https://doi.org/10.1038/s43247-020-00029-y