UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN
TECNOLOGÍAS DE LA INFORMACIÓN

"Programación y testeo de módulos de análisis y visualización de datos"

TRABAJO FIN DE GRADO

Febrero - 2024

AUTOR: Eugenio Canals Orts
DIRECTORES: Kristina Polotskaya
Alejandro Rabasa Dolado

AGRADECIMIENTOS

Quería dar las gracias a:

Todos aquellos compañeros de los Grados en Ingeniería Electrónica e Informática que con el transcurso del tiempo pasaron a ser grandes amistades.

El café inesperado en el primer año de universidad con el profesor José María, el cual me mostró unos de los lados más humanos, pero a la vez que más ocultamos; el miedo y la frustración ante diferentes situaciones que nos encontramos a lo largo de la vida, así como el esfuerzo y fe en uno mismo que debemos tener para superar estos obstáculos.

Cada uno de los profesores y profesoras que, a lo largo de estos años, me han permitido aprender tanto y de manera tan distinta, pues a veces escuchar y entender una frase de una persona con experiencia vale más que diez temas de teoría juntos.

Mis tutores de TFG Kristina y Álex, por lo que he aprendido y sigo aprendiendo de ellos, su simpatía, confianza y presteza en todo momento.

Los amigos de toda la vida.

Mi familia, que, aunque sea pequeña es muy grande.

Las grandes personas de mi vida: mi madre por estar siempre y darme todo, aunque tenga que mover montañas, mi padre, mi hermana y mis abuelos por su amor eterno. Y a mi gran compañera de vida Marisa por estar siempre y de manera incondicional.

RESUMEN

En la actual era digital en constante evolución, la generación masiva de datos en todos los ámbitos de la sociedad ha planteado desafíos y oportunidades para las organizaciones. Estos datos tienen el potencial de mejorar la Toma de Decisiones estratégicas y operativas, pero su mero volumen no es suficiente para aprovechar su valor. Por tanto, la Ciencia de Datos surge como la disciplina que busca extraer conocimiento y patrones a partir de datos específicos. Y de manera subyacente el Análisis de Datos, mediante métodos y algoritmos, desempeña un papel fundamental en la obtención de conocimientos para respaldar la Toma de Decisiones.

En paralelo, el concepto de Big Data ha ganado relevancia al referirse a la acumulación masiva de datos estructurados y no estructurados que superan la capacidad de las herramientas tradicionales de procesamiento y análisis. El desafío radica en extraer información significativa de este volumen de datos, donde las herramientas de Ciencia de Datos y análisis juegan un rol crucial.

Estas herramientas permiten a las organizaciones adquirir información valiosa, identificar tendencias, predecir comportamientos y tomar decisiones. Al aplicar técnicas de Machine Learning y minería de datos, se pueden descubrir correlaciones ocultas y patrones relevantes, incluso en conjuntos de datos extremadamente grandes y complejos.

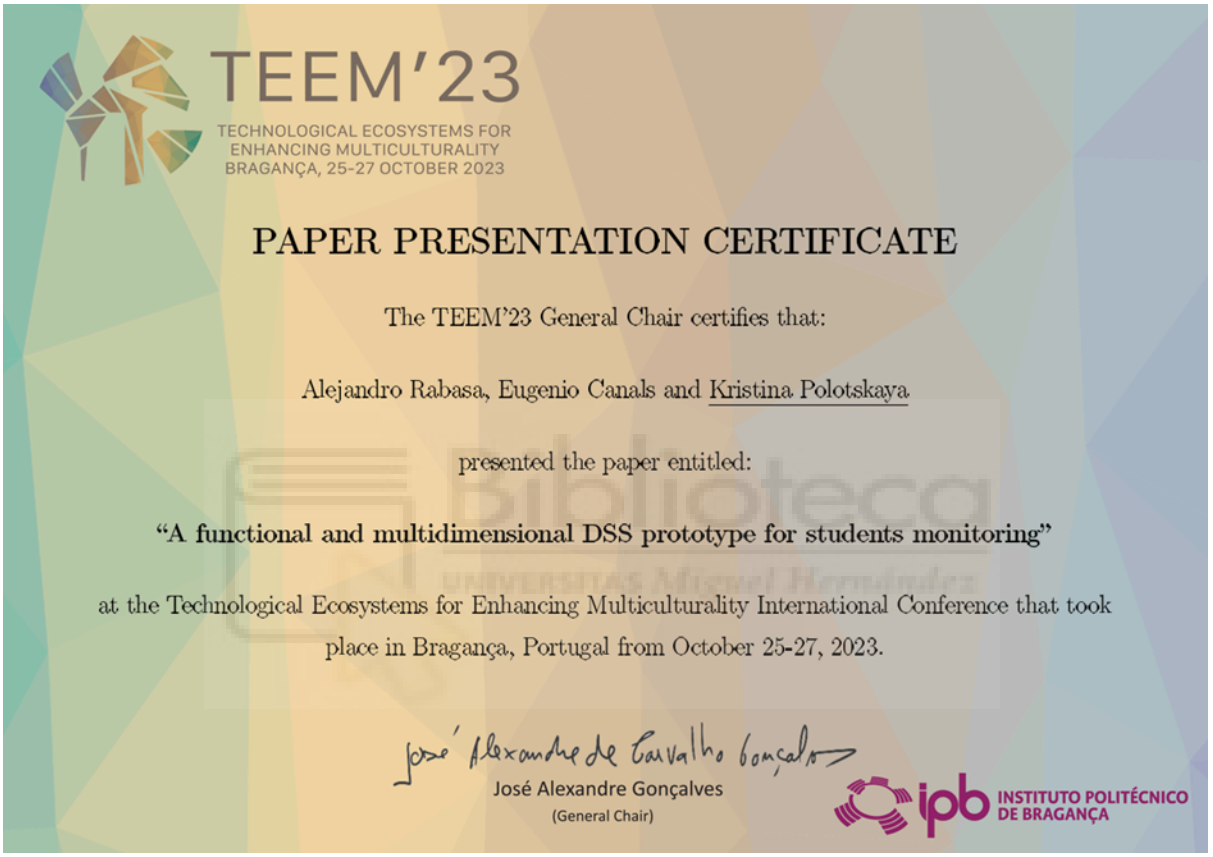
En este contexto, la justificación de este proyecto reside en la necesidad actual de Análisis y Visualización de Datos para tomar decisiones informadas en diversos sectores. Ante el crecimiento constante en volumen y complejidad de los datos, desarrollar una herramienta web dedicada para este propósito puede proveer beneficios significativos y cumplir con las necesidades de entidades que buscan aprovechar al máximo la información contenida en sus datos.

La creación de una aplicación desde cero ofrece un mayor control sobre funcionalidades y especificaciones, permitiendo adaptarla a necesidades particulares. Esto contrasta con las soluciones comerciales menos personalizables. Además, el proceso de desarrollo implica una investigación exhaustiva de tecnologías existentes y su implementación, determinando la viabilidad de una aplicación web enfocada en el tratamiento de datos.

Palabras clave: Ciencia de Datos, Análisis de Datos, Minería de Datos, Machine Learning, Big Data, Visualización de Datos, Toma de Decisiones, Herramientas Web, Python y R.

TECHNOLOGICAL ECOSYSTEMS FOR ENHANCING MULTICULTURALITY (TEEM)

El desarrollo del presente Trabajo de Fin de Grado condujo a la participación y publicación del artículo titulado "A functional and multidimensional DSS prototype for students monitoring", en colaboración con los autores Alejandro Rabasa y Kristina Polotskaya en el congreso de TEEM.



TEEM'23

TECHNOLOGICAL ECOSYSTEMS FOR ENHANCING MULTICULTURALITY BRAGANÇA, 25-27 OCTOBER 2023

PAPER PRESENTATION CERTIFICATE

The TEEM'23 General Chair certifies that:

Alejandro Rabasa, Eugenio Canals and Kristina Polotskaya

presented the paper entitled:

"A functional and multidimensional DSS prototype for students monitoring"

at the Technological Ecosystems for Enhancing Multiculturality International Conference that took place in Bragança, Portugal from October 25-27, 2023.

José Alexandre Gonçalves
(General Chair)

ipb INSTITUTO POLITÉCNICO DE BRAGANÇA

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1
Introducción

1.1.- ENTORNO DE APLICACIÓN

En la era actual, estamos inmersos en un mundo digital en constante evolución, donde se genera una cantidad ingente de datos en todo momento y en cualquier ámbito de nuestra sociedad. Esta explosión de datos ha creado desafíos y oportunidades para las organizaciones, ya que éstos tienen un potencial significativo para mejorar la Toma de Decisiones estratégicas y operativas. Sin embargo, el mero volumen de datos no es suficiente para aprovechar su valor; se requiere de herramientas y técnicas específicas para su tratamiento y análisis.

Es aquí donde entra en juego la Ciencia de Datos, una disciplina cuyo objetivo es extraer conocimiento y revelar patrones a partir de unos datos determinados. En este contexto encontramos el Análisis de Datos que, a través de la aplicación de métodos y algoritmos, permite obtener conocimientos a partir de los datos y respaldar la Toma de Decisiones.

En paralelo a estos conceptos anteriores, el de Big Data ha cobrado una relevancia abrumadora. Referido a la acumulación masiva de datos, ya sean estructurados como no estructurados, que superan la capacidad de las herramientas tradicionales de procesamiento y análisis.

El desafío radica en extraer información significativa de este océano de datos, donde las herramientas de Ciencia de Datos y análisis juegan un papel crucial.

Las herramientas de Ciencia de Datos permiten a las organizaciones adquirir información valiosa a partir de los datos, identificar tendencias, predecir comportamientos y tomar decisiones más informadas. Estas herramientas proporcionan capacidades avanzadas de visualización, modelado estadístico, análisis predictivo y generación de informes interactivos. Al aplicar técnicas de Machine Learning y Minería de Datos, se pueden descubrir correlaciones ocultas y patrones relevantes, incluso en conjuntos de datos extremadamente grandes y complejos.

1.2.- JUSTIFICACIÓN DEL PROYECTO

En el entorno actual, el Análisis y Visualización de Datos se han convertido en componentes esenciales para la Toma de Decisiones informadas en una amplia gama de industrias y sectores. Con el creciente volumen y complejidad de los datos disponibles, es importante contar con herramientas que faciliten su análisis y visualización de manera efectiva y accesible.

En este sentido, la creación de una herramienta web dedicada al Análisis y Visualización de Datos puede ofrecer beneficios significativos y satisfacer las necesidades de aquellas entidades que buscan aprovechar al máximo la información contenida en los datos.

El desarrollo, partiendo desde el inicio, de una aplicación enfocada al tratamiento de datos supone un mayor control sobre las funcionalidades y especificaciones que ésta pueda tener, pudiéndose agregar en un futuro nuevas características. Es decir, a una entidad le puede resultar más útil un aplicativo enfocado a sus necesidades reales que disponer de una herramienta ya existente en el mercado con servicios mucho menos personalizados.

También, el propio proceso para llevar a cabo este proyecto requiere de una investigación previa de las tecnologías existentes y puesta en marcha de las mismas, dando como resultado un veredicto sobre la viabilidad para desarrollar una aplicación web enfocada al tratamiento de datos.

1.3.- OBJETIVOS

Una vez planteado el proyecto de desarrollo de una aplicación web para el Análisis y Visualización de Datos, a continuación, se exponen los objetivos que se pretenden alcanzar.

Objetivos principales:

- **Facilitar el uso de la aplicación**: garantizar que cualquier usuario con conocimientos básicos de informática debe ser capaz de interactuar con la aplicación.

- **Proteger la aplicación**: identificar al usuario mediante un rol y unas credenciales para acceder a la aplicación.

- **Disponer de un sistema de ayudas**: ofrecer al usuario una serie de ayudas a lo largo de la aplicación para que sepa qué está realizando en cualquier momento.

- **Obtener resultados visuales**: presentar resultados gráficos muy vistosos para facilitar la interpretación de la información al usuario.

- **Interactuar con la aplicación**: manipular los resultados, no deben estar limitados a gráficos y tablas estáticas, el usuario tiene el poder de modificar dichas salidas para ajustarlas a sus necesidades.

Objetivos contribuyentes:

- **Investigar las tecnologías disponibles**: recopilar información sobre las diferentes tecnologías existentes para el desarrollo de una aplicación en el ámbito de la Ciencia de Datos.

- **Comprobar la viabilidad**: demostrar si se puede llevar a cabo el proyecto a raíz de los objetivos planteados anteriormente.

Objetivos personales:

- **Poner en práctica conocimientos**: aplicar en un proyecto la educación y disciplina adquirida durante el grado.

- **Aprender nuevas tecnologías**: conocer en más profundidad lenguajes de programación, bibliotecas y frameworks que sean frecuentemente utilizados en el contexto de la Ciencia de Datos.

- **Adquirir experiencia**: resolver posibles problemas que puedan surgir, ya que no todo es conocido a la hora de llevar a cabo el desarrollo de un proyecto.

Objetivo transversal:

- **Crear un prototipo de aplicación explotable**: desarrollar un programa que pueda ser aplicado en un entorno real en el futuro, teniendo en cuenta que el trabajo resultante puede presentar ciertas limitaciones.

Resaltar que, pese a hacer una distinción entre objetivos principales y contribuyentes, éstos últimos son imprescindibles para conseguir los principales.

1.4.- QUÉ NO SE PRETENDE

El tiempo es un factor limitante y por tanto, se debe acotar el trabajo a realizar. Debido a este motivo, la aplicación web a desarrollar presenta las siguientes limitaciones:

- **Preprocesamiento de datos**: la aplicación web a desarrollar prescinde de la parte del pretratamiento, implementando directamente funcionalidades para la visualización y ejecución de algoritmos sobre los datos para obtener resultados. Esto implica que los datos de entrada deben estar preprocesados.

- **Implementación en entorno web real**: el aplicativo se ejecuta a través de la web pero se limita a un ámbito local, no se contempla el despliegue de una infraestructura completa en línea. Dicha restricción, deriva en el uso de servidores locales web.

- **Desarrollar una aplicación web completa**: debido a las restricciones temporales, no es viable crear una aplicación web íntegra. Sin embargo, el enfoque se ha centrado en alcanzar otros objetivos previamente expuestos. Estos objetivos incluyen la creación de un prototipo funcional que permita demostrar la viabilidad del proyecto, la implementación de características clave para el Análisis de Datos y la interacción con la aplicación, y el diseño de una interfaz de usuario intuitiva y atractiva. Con el enfoque en estos elementos, podemos sentar las bases para futuras iteraciones y mejoras en el desarrollo de la aplicación web completa en un futuro.

Capítulo 2 Antecedentes y estado de la cuestión

La búsqueda de información es una etapa fundamental en cualquier trabajo académico, ya que permite conocer el estado del arte, las fuentes disponibles y los criterios de calidad de la investigación. En este capítulo, se presenta la búsqueda de información sobre el tema del trabajo, así como los resultados obtenidos y el análisis crítico de los mismos.

El objetivo es adquirir todo el conocimiento posible en esta fase previa a la formulación y resolución del problema planteado, es decir, la creación de una aplicación web para el testeo de módulos de Análisis y Visualización de Datos.

2.1.- SITUACIÓN ACTUAL

En el ámbito de la Ciencia de Datos, es fundamental contar con herramientas que permitan analizar y visualizar datos. Estas herramientas deben facilitar la integración, la depuración y la validación de los resultados obtenidos por los módulos, así como ofrecer una interfaz amigable y accesible para los usuarios finales. Ante este entorno, se plantea la necesidad de

crear una aplicación web que sirva como plataforma para el testeo de estos módulos, y que permita su ejecución y visualización desde un navegador web.

El propósito de este capítulo es exponer la situación actual en el que se quiere investigar qué herramientas hay para crear una aplicación web para el testeo de módulos de Análisis y Visualización de Datos, y cuáles son los requisitos, ventajas y desafíos que implican su uso.

2.2.- HERRAMIENTAS DISPONIBLES

Para abordar el desarrollo de una aplicación web, actualmente, el uso de frameworks y bibliotecas son las prácticas por excelencia. Antes de examinar las posibles herramientas para dicho objetivo, describiremos brevemente de qué se tratan.

Los frameworks [2.1] surgen ante la necesidad de estructurar y normalizar el código de un sistema, ya que previamente a su aparición, en el desarrollo de un sistema de información lo que entendemos como "datos" era la única información estructurada.
Por ende, el uso de frameworks ofrecen una estructura a la hora de realizar el desarrollo e implementación de aplicaciones. Adicionalmente, disminuyen los tiempos durante la realización de un proyecto, así como una mayor escalabilidad y mantenimiento.
Por otra parte, agregan funcionalidades adicionales al lenguaje de programación empleado en la aplicación y automatizan patrones de programación enfocados al tipo de desarrollo en cuestión, mejorando así la estructura, legibilidad y posterior sostenibilidad del código.

Por otra parte, recordemos que [2.2], una biblioteca comúnmente llamada también "librería" por su interpretación directa del inglés library, es un concepto previo al de framework. Las bibliotecas consisten en una serie de clases o métodos que ofrecen a otra aplicación su comportamiento, es decir, cómo deben realizar un procedimiento en cuestión. Se distinguen de los frameworks también porque no ofrecen un control de flujo de datos interno, herencia y patrones de diseño.

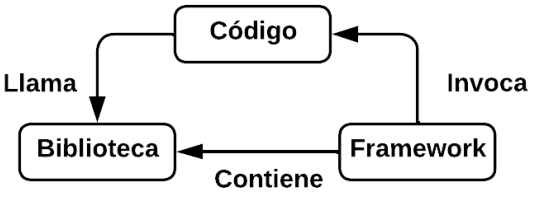


Figura 2.1: Biblioteca VS Framework.

Hemos ubicado la utilidad de los frameworks y bibliotecas en el paradigma del desarrollo a la hora de construir una aplicación. Regresando al tema que nos atañe, la creación de una

aplicación web que permita la conexión con módulos para el Análisis y Visualización de Datos, describiremos las posibles herramientas candidatas.

2.2.1.- React

Esta herramienta [2.3] de código abierto y mantenida por la comunidad de desarrolladores y Facebook, se trata de una biblioteca para crear interfaces gráficas para el usuario y aplicaciones web de una sola página (SPA) a través del lenguaje de programación JavaScript.

La base de construcción de la aplicación es a través de los denominados “Componentes", es decir, fragmentos de código encapsulados que representan una parte de la interfaz gráfica, manejando éstos su propio estado. Del concepto anterior, calificamos a React como declarativo, ya que, en función de cada uno de los estados de los componentes, éstos se actualizan y renderizan de manera independiente entre ellos, definiendo así cómo debe comportarse cada uno dentro de la aplicación. Se puede transmitir información entre componentes mediante el uso de lo que se conoce como “Props”.

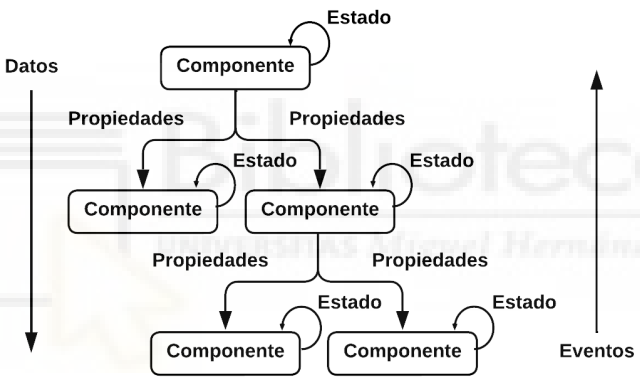


Figura 2.2: Lógica de funcionamiento de React.

A raíz de la previa descripción, podríamos añadir a la definición [2.4] que su uso está destinado a la capa de “Vista” en la estructura de construcción de aplicaciones Modelo-Vista-Controlador.

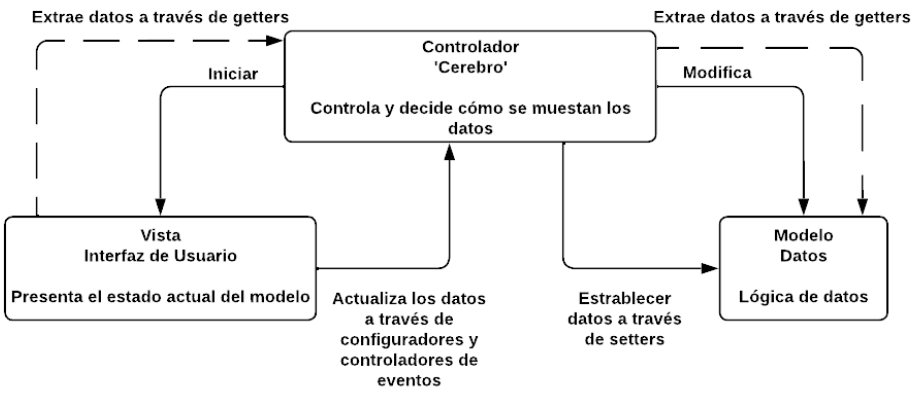


Figura 2.3: Arquitectura Modelo-Vista-Controlador.

React cuenta con su propio DOM virtual (Modelo de Objetos del Documento) lo cual permite aumentar la eficiencia de la aplicación al poder trabajar en paralelo con el DOM integrado del navegador web empleado.

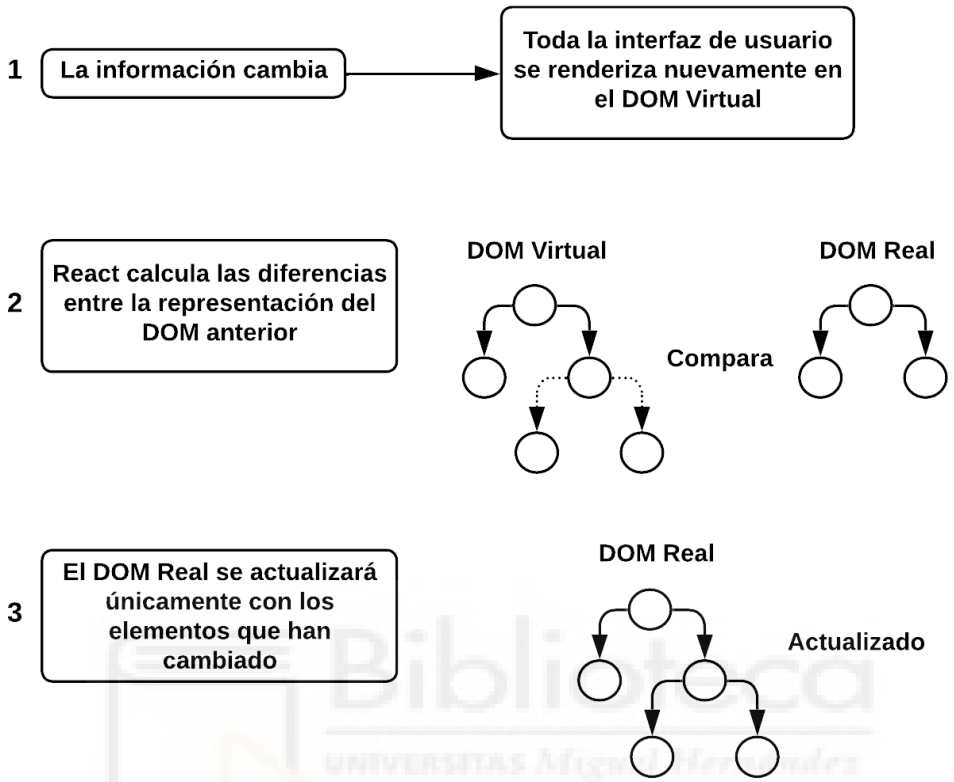


Figura 2.4: Cómo funciona el Modelo de Objetos del Documento Virtual.

Otra característica importante es su compatibilidad con módulos construidos con el lenguaje de programación R y Python, extremadamente útiles en el procesamiento de datos.
Para lograr este objetivo, se puede utilizar R en el backend y React en el frontend, comunicados a través de la API Shiny, facilitando esta tarea [2.5], el uso del paquete "reactR".

También, pueden usarse los datos generados por el lenguaje de programación R o Python, en un formato JSON (JavaScript Object Notation), los cuales pueden ser leídos por ReactJS.

React, dispone de una amplia documentación y comunidad activa. Otro punto a mencionar [2.6] es que su curva de aprendizaje es inicialmente costosa en comparación a otras herramientas.

Un detalle a tener en cuenta es que React, no se trata de un framework per se, aunque por la comunidad de desarrolladores está generalmente aceptado y extendido el uso de este término. Siendo la definición que da la propia organización: "Una biblioteca de JavaScript para construir interfaces de usuario".

2.2.2.- Django

Django [2.7] se trata de un framework de alto nivel y de código abierto para el desarrollo web escrito bajo el lenguaje de programación Python. Creado por experimentados programadores su filosofía es facilitar el trabajo. Es un framework completo ya que ofrece de serie prácticamente todas las funcionalidades que se esperan a la hora de crear un aplicativo web.

Presenta una gran versatilidad, es decir, no está limitado para desarrollar un tipo específico de sitio web. Puede utilizarse para desarrollar aplicaciones para la parte del backend [2.8], frontend [2.9],[2.10] a través de las denominadas "Plantillas" y "Vistas", o de manera completa (full stack).

Si se desea, permite acoplarse a cualquier framework en el frontend, no es necesario emplear el propio Django para este propósito. A su vez, permite extender sus funcionalidades debido a su diseño modular.

Es fácilmente escalable, por su compatibilidad con las arquitecturas hardware en las que cada sección es independiente al resto, pudiendo así modificar, reemplazar o agregar componentes en función de la necesidad requerida en cada momento.

La seguridad es otro de sus aspectos clave, permitiendo, por ejemplo; gestionar la administración de cuentas de manera segura, funciones hash de criptografía para datos sensibles como contraseñas, prevención de ataques de inyección SQL, scripts intrusivos, falsificación de solicitudes y clickjacking.

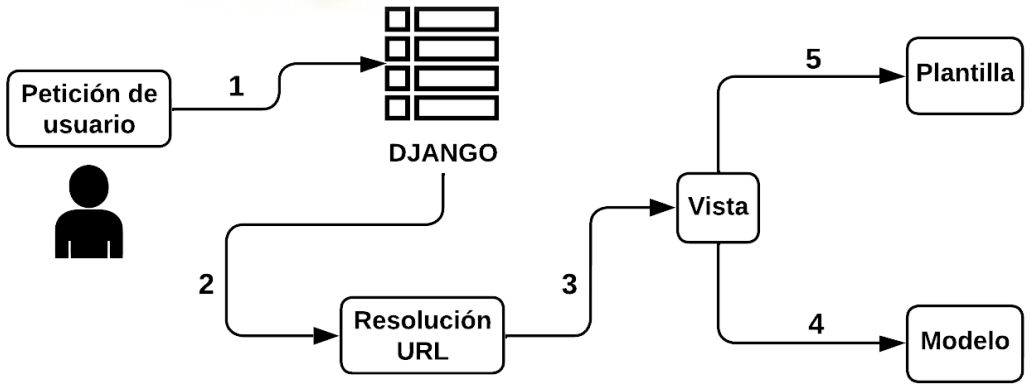


Figura 2.5: Lógica de funcionamiento de Django.

Su código es mantenible y reutilizable gracias al uso de patrones de diseño, además agrupa funcionalidades en módulos siguiendo la metodología de Modelo-Vista-Plantilla (MVP) a diferencia de otras arquitecturas como la de Modelo-Vista-Controlador (MVC). Django al no estar ligado a una plataforma en concreto es altamente portable, pudiéndose ejecutar en entornos de Windows, Mac OS y Linux.

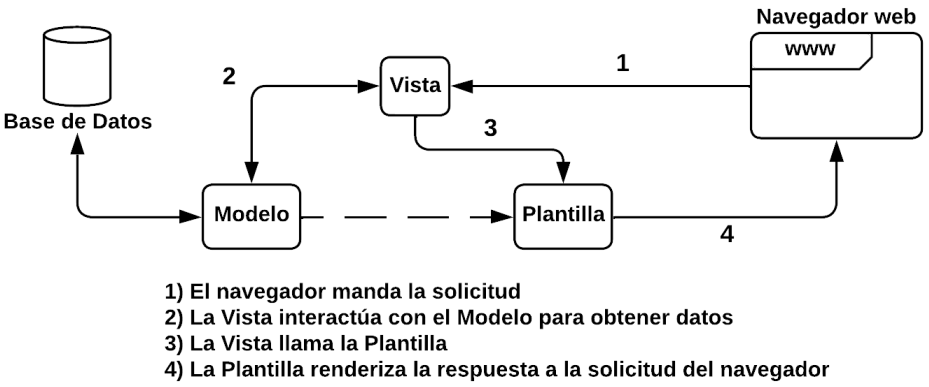


Figura 2.6: Modelo-Vista-Plantilla.

En cuanto a su integración con el Análisis y Visualización de Datos, se puede hacer a través del lenguaje Python con el uso de bibliotecas como Pandas o NumPy para analizar datos y Matplotlib o Seaborn para su visualización. También [2.11] permite la comunicación y gestión de datos con R mediante el módulo “django-rpy2”.

Django dispone de una comunidad y documentación muy extensa accesible al público. Su nivel de aprendizaje [2.12] resulta más lineal y amistoso, debido en parte a que Python, el lenguaje de programación en el que está escrito, dispone también de estas características.

2.2.3.- Angular

Angular [2.13] es un framework de código abierto basado en JavaScript y TypeScript que se usa para crear aplicaciones web de una sola página (SPA) y progresivas (PWA).
Una SPA [2.14] es una aplicación web que carga una sola página HTML y actualiza dinámicamente el contenido según la interacción del usuario. Mientras que una PWA [2.15] es una aplicación web que ofrece una experiencia similar a una aplicación nativa, con funciones como notificaciones, acceso sin conexión y acceso al hardware del dispositivo.

Utiliza plantillas declarativas, inyección de dependencias y herramientas de extremo a extremo para facilitar el desarrollo web frontend.
Las plantillas declarativas [2.16] son fragmentos de HTML que definen la estructura y el comportamiento de la interfaz de usuario. La inyección de dependencias es un patrón de diseño que permite a un objeto recibir sus dependencias (otros objetos o servicios) desde una fuente externa, en lugar de crearlas por sí mismo [2.17]. Las herramientas de extremo a extremo [2.18] son aquellas que permiten automatizar el proceso de desarrollo, prueba y despliegue de una aplicación web.

Se basa en clases tipo “Componentes", cuyas propiedades son las usadas para hacer el binding de los datos [2.19], éste es un proceso con el que los usuarios pueden manipular elementos de una página web a través de un navegador. Los componentes son bloques de código reutilizables que contienen la lógica y la vista de una parte de la aplicación [2.20].

Angular también ofrece directivas, que son atributos personalizados que modifican el comportamiento o la apariencia de los elementos HTML [2.21].

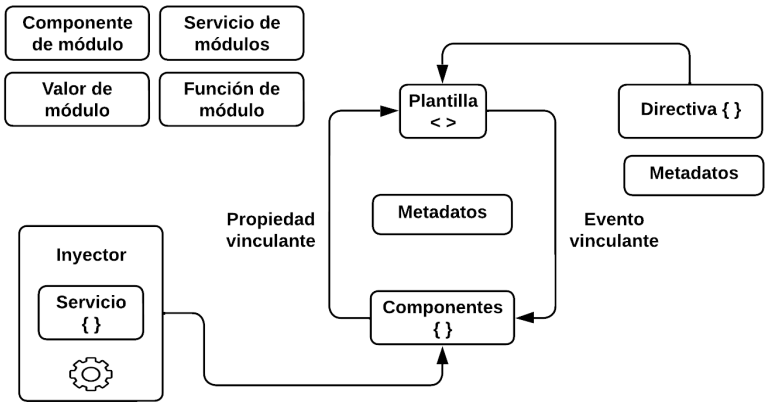


Figura 2.7: Lógica de funcionamiento de Angular.

Es posible integrar Angular con módulos de Python y R, por ejemplo, creando una API en el backend y que la aplicación se conecte a ésta, o a través de Azure Databricks, una plataforma de Análisis de Datos [2.22].

No sigue estrictamente el patrón de diseño Modelo-Vista-Controlador (MVC), sino que utiliza una variante llamada Modelo-Vista-Modelo (MVVM) o simplemente Vista-Modelo. En este patrón, el controlador es reemplazado por un modelo de vista que actúa como intermediario entre la Vista y el Modelo, manejando la lógica de la interfaz de usuario y exponiendo datos del Modelo a la Vista [2.23].

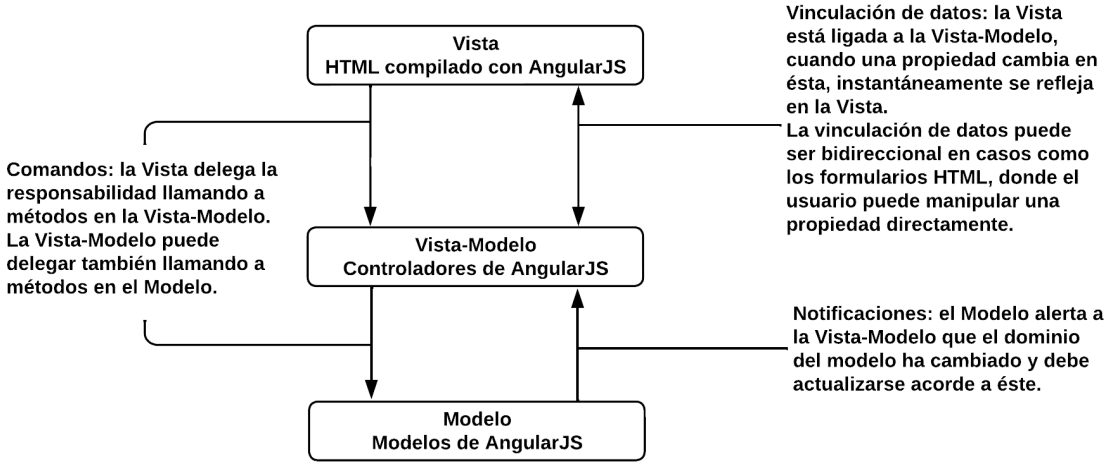


Figura 2.8: Funcionamiento del Modelo-Vista-Modelo.

Mantenido y actualizado por Google, cuenta con una amplia comunidad de desarrolladores que contribuyen a su mejora y documentación [2.24]. Entre sus principales ventajas se encuentran su velocidad y rendimiento, su productividad, su compatibilidad móvil y de escritorio, así como su soporte para el SEO. En cambio, tiene una curva de aprendizaje de nivel medio a elevado.

2.2.4.- Plotly Dash

Es una plataforma de visualización de datos interactiva y de alta calidad que permite a los usuarios construir aplicaciones web dinámicas basadas en Python [2.25]. Dash se basa en la biblioteca de visualización de datos Plotly [2.26] y en el framework para backend de aplicaciones web Flask de Python [2.27]. React.js también forma parte de su construcción.

Entre las principales características de Plotly Dash encontramos la creación de: visualizaciones interactivas y personalizadas, aplicaciones web dinámicas, paneles de control interactivos y widgets personalizados.

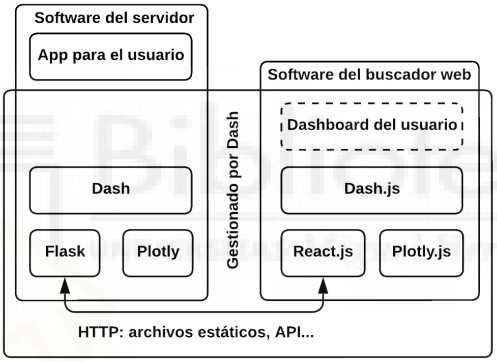


Figura 2.9: Lógica de funcionamiento de Plotly Dash.

Diseñado para funcionar con Python, tiene una gran integración con las bibliotecas científicas como Pandas, NumPy y Scikit-Learn [2.26]. También es posible utilizar Plotly Dash con R mediante el paquete “dashR” [2.28].

Se centra en tres elementos principales para construir aplicaciones web dinámicas basadas en datos, en primer lugar los Layouts para definir la estructura de la aplicación, que incluye la ubicación de los componentes y la apariencia visual de la aplicación, los cuales pueden ser personalizados y se pueden crear widgets interactivos [2.29]. En segundo lugar, las denominadas Callbacks con el objetivo de establecer las interacciones entre los componentes de la aplicación. Éstas son funciones de Python que se ejecutan cuando ocurre un evento, como un clic en un botón o un cambio en un menú desplegable [2.30]. Y en tercer lugar, los Componentes, objetos que se utilizan para construir la interfaz de usuario de la aplicación, por ejemplo: tablas, menús desplegables, botones y campos de entrada [2.31].
Al igual que otras herramientas, como por ejemplo React, el patrón de diseño que utiliza es el de Modelo-Vista-Controlador.

La curva de aprendizaje de Plotly Dash puede ser moderada debido a la necesidad de comprender las bases de Flask y Plotly, aunque la documentación oficial y la comunidad en línea pueden ayudar a acelerar el proceso de aprendizaje. Sin embargo, una vez que se domina la plataforma, puede ser una herramienta poderosa y eficiente para crear aplicaciones web dinámicas y visualizaciones interactivas.
Plotly Dash cuenta con una comunidad activa en constante crecimiento de desarrolladores y usuarios que comparten sus experiencias junto con sus conocimientos a través de foros, tutoriales y proyectos de código abierto [2.32].

2.2.5.- Shiny R

Shiny es un paquete que añade funcionalidades al lenguaje de programación R, el cual permite crear aplicaciones web, basadas en la interacción del usuario con los resultados Análisis y Visualización de Datos. Las aplicaciones pueden alojarse en un sitio web o acoplarlas directamente en archivos de R [2.33].

En caso de crear una aplicación web, sigue el patrón de Modelo-Vista-Controlador, donde la parte de la interfaz de usuario está compuesta por elementos como: botones, cuadros de texto, deslizadores, widgets… Es decir, cualquier componente que permita la interacción con los datos, pudiendo así contemplar los resultados en tiempo real. Mientras que en la parte del servidor es donde se controla toda la lógica de la aplicación.

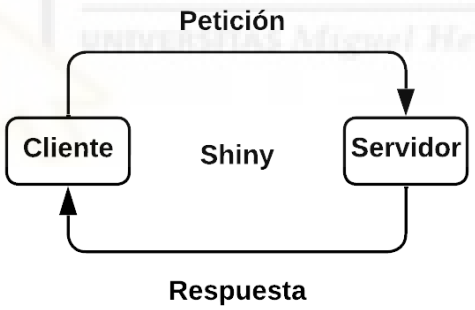


Figura 2.10: Lógica de funcionamiento de aplicaciones web de Shiny.

Shiny puede integrarse con Python a través del paquete “reticulate” de R, éste permite a los usuarios llamar a funciones de Python desde R y viceversa, abriendo la puerta para combinar el potencial de ambos lenguajes en una sola aplicación web. También, es posible utilizar reconocidas bibliotecas para el Análisis de Datos como Pandas, NumPy o Matplotlib [2.34].

Al ser una herramienta de R, Shiny se integra de forma nativa con este lenguaje y permite a los usuarios utilizar todas las funciones y bibliotecas de R en sus aplicaciones web.

El aprendizaje de Shiny no destaca por su excesiva dificultad, esto es debido a la cuantiosa información disponible en línea incluyendo documentación, tutoriales y ejemplos de código, que pueden ayudar a los usuarios a aprender Shiny R [2.35].

Su comunidad es activa, creando constantemente nuevos paquetes y funcionalidades que aumentan las capacidades de Shiny. Por ejemplo, hay paquetes que permiten la autenticación de usuarios, la creación de paneles de control y la creación de visualizaciones avanzadas. La comunidad también ha creado una gran cantidad de temas y plantillas para que los usuarios puedan personalizar fácilmente la apariencia de sus aplicaciones.

2.2.4.- Resumen de las herramientas

A continuación, a través de la siguiente tabla se expone cada herramienta descrita anteriormente, ofreciendo así, una manera cómoda para ver las principales características y realizar una comparación entre ellas.

Tabla 2.1: Resumen de las herramientas expuestas.

Propiedades	React	Django	Angular	Plotly Dash	Shiny R
Lenguaje	JavaScript	Python	TypeScript	Python	R
Uso	Interfaz de usuario	Aplicaciones y sitios web	Interfaz de usuario	Aplicaciones web basadas en datos	Aplicaciones web basadas en datos
Enfoque	Biblioteca frontend	Framework frontend y backend	Framework frontend	Framework frontend y backend	Paquete de R
Principales elementos	Componentes	Modelos, vistas y plantillas	Componentes	Componentes	Componentes
Patrón de diseño	MVC	MVP	MVVM	MVC	MVC
Integración con Python	A través de bibliotecas	Totalmente integrado	A través de bibliotecas	Totalmente integrado	A través de bibliotecas
Integración con R	A través de bibliotecas	A través de bibliotecas	A través de bibliotecas	A través de bibliotecas	Totalmente integrado
Curva de aprendizaje	Moderada a alta	Moderada	Moderada	Moderada	Moderada
Comunidad	Muy grande y activa	Grande y activa	Grande y activa	Creciente y activa	Grande y activa

2.3.- VALORACIÓN

A lo largo de este capítulo hemos visto cinco posibles herramientas para la creación de una aplicación web para el testeo de módulos de Análisis y Visualización de Datos: React, Django, Angular, Plotly Dash y Shiny R.

Todas contienen características apropiadas para lograr el propósito expuesto, y a su vez, diferentes maneras de implementarlas, mediante las tecnologías que las amparan. Esto demuestra que, en el ámbito en el que estamos realizando la investigación, no existe una herramienta fundamental y exclusiva, sin embargo, a raíz de los estudios que se han realizado en el capítulo, podemos acotar la información obtenida y seleccionar dos de ellas como las candidatas más adecuadas.

Tras los resultados de búsqueda, las herramientas seleccionadas son Django y Shiny R.

Sobre Django podemos destacar que es una herramienta ampliamente conocida y utilizada en el mercado, lo que se traduce en un soporte activo y grande de los desarrolladores y de la comunidad, junto con un ciclo de vida del software muy longevo.
Respecto a Shiny, se trata de una opción muy apropiada ya que este paquete está hecho específicamente para el Análisis de Datos.

Un aspecto clave que comparten ambas opciones es el lenguaje de programación en el que están basadas; Python y R, para Django y Shiny, respectivamente. Lenguajes utilizados para el Análisis y Visualización de Datos muy extendidos y potentes. A su vez, las dos herramientas permiten incorporar otros lenguajes como JavaScript, HTML y CSS dentro de la misma aplicación.

En el siguiente capítulo se profundizará sobre Django y Shiny, conociendo en más detalle su arquitectura, funcionamiento y la valoración de ambas.

Capítulo 3
Hipótesis de trabajo

En esta parte del trabajo se expone con más detalle las dos herramientas, seleccionadas en el capítulo anterior, como las mejores candidatas para desarrollar una aplicación web para el testeo de módulos de Análisis y Visualización de Datos: Django y Shiny.

3.1.- DJANGO

Django es un framework web de código abierto y gratuito basado en Python que se utiliza para desarrollar aplicaciones web de alta calidad y escalables. Fue creado en 2003 por un equipo de desarrolladores de web en Lawrence Journal-World, una compañía de medios de comunicación en Kansas. Se centra en la eficiencia, la simplicidad y la reutilización del código, lo que lo hace popular entre los desarrolladores [2.11].

Django sigue un patrón de arquitectura MVP (Modelo-Vista-Plantilla) donde los modelos representan los datos, las vistas procesan la lógica de la aplicación y las plantillas se encargan

de la presentación de los datos. Esta estructura ayuda a mantener un código organizado [2.16].

Su utilidad en el Análisis de Datos radica en su capacidad para manejar grandes conjuntos de datos y la integración con otras bibliotecas de análisis como Pandas, NumPy, Matplotlib o Seaborn. A través del módulo "django-rpy2" se puede integrar con el lenguaje de programación R [2.17].

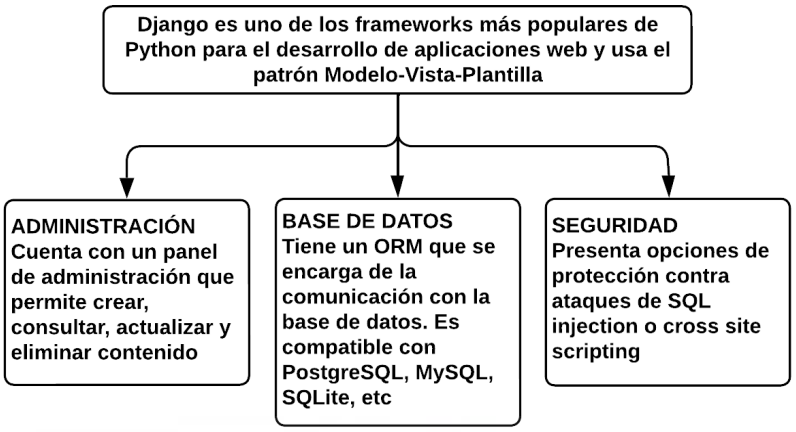


Figura 3.1: Resumen sobre qué es Django.

3.1.1.- Lenguaje de programación

Python [3.1], a grandes rasgos, se trata de un lenguaje de programación de alto nivel, interpretado, multipropósito, con tipado dinámico y orientado a objetos. Su uso ha sido creciente de manera continuada desde los últimos años, hasta convertirse en la actualidad en uno de los lenguajes más populares para el desarrollo de software.

El ámbito de uso de Python no es específico, en el sentido de que puede utilizarse para aplicaciones de escritorio con interfaz gráfica, smartphone y científicas, comunicaciones de red, desarrollo de videojuegos y aplicaciones para la web. Puede aplicarse a diversas plataformas y sistemas operativos, como Linux, Mac OS y Windows. Incluso en smartphones como hizo Nokia con Symbian.

Los motivos por los que es elegido son su potencia, flexibilidad y sintaxis clara y concisa. Por otra parte, no es necesario designar tiempo de compilación ya que es interpretado.

Es Open Source, cualquier persona puede hacer uso de Python sin necesidad de pagar licencia, incluso su intérprete se distribuye gratis entre las distintas plataformas. La última versión hasta la fecha es conocida como Python 3.

Pese a ser mantenido actualmente por su comunidad (Python Software Foundation), su creación se dio a principios de los noventa por parte de Guido van Rossum, un investigador

holandés, el cual desarrolló Python para sustituir el lenguaje ABC para el sistema operativo Amoeba. Su nombre deriva del grupo humorístico de la época Monty Python, del cual Guido era fan.

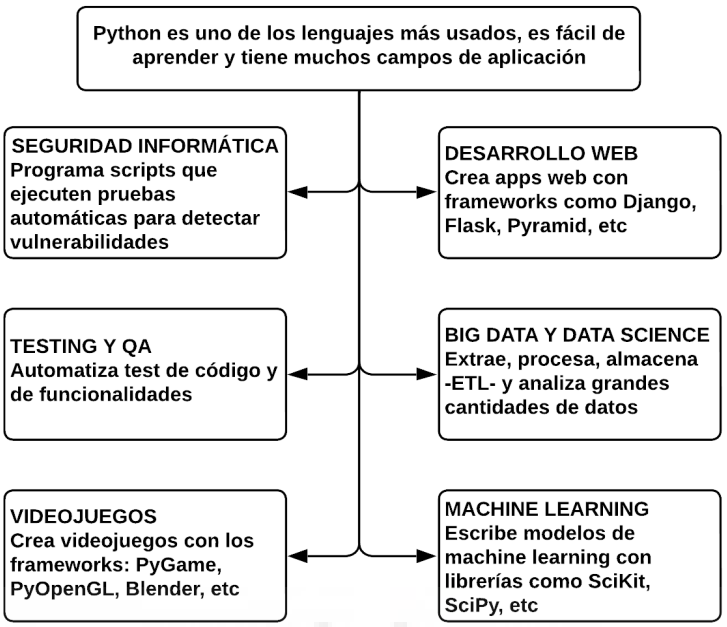


Figura 3.2: Principales campos de uso de Python.

3.1.2.- Patrón de diseño

Django utiliza el patrón de diseño MVP (Modelo-Vista-Plantilla), que es una variante del patrón MVC (Modelo-Vista-Controlador) [2.16]. En el patrón MVP, la plantilla se encarga de presentar los datos, la vista se encarga de procesar la información de la solicitud del usuario y la lógica de negocios, y el modelo representa la estructura de datos y la interfaz para acceder a ella. Estos elementos se describen con más detalle a continuación:

- **Modelos**: permiten la definición de las estructuras de datos que se utilizarán en la aplicación. Un modelo en Django es una clase de Python que representa una tabla en una base de datos relacional, cada atributo de la clase representa una columna en la tabla y el modelo define las relaciones que existen entre las diferentes tablas.

  En los modelos se definen campos para almacenar diferentes tipos de datos, tales como cadenas de texto, números, fechas y relaciones con otras tablas. También, se pueden añadir validaciones para garantizar la integridad de los datos y se pueden definir métodos para manipular los datos de la tabla.

  Por tanto, se utilizan para crear, leer, actualizar y eliminar registros en la base de datos, y se integran con el ORM (Object-Relational Mapping) para facilitar el acceso y manipulación de los datos [3.2].

- **Vistas**: son los elementos encargados de procesar las solicitudes HTTP y devolver una respuesta HTTP, que generalmente es una página web renderizada con contenido dinámico. Las vistas son funciones de Python que se definen en los archivos de Python llamados “views.py” y se conectan a las URLs mediante el enrutador de URL de Django.

  Las vistas pueden realizar diversas tareas, como interactuar con los modelos de la base de datos, procesar datos de formularios, enviar correos electrónicos, entre otras. Django proporciona un conjunto de vistas genéricas integradas que simplifican la creación de vistas para tareas comunes, como la creación, actualización y eliminación de objetos en la base de datos.

  También son conocidas por su capacidad de integrarse con plantillas, lo que permite la creación de páginas web altamente personalizables y escalables [2.14].

- **Plantillas o Templates**: son archivos HTML que se utilizan para definir la estructura y el contenido de las páginas web. Permiten separar la lógica de la presentación, lo que hace que sea más fácil mantener el código y hacer cambios en el diseño sin afectar la funcionalidad. Se basan en un lenguaje de marcado llamado Django Template Language (DTL), que permite incrustar código de Python en el HTML y acceder a variables, condiciones y bucles. Se pueden definir bloques y extensiones que se pueden reutilizar en diferentes páginas.

Django proporciona un sistema de herencia de plantillas que permite crear una plantilla base y extenderla con otras plantillas que sólo cambian partes específicas. Esto es especialmente útil para sitios web con un diseño consistente en todas las páginas [2.13].

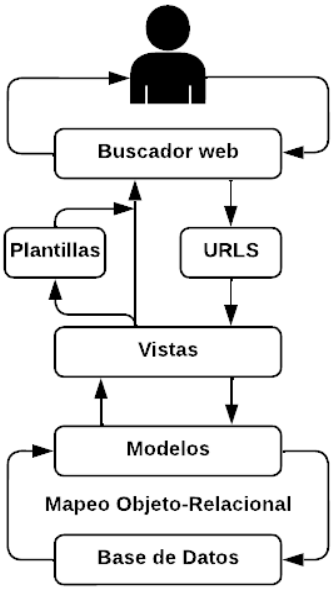


Figura 3.3: Funcionamiento del Modelo-Vista-Plantilla.

3.1.3.- Librerías

Las bibliotecas (o librerías) de Python más destacadas para Django que se pueden utilizar en el Análisis de Datos son las siguientes [3.3]:

- **NumPy**: para el cálculo numérico que se utiliza comúnmente en el Análisis de Datos. Resuelve en parte el problema de la lentitud mediante la provisión de arreglos multidimensionales y funciones y operadores que operan eficientemente en éstos.

  Sus características clave incluyen la provisión de funciones precompiladas rápidas para rutinas numéricas, el cómputo orientado a arreglos para una mejor eficiencia, el soporte para un enfoque orientado a objetos, y cálculos más compactos y rápidos con la vectorización. Forma la base de otras bibliotecas como SciPy y Scikit-learn.

- **Pandas**: es una biblioteca de Python para Análisis de Datos que es imprescindible en el ciclo de vida de la Ciencia de Datos. Junto con NumPy y Matplotlib, es la biblioteca de Python más popular y ampliamente utilizada para el Análisis de Datos. Pandas proporciona estructuras de datos rápidas y flexibles, como los DataFrame, que están diseñados para trabajar con datos estructurados de manera rápida e intuitiva.
  Las principales características de Pandas son su sintaxis elocuente y sus ricas funcionalidades, que brindan la libertad de trabajar con datos faltantes, permite crear funciones propias y ejecutarlas en una serie de datos, y proporciona herramientas de manipulación de datos de alto nivel.

  Pandas es útil para el procesamiento y limpieza de datos generales, trabajos ETL (extracción, transformación y carga) para la transformación de datos y almacenamiento de datos, ya que tiene excelente soporte para cargar archivos CSV en su formato de DataFrame.

- **Matplotlib**: es una biblioteca de Visualización de Datos en Python. Se utiliza principalmente para crear gráficos y visualizaciones de datos de alta calidad en diferentes formatos, como gráficos de línea, barras, histogramas, gráficos de dispersión, gráficos de contorno, entre otros.

  Tiene una amplia gama de funciones y herramientas que permiten personalizar la apariencia y el estilo de los gráficos para hacerlos más atractivos y comunicar de manera efectiva los hallazgos de los datos. También se puede integrar con otras bibliotecas de Python, como NumPy y Pandas, para trabajar de manera más eficiente con datos grandes y complejos.

- **SciPy**: es reconocida por su capacidad para extender NumPy y proporcionar una amplia variedad de funciones y algoritmos eficientes para cálculos científicos y técnicos.

Sus características incluyen una colección de algoritmos y funciones construidos en la extensión NumPy de Python, comandos de alto nivel para manipulación y Visualización de Datos, procesamiento de imágenes multidimensionales con el submódulo “SciPy.ndimage”, y funciones integradas para resolver ecuaciones diferenciales.

- **SciPy**: permite operaciones de imagen multidimensionales, resolución de ecuaciones diferenciales y transformada de Fourier, algoritmos de optimización y álgebra lineal.

- **Scikit-learn**: es una biblioteca de aprendizaje automático muy utilizada y activamente desarrollada para Python. Se integra fácilmente con otras bibliotecas de programación de aprendizaje automático como NumPy y Pandas. Scikit-learn es una de las bibliotecas de aprendizaje automático más populares para algoritmos clásicos de aprendizaje automático.

  Está construido sobre dos bibliotecas básicas de Python, NumPy y SciPy. Scikit-learn admite la mayoría de los algoritmos de aprendizaje supervisado y no supervisado. También se puede utilizar para Minería de Datos y Análisis de Datos, lo que lo convierte en una herramienta excelente para quienes se inician en el aprendizaje automático.

- **TensorFlow**: es un marco para definir y ejecutar cálculos que involucran tensores, que son objetos computacionales parcialmente definidos que eventualmente producen un valor.

  Las características clave de TensorFlow son mejores visualizaciones de gráficos computacionales, reducción de errores en un 50 a 60 por ciento en el aprendizaje automático neuronal, computación paralela para ejecutar modelos complejos, gestión de bibliotecas sin problemas respaldada por Google y actualizaciones más rápidas y nuevas versiones frecuentes para proporcionar las últimas características.

  TensorFlow sirve para el reconocimiento de voz e imagen, aplicaciones basadas en texto, análisis de series de tiempo y detección de video.

- **PyTorch**: es considerada una de las mejores bibliotecas de aprendizaje automático y aprendizaje profundo y compite fuertemente con Tensor Flow. De código abierto y popular basada en Torch, que es otra biblioteca de código abierto de aprendizaje automático implementada en C con un envoltorio en Lua.

  Ofrece una amplia variedad de herramientas y bibliotecas que admiten programas de visión por computadora, procesamiento de lenguaje natural (NLP) y muchos otros programas de aprendizaje automático. Permite a los desarrolladores realizar cálculos en tensores con aceleración de GPU y también ayuda a crear gráficos computacionales.

3.1.4.- Interfaz de Programación de Aplicaciones (API)

Django ofrece la posibilidad de incorporar APIs a la aplicación en cuestión. Su finalidad es permitir la comunicación y el intercambio de datos entre Django y otras aplicaciones o sistemas externos, facilitando la integración, el desarrollo de aplicaciones cliente y el acceso a datos o funcionalidades específicas.

- **Django REST** framework [3.4] es un conjunto de herramientas para construir aplicaciones web API utilizando el framework Django de Python. Permite la creación de API RESTful que se comunican con aplicaciones web y móviles.

  Para el Análisis de Datos, Django REST framework proporciona una forma fácil y rápida de crear una API REST que pueda exponer los datos almacenados en una base de datos. Los datos se pueden extraer, filtrar y manipular utilizando una variedad de herramientas de Análisis de Datos de Python, como pandas, NumPy, Scikit-learn y otros.

  Además, Django REST framework proporciona una amplia gama de herramientas para autenticación, autenticación de tokens, paginación, pruebas, documentación y más, lo que lo hace muy útil para el Análisis de Datos en proyectos de desarrollo web. Con su soporte para formatos de intercambio de datos estándar como JSON y XML, Django REST framework hace que la integración de una API sea fácil y flexible, lo que permite a los desarrolladores de Análisis de Datos trabajar de manera más eficiente con grandes conjuntos de datos.

- **FastAPI** [3.5] es un framework web de alto rendimiento y fácil de usar para construir APIs en Python. Es conocido por su enfoque en la velocidad y la facilidad de desarrollo, lo que lo hace ideal para proyectos en los que se requiere un alto rendimiento.

  FastAPI utiliza la tipificación de Python para ayudar a los desarrolladores a detectar errores en tiempo de compilación, lo que facilita el desarrollo de aplicaciones sin errores. También incluye características como el soporte para la documentación automática de una API, la validación de esquemas JSON y la generación automática de código cliente.

  En cuanto a su uso para el Análisis de Datos, FastAPI se puede utilizar para construir APIs que permitan a los usuarios interactuar con sus datos de diversas maneras. Por ejemplo, se puede utilizar para construir una API que permita a los usuarios cargar sus datos en una base de datos, realizar análisis sobre esos datos y luego visualizar los resultados de manera interactiva.

FastAPI también se integra bien con bibliotecas populares de Análisis de Datos como NumPy, Pandas y Scikit-learn, lo que facilita la construcción de aplicaciones de Análisis de Datos de alto rendimiento y escalables.

3.2.- SHINY

Shiny [2.33] es un paquete de R que permite crear aplicaciones interactivas y dinámicas directamente desde este lenguaje. Desarrollado por RStudio es ampliamente utilizado en el entorno de la Ciencia de Datos y el análisis estadístico, popularidad que ha logrado combinando la capacidad de realizar análisis avanzados con una interfaz intuitiva para los usuarios.

Se basa en el paradigma de la programación reactiva, lo que significa que los elementos de la interfaz y los valores en el servidor se actualizan automáticamente en respuesta a las interacciones de los usuarios. Esto facilita la creación de aplicaciones web en las que los resultados y visualizaciones se actualizan dinámicamente a medida que los usuarios interactúan con los widgets y los datos.

La principal ventaja de Shiny es que permite aprovechar todo el poder y la flexibilidad de R para realizar análisis y cálculos complejos, y al mismo tiempo proporcionar una interfaz intuitiva y amigable para que los usuarios interactúen con los datos y resultados.

Cuenta con una amplia comunidad de usuarios y una gran cantidad de recursos disponibles, como tutoriales, ejemplos y paquetes complementarios. Esto facilita el aprendizaje y la implementación de aplicaciones Shiny, ya que los desarrolladores pueden aprovechar el conocimiento compartido y las mejores prácticas.

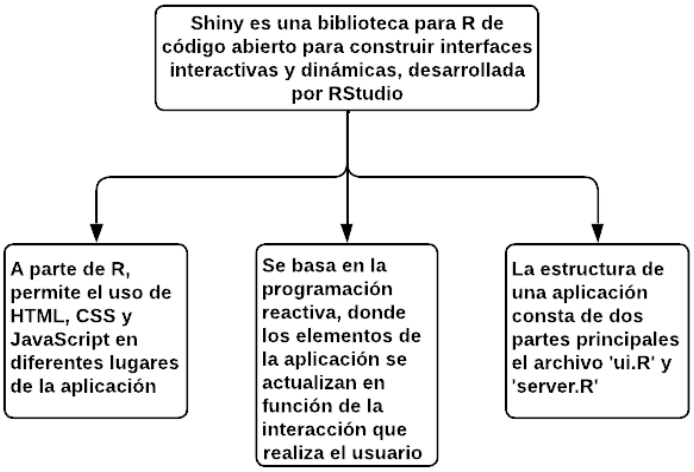


Figura 3.4: Resumen sobre qué es Shiny.

3.2.1.- Lenguajes de programación

R [3.6][3.7] es un lenguaje de programación y un entorno de software ampliamente utilizado en el ámbito de la estadística, el Análisis de Datos y la Ciencia de Datos. Fue desarrollado originalmente por Robert Gentleman y Ross Ihaka en los años 90 en la Universidad de Auckland, Nueva Zelanda.

Una de las principales fortalezas de R es su amplia colección de paquetes y librerías especializadas, que cubren una amplia variedad de áreas como análisis estadístico, Minería de Datos, aprendizaje automático, visualización y mucho más. Estos paquetes permiten a los usuarios acceder a una amplia gama de algoritmos y técnicas avanzadas de manera sencilla.

Ofrece una sintaxis intuitiva y expresiva, lo que facilita la manipulación y transformación de datos, así como la creación de gráficos y visualizaciones. Además, es un lenguaje interpretado, lo que permite una interacción rápida y flexible con los datos y los resultados.

El entorno de desarrollo de R incluye una consola interactiva en la que se pueden ejecutar comandos, así como una interfaz gráfica de usuario (GUI) llamada RStudio, que proporciona una experiencia de desarrollo más completa con funciones como la edición de código, la gestión de archivos y la visualización de resultados.

Es ampliamente utilizado en el sector académico, la investigación y la industria debido a su capacidad para realizar análisis estadísticos complejos, modelado de datos, visualizaciones interactivas y la posibilidad de generar informes y presentaciones profesionales. Además, su naturaleza de código abierto y su comunidad activa hacen que R sea una opción popular para aquellos que buscan colaboración y recursos adicionales.

R se distribuye bajo una licencia de software libre llamada GNU General Public License (GPL). La GPL es una licencia de copyleft, lo que significa que permite a los usuarios utilizar, modificar y distribuir el software de forma gratuita, siempre y cuando se respeten ciertas condiciones. Esta licencia garantiza la colaboración y el intercambio de conocimientos entre la comunidad de usuarios de R, fomentando el desarrollo y la mejora continua del software. Además, proporciona protección para los derechos de los usuarios y promueve la transparencia en el uso y distribución del software.

Pese a que Shiny se basa principalmente en R, también admite la integración de otros lenguajes para ampliar su funcionalidad como: HTML, CSS, JavaScript y Python.

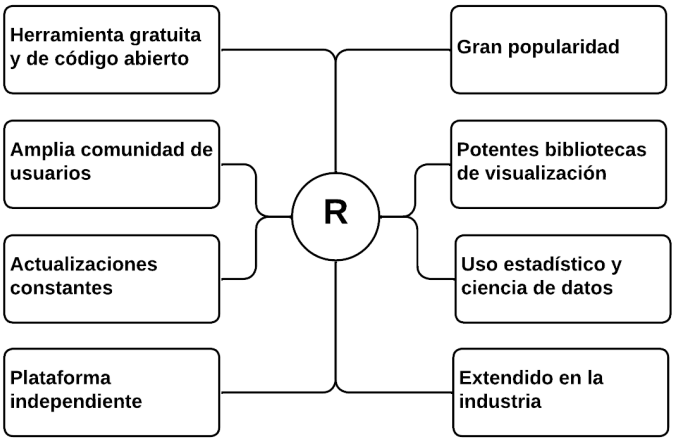


Figura 3.5: Principales características de R.

3.2.2.- Elementos de su arquitectura

Shiny está compuesto por dos elementos principales, los cuales son: la interfaz de usuario (IU) y la parte del servidor. Son dos componentes fundamentales que trabajan conjuntamente para construir una aplicación web interactiva.

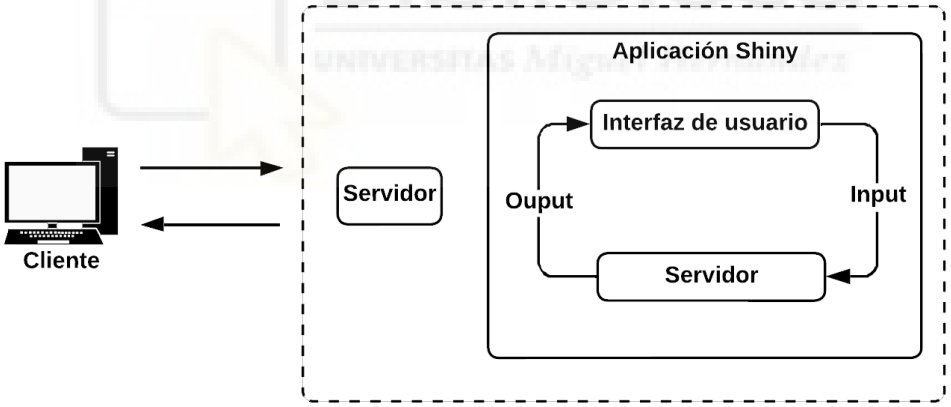


Figura 3.6: Elementos principales de Shiny.

La interfaz de usuario gestiona cómo se muestra y organiza visualmente la aplicación. Se define utilizando funciones y objetos de R que permiten diseñar la apariencia.

Los componentes que se pueden definir en la IU son:

- **Controles de entrada**: elementos interactivos como campos de texto, botones, deslizadores, listas desplegables y casillas de verificación. Permiten al usuario interactuar con la aplicación y proporcionar datos de entrada.

- **Salidas**: áreas donde se muestran los resultados o visualizaciones generadas por la aplicación. Pueden ser gráficos, tablas, informes, mapas u otros elementos visuales que se actualizan en respuesta a las acciones del usuario o cambios en los datos.

- **Diseño y disposición**: funciones para estructurar y organizar los elementos de la interfaz de usuario. Esto incluye la creación de paneles, pestañas y diseños de cuadrícula para una presentación visual adecuada.

Respecto al servidor, es la parte donde se define la lógica y el comportamiento de la aplicación. Se encarga de procesar las entradas del usuario, realizar cálculos y generar las salidas correspondientes.

Las acciones que realizar el servidor son:

- **Procesamiento de datos**: operaciones como filtrado, manipulación y agregación de datos. Esto permite adaptar los datos de entrada para su análisis y visualización.

- **Análisis y modelado estadístico**: código R para realizar análisis estadísticos, ajustar modelos, realizar predicciones u otras tareas analíticas.

- **Actualización reactiva**: el servidor puede enviar actualizaciones a la interfaz de usuario en tiempo real en respuesta a las acciones del usuario o cambios en los datos de entrada.

3.2.3.- Patrón de diseño

Si bien es cierto que Shiny es una biblioteca para extender la funcionalidad de R, y no un framework como tal, salvando las distancias, podemos enmarcar a Shiny dentro del patrón de diseño Modelo-Vista-Controlador (MVC).

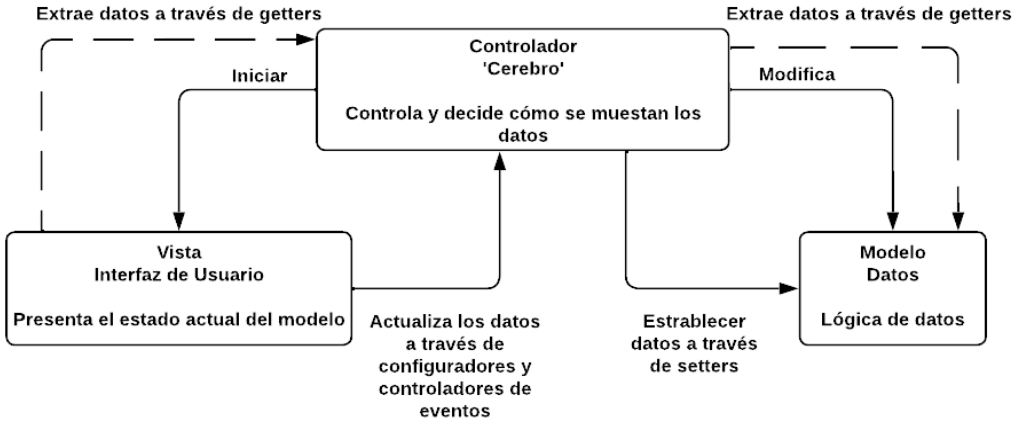


Figura 3.7: Funcionamiento del patrón de diseño Modelo-Vista-Controlador.

En el contexto de esta herramienta el patrón de diseño está organizado de la siguiente manera:

- **Modelo**: representa los datos y la lógica de la aplicación. En Shiny esta parte incluye el procesamiento de datos, análisis estadístico, modelado u otras operaciones relacionadas con los datos subyacentes.

- **Vista**: es la interfaz de usuario. Representa cómo se expone la información al usuario y cómo interactúa éste con la aplicación. La vista se define utilizando objetos y funciones de R que definen la apariencia, como los controles de entrada y las salidas.

- **Controlador**: actúa como un intermediario entre el modelo y la vista. Se encarga de manejar las interacciones del usuario y coordinar las acciones correspondientes en el modelo y la vista. En Shiny, el controlador está implícito en el entorno del servidor, donde se define la lógica y el comportamiento de la aplicación.

3.2.3.- Librerías

Las librerías que se describen a continuación: shiny, shinythemes, DT, plotly, ggplot2, arules, arulesViz, rpart.plot, caret, caTools y shinyjs, son las más representativas, entre otras, para el Análisis y Visualización de Datos en Shiny, sin embargo, esta herramienta dispone de muchas más librerías para diferentes objetivos.

- **Shiny**: es una librería fundamental para la construcción de aplicaciones web interactivas en R. Proporciona una forma sencilla de crear interfaces de usuario (UI) interactivas y conectarlas con el código de R en el servidor. Shiny permite desarrollar aplicaciones web para el Análisis y Visualización de Datos sin la necesidad de conocimientos avanzados en tecnologías web.

- **Shinythemes**: extensión de Shiny que proporciona una variedad de temas y estilos visuales predefinidos para personalizar la apariencia de las aplicaciones Shiny. Permite cambiar fácilmente la apariencia de los elementos de la UI, como colores, fuentes y diseños.

- **DT**: esta librería, que significa "DataTables", ofrece herramientas para crear y personalizar tablas interactivas en R. Permite mostrar y explorar datos tabulares de forma flexible y dinámica. DT proporciona opciones de filtrado, ordenamiento, paginación y búsqueda en las tablas, así como características de resaltado y formato personalizado.

- **Plotly**: librería de visualización interactiva que permite crear gráficos dinámicos y atractivos en R. Proporciona una amplia variedad de tipos de gráficos, como gráficos de dispersión, barras, líneas, superficies y más. Plotly permite la interacción con los gráficos, como zoom, selección de puntos y exportación de imágenes.

- **ggplot2**: es una librería de visualización ampliamente utilizada en R. Ofrece una gramática de gráficos que permite construir de manera fácil y concisa una amplia gama de gráficos estadísticos. Ggplot2 se basa en la idea de "capas" para agregar componentes como puntos, líneas, barras y etiquetas a los gráficos, lo que facilita la creación de visualizaciones complejas y personalizadas.

- **arules**: librería de Minería de Datos para el análisis de reglas de asociación en conjuntos de datos transaccionales. Permite descubrir patrones y relaciones frecuentes en datos de transacciones, como, por ejemplo, las compras de los clientes. Arules ofrece herramientas para la generación de reglas de asociación, la evaluación de su calidad y el soporte para operaciones de manipulación y visualización de reglas.

- **arulesViz**: es una extensión de la librería arules que proporciona funciones y métodos para visualizar y explorar reglas de asociación. Permite crear gráficos y diagramas interactivos para mostrar patrones de asociación y evaluar su calidad. arulesViz facilita la interpretación y comprensión de las reglas de asociación mediante representaciones visuales efectivas.

- **rpart.plot**: es una librería para visualizar árboles de decisión creados con la librería rpart. Proporciona métodos para representar de forma gráfica la estructura y las divisiones en un árbol de decisión. Rpart.plot ofrece opciones para personalizar la apariencia del árbol, como colores, tamaños de nodo y etiquetas.

- **caret**: (Classification And REgression Training) es una librería utilizada para entrenar y evaluar modelos de aprendizaje automático en R. Proporciona una interfaz unificada para diferentes algoritmos de clasificación y regresión, y permite realizar tareas como selección de características, ajuste de hiper parámetros y validación cruzada de forma sencilla.

- **caTools**: librería que ofrece una colección de funciones para realizar diversas tareas de Análisis de Datos en R. Incluye funciones útiles para el muestreo aleatorio, la manipulación de fechas y horas, el redondeo de números, la codificación de variables categóricas y más.

- **shinyjs**: es una librería que proporciona funciones y métodos adicionales para controlar y manipular elementos de la UI en aplicaciones Shiny. Permite realizar acciones como mostrar u ocultar elementos, habilitar o deshabilitar controles, cambiar propiedades de estilo y realizar animaciones en la UI de forma dinámica.

3.3.- DEDUCCIONES SOBRE LAS HERRAMIENTAS

Tras un estudio y análisis más detallado de las tecnologías Django y Shiny, se ha determinado que ambas serán empleadas de manera conjunta en el desarrollo de la aplicación web destinada al Análisis y Visualización de Datos. Esta decisión se basa en las siguientes ventajas y características particulares que cada una ofrece.

Se ha identificado que Django, gracias a su versatilidad como framework de desarrollo web, escrito en el lenguaje multipropósito Python, permite estructurar la aplicación de manera ordenada y eficiente, simplificando el proceso de desarrollo y mantenimiento, a través del patrón de diseño Modelo-Vista-Plantilla.

Uno de los puntos clave para simular la seguridad de la aplicación es la inclusión de una página de login para proteger el acceso a los datos y funcionalidades. En este aspecto, Django ofrece mecanismos robustos para autenticación y autorización.

Por otro lado, la potencia de Shiny en el ámbito de la visualización y Análisis de Datos interactivos, desarrollada en R, proporciona una plataforma sólida para crear visualizaciones atractivas e interactivas.

Con R, se obtiene acceso a una amplia gama de paquetes y bibliotecas especializadas en análisis estadístico y Ciencia de Datos. Esto brinda la posibilidad de implementar técnicas de visualización y manipulación de datos, que enriquecen la experiencia del usuario y potencian la obtención de resultados para la Toma de Decisiones informadas.

La elección de embeber y combinar Shiny en Django se fundamenta en las restricciones temporales y objetivos inicialmente mencionadas en el proyecto. La curva de aprendizaje de Shiny es considerablemente más suave que la de Django, lo que acelera significativamente el desarrollo de la aplicación. De optar únicamente por Django, el tiempo requerido es notablemente mayor, mientras que una elección exclusiva de Shiny deja la parte de autenticación desatendida o insuficientemente abordada. Por consiguiente, la unión de ambas tecnologías representa la solución óptima, permitiendo aprovechar lo mejor de las dos herramientas y satisfacer plenamente los objetivos del proyecto en el tiempo establecido.

Capítulo 4
Metodología y resultados

El desarrollo de software debe estar respaldado por una serie de técnicas y herramientas que permitan asistir a los desarrolladores a lo largo de este proceso sistemático. En este capítulo se plasma la metodología seguida para llevar a cabo las diferentes etapas de la ingeniería del software.

Concretamente, el objetivo propuesto es la creación de una aplicación web utilizando el framework Django junto con Shiny R para la programación y testeo de módulos de Análisis y Visualización de Datos. Debido al alcance limitado que presenta este trabajo en su conjunto, la aplicación a desarrollar seguirá la metodología que se espera de la ingeniería del software, pero no resultará en una aplicación web profesional, ya que el objetivo principal es demostrar la funcionalidad del aplicativo con características limitadas.

4.1.- PLANIFICACIÓN DEL PROYECTO

Para la planificación del proyecto se ha seleccionado el ciclo de vida iterativo en espiral [4.1]. Fue creado originalmente por Boehm en 1988. En lugar de presentar el proceso de software como una serie lineal de actividades, se representa como una espiral. Cada vuelta de la espiral simboliza una fase del proceso de software. Por ejemplo, el ciclo más cercano al centro podría referirse a la evaluación de la viabilidad del sistema, el siguiente ciclo a la definición de los requerimientos, el siguiente ciclo al diseño del sistema, y así sucesivamente.

La espiral se divide en cuatro secciones que representan distintas fases del proceso:

1. **Definición de objetivos**: en esta fase se establecen los objetivos específicos del proyecto, se identifican las limitaciones del proceso y el producto, y se elabora un plan detallado de gestión. También se identifican los posibles riesgos del proyecto y se planean estrategias alternativas para hacerles frente.

2. **Evaluación y reducción de riesgos**: se realiza un análisis minucioso de cada uno de los riesgos del proyecto identificados y se establecen pasos para reducirlos. Por ejemplo, si existe el riesgo de tener requerimientos inapropiados, se puede desarrollar un prototipo del sistema para mitigar dicho riesgo.

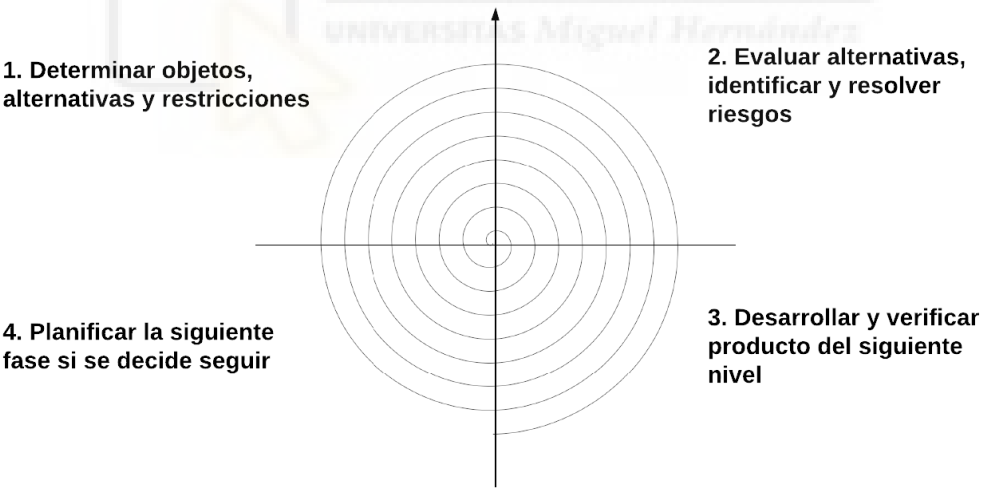


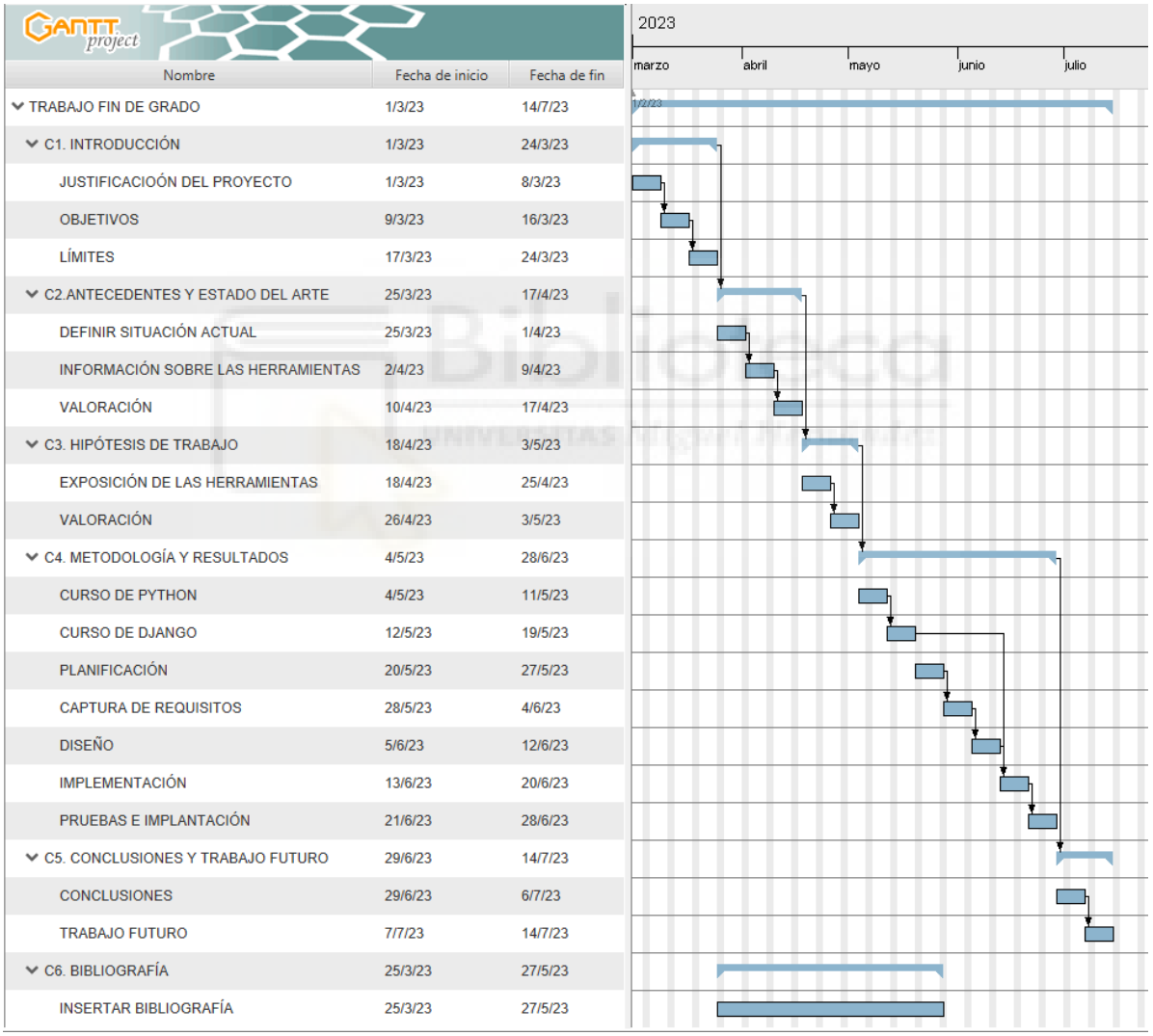
Figura 4.1: Funcionamiento del modelo en espiral.

3. **Desarrollo y validación:** se elige un modelo de desarrollo adecuado después de evaluar los riesgos. Por ejemplo, si los riesgos se concentran en la interfaz de usuario, un modelo apropiado podría ser la construcción de prototipos evolutivos. Si la seguridad es la principal preocupación, entonces un modelo basado en transformaciones formales podría ser más adecuado.

4. **Planificación:** se revisa el proyecto y se decide si se debe continuar con el siguiente ciclo de la espiral. Si se decide avanzar, se elaboran los planes necesarios para la siguiente fase del proyecto.

El modelo iterativo en espiral es recomendado, a contraposición del modelo secuencial en cascada, porque permite una mayor flexibilidad, control de riesgos, calidad y participación del cliente.

A través del siguiente diagrama de Gantt se muestra la evolución temporal del proyecto y la relación existente entre las diferentes fases y subtareas, que componen en su conjunto la totalidad del proyecto, desde su inicio hasta su finalización.



Nombre	Fecha de inicio	Fecha de fin
TRABAJO FIN DE GRADO	1/3/23	14/7/23
C1. INTRODUCCIÓN	1/3/23	24/3/23
JUSTIFICACIOÓN DEL PROYECTO	1/3/23	8/3/23
OBJETIVOS	9/3/23	16/3/23
LÍMITES	17/3/23	24/3/23
C2.ANTECEDENTES Y ESTADO DEL ARTE	25/3/23	17/4/23
DEFINIR SITUACIÓN ACTUAL	25/3/23	1/4/23
INFORMACIÓN SOBRE LAS HERRAMIENTAS	2/4/23	9/4/23
VALORACIÓN	10/4/23	17/4/23
C3. HIPÓTESIS DE TRABAJO	18/4/23	3/5/23
EXPOSICIÓN DE LAS HERRAMIENTAS	18/4/23	25/4/23
VALORACIÓN	26/4/23	3/5/23
C4. METODOLOGÍA Y RESULTADOS	4/5/23	28/6/23
CURSO DE PYTHON	4/5/23	11/5/23
CURSO DE DJANGO	12/5/23	19/5/23
PLANIFICACIÓN	20/5/23	27/5/23
CAPTURA DE REQUISITOS	28/5/23	4/6/23
DISEÑO	5/6/23	12/6/23
IMPLEMENTACIÓN	13/6/23	20/6/23
PRUEBAS E IMPLANTACIÓN	21/6/23	28/6/23
C5. CONCLUSIONES Y TRABAJO FUTURO	29/6/23	14/7/23
CONCLUSIONES	29/6/23	6/7/23
TRABAJO FUTURO	7/7/23	14/7/23
C6. BIBLIOGRAFÍA	25/3/23	27/5/23
INSERTAR BIBLIOGRAFÍA	25/3/23	27/5/23

Figura 4.2: Diagrama de Gantt del proyecto.

4.2.- CAPTURA DE REQUISITOS

Los roles presentes en el sistema de la aplicación web para el Análisis y Visualización de Datos son: “System operator”, “Manager”, “Teacher” y “Student”.

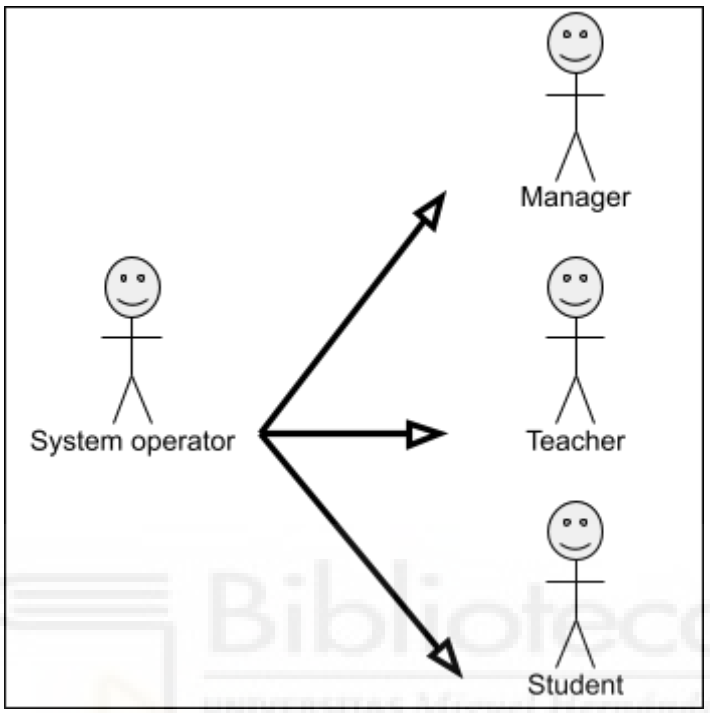


Figura 4.3: Jerarquía de usuarios y sus roles.

Como se puede apreciar en la figura anterior el “System operator” está en la cúspide de la jerarquía y además de desempeñar las acciones propias de su rol, puede llevar a cabo las acciones de los roles que contiene a su derecha.

Es importante mencionar que el desarrollo de la aplicación es un prototipo y no se tiene como objetivo (en el marco del TFG) implantarla de manera completa. Esto desemboca en que la descripción de los usuarios se realiza respecto a sus definiciones generales de rol, sin añadir las tareas específicas de cada uno de los actores dentro de la aplicación. Por ejemplo, del “Manager” se obvia que puede asignar roles y permisos a otros usuarios, gestionar el contenido y la organización de cursos y tomar decisiones estratégicas relacionadas con el proceso educativo, ya que esto implica la implementación de funcionalidades que por límite de tiempo no se añaden en el proyecto.

Pese a esto, en el proyecto se incluyen los casos de uso fundamentales para garantizar el funcionamiento de la aplicación y alcanzar los objetivos propuestos.

A continuación, en cada tabla se expone a cada uno de dichos roles de la aplicación.

Tabla 4.1: Definición del rol “System operator”.

Usuario	*System operator*
Descripción	Encargado de administrar y mantener en funcionamiento la infraestructura y componentes del sistema.
Casos de uso	*CU-1, CU-2, CU-3, CU-4, CU-5 y CU-6*

Tabla 4.2: Definición del rol “Manager".

Usuario	*Manager*
Descripción	Es responsable de la gestión y coordinación de actividades, recursos y personal dentro de un ámbito específico.
Casos de uso	*CU-1, CU-2, CU-3, CU-4 y CU-5*

Tabla 4.3: Definición del rol “Teacher".

Usuario	*Teacher*
Descripción	Educador o instructor, creando y administrando cursos, evaluaciones y proporcionando guía educativa a los estudiantes.
Casos de uso	*CU-1, CU-2, CU-3, CU-4 y CU-5*

Tabla 4.4: Definición del rol “Student".

Usuario	*Student*
Descripción	Participa en el proceso de aprendizaje, adquiriendo conocimientos y habilidades.
Casos de uso	*CU-1, CU-2, CU-3, CU-4 y CU-5*

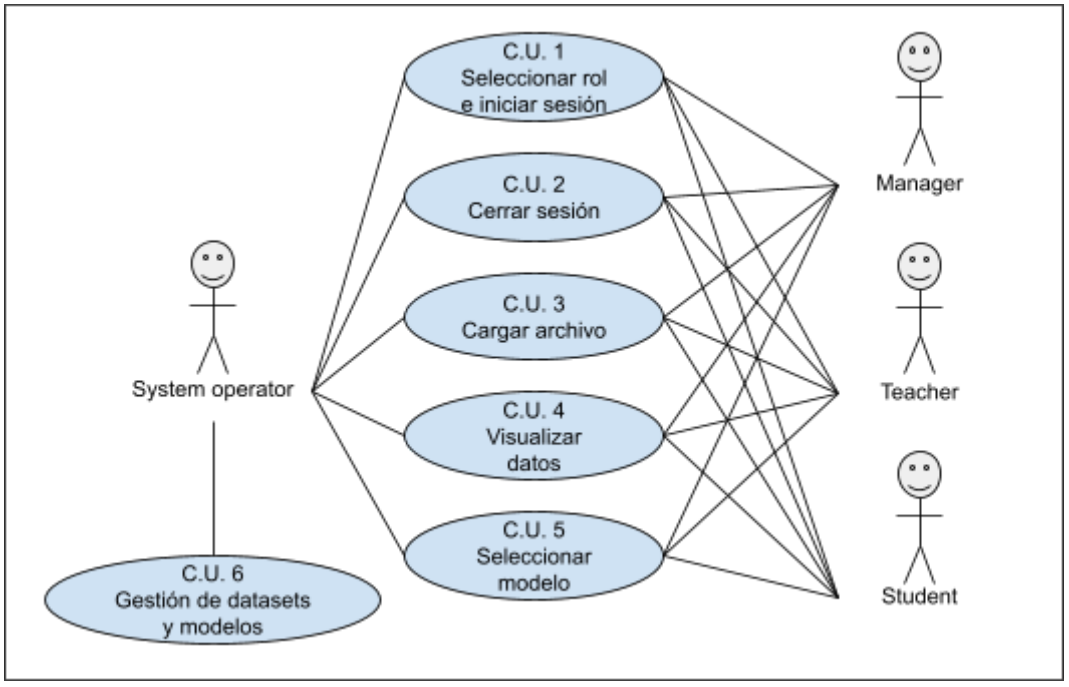


Figura 4.4: Roles de usuario y los casos de uso que pueden desempeñar.

Mediante el uso de tablas se especifica con más detalle los diferentes casos de uso.

Tabla 4.5: Caso de uso 1 “Seleccionar rol e iniciar sesión”.

C.U. 1	*Seleccionar rol e iniciar sesión*
Actores	*System operator, Manager, Teacher y Student*
Descripción	Opción de la aplicación para escoger el rol de usuario e iniciar sesión en la aplicación.
Dependencias	
Precondición	
Secuencia normal	P1 - Seleccionar el rol de usuario P2 - Introducir credenciales P3 - Pulsar el botón “Access”
Poscondición	El usuario estará identificado en la aplicación.
Excepciones	Si en P3 las credenciales son incorrectas, aparecerá un mensaje avisando al usuario.
Comentarios	

Tabla 4.6: Caso de uso 2 “Cerrar sesión”.

C.U. 2	*Cerrar sesión*
Actores	*System operator, Manager, Teacher y Student*
Descripción	Opción de la aplicación para cerrar sesión.
Dependencias	CU-1
Precondición	Haber ejecutado el CU-1.
Secuencia normal	P1 - Pulsar el botón “Logout” de la aplicación
Poscondición	El usuario habrá cerrado sesión en la aplicación y será redirigido a la página de inicio de sesión.
Excepciones	
Comentarios	

Tabla 4.7: Caso de uso 3 “Cargar archivo”.

C.U. 3	*Cargar archivo*
Actores	*System operator, Manager, Teacher y Student*
Descripción	Opción de la aplicación que permite cargar un archivo con formato RData.
Dependencias	
Precondición	Disponer de un archivo con extensión RData.
Secuencia normal	P1 - Pulsar el botón “Browse” P2 - Escoger el archivo RData deseado P3 - Pulsar el botón “Abrir”
Poscondición	La aplicación muestra el archivo RData seleccionado.
Excepciones	Si en P3 el archivo cargado no es con formato RData aparecerá un mensaje informando al usuario.
Comentarios	El usuario podrá detener la acción de seleccionar un archivo pulsando el botón “Cancelar”.

Tabla 4.8: Caso de uso 4 “Visualizar datos”.

C.U. 4	*Visualizar datos*
Actores	*System operator, Manager, Teacher y Student*
Descripción	Opción de la aplicación para visualizar los datos que contiene el archivo seleccionado.
Dependencias	CU-3
Precondición	Haber ejecutado el CU-3.
Secuencia normal	P1 - Pulsar la opción “Data preview” P2 - Escoger el tipo de visualización P3 - Seleccionar las variables
Poscondición	La aplicación mostrará en función del gráfico seleccionado los datos que contiene el archivo.
Excepciones	Si en P3 las variables seleccionadas no generan un resultado, a través del algoritmo, la salida se queda en blanco para evitar gráficas y/o cálculos indeseados.
Comentarios	El usuario podrá seleccionar en todo momento las variables que desee y los resultados se actualizarán en tiempo real.

Tabla 4.9: Caso de uso 5 “Seleccionar modelo”.

C.U. 5	*Seleccionar modelo*
Actores	*System operator, Manager, Teacher y Student*
Descripción	Opción de la aplicación para escoger un modelo con el cual procesar los datos.
Dependencias	CU-3
Precondición	Haber ejecutado el CU-3
Secuencia normal	P1 - Pulsar la opción “Models” P2 - Escoger el modelo deseado P3 - Seleccionar las variables
Poscondición	La aplicación habrá procesado los datos con el modelo seleccionado.
Excepciones	Si en P3 las variables seleccionadas no generan un resultado, a través del algoritmo, la salida se queda en blanco para evitar gráficas y/o cálculos indeseados .
Comentarios	El usuario podrá seleccionar en todo momento las variables que desee y los resultados se actualizarán en tiempo real.

Tabla 4.10: Caso de uso 6 “Gestión de datasets y modelos”.

C.U. 6	*Gestión de datasets y modelos*
Actores	*System operator*
Descripción	El usuario tiene el poder de acceder al código fuente para realizar modificaciones y mantenimientos de la aplicación. También puede generar archivos RData para su posterior uso.
Dependencias	
Precondición	Realizar una copia de seguridad del estado actual del sistema.
Secuencia normal	P1 - Detener el servidor P2 - Acceder al código fuente y/o a RStudio P3 - Realizar las modificaciones y/o generación de datasets P4 - Guardar cambios P5 - Poner en marcha nuevamente el servidor
Poscondición	Se habrán realizado las tareas de mantenimiento y generación de archivos en el sistema.
Excepciones	Si el resultado obtenido no es el deseado, volver al estado anterior del sistema a través de la copia de seguridad.
Comentarios	Los privilegios para realizar dichas tareas deben ser restringidos para evitar problemas de ciberseguridad en la aplicación.

4.3.- DISEÑO

Para el diseño de la interfaz gráfica de la aplicación se ha buscado una distribución lo más sencilla y limpia posible, que ofrezca una experiencia de usuario intuitiva y sin complicaciones. Para ello, se dispone de una estructura que permite al usuario acceder a las diferentes partes del sistema de manera ágil y eficiente, sin necesidad de navegar a través de múltiples páginas.

En lugar de una navegación tradicional, se opta por implementar un menú de pestañas que permanece fijo en la parte superior de la página. Cada pestaña representa una sección específica de la aplicación, y al seleccionar una de ellas, el contenido correspondiente se muestra de manera dinámica y adaptada a la pestaña elegida.

Esta distribución coherente y consistente en el resto de la página, aporta una sensación de familiaridad y facilita la navegación para el usuario.

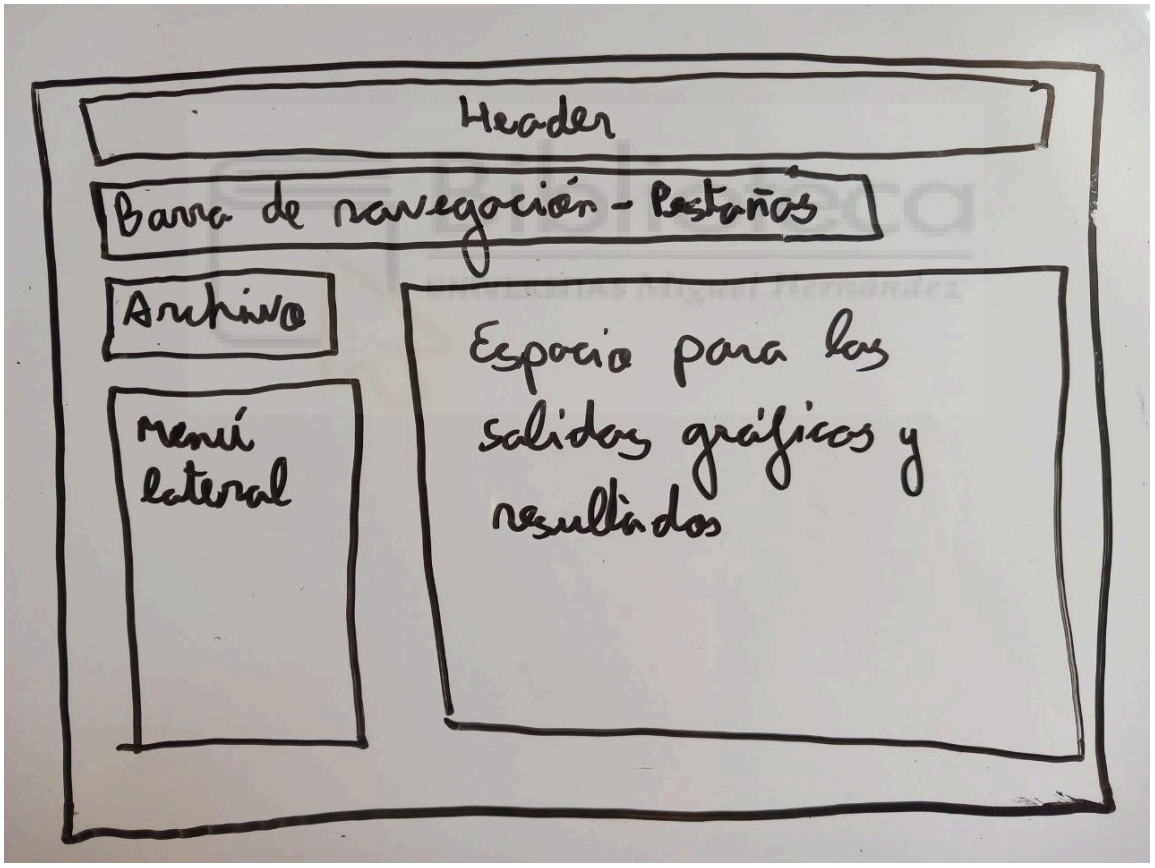


Figura 4.5: Boceto a mano alzada de la interfaz gráfica de la aplicación web.

Como la aplicación web no presenta una base de datos, por ser un prototipo, y parte de la premisa de ser accesible, el diagrama de lenguaje unificado de modelado (UML) seleccionado, para mostrar de manera global sin entrar en detalle el diseño de la aplicación, es el siguiente diagrama de flujo.

Figura 4.6: Diagrama de flujo que describe el funcionamiento general de la aplicación.

4.4.- IMPLEMENTACIÓN

El prototipo de la aplicación web desarrollada ofrece la posibilidad de elegir un rol de usuario, introducir unas claves de acceso y acceder a la aplicación para visualizar y aplicar métodos y algoritmos a un conjunto de datos almacenados en un fichero, así como el cierre de sesión.

En su conjunto la aplicación es amigable con el usuario, dispone de un sistema de ayudas y los resultados son salidas gráficas muy llamativas e interactivas.

Estas características quedan recogidas y comentadas mediante capturas de pantalla a continuación.

Para los diferentes roles de usuario encontramos los siguientes tipos: System operator, Manager, Teacher y Student. Por motivos de simplificación, se ha implementado el acceso solamente del rol como Teacher, para el resto de roles el proceso es análogo.

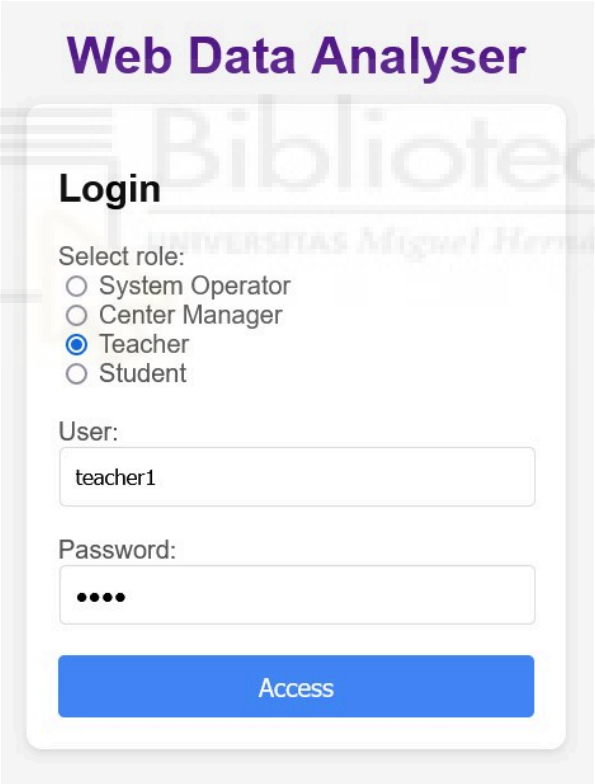


Figura 4.7: Selección de rol e inicio de sesión en la aplicación.

Una vez iniciada la sesión, en la parte superior de la página encontramos un mensaje personalizado en función del rol y nombre de usuario introducidos previamente, junto con el botón para cerrar sesión.

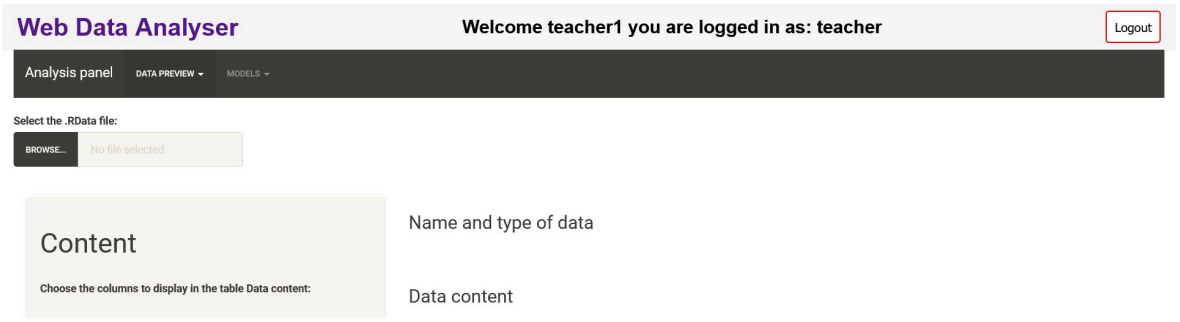


Figura 4.8: Disposición de la página tras iniciar sesión.

En el menú de navegación se sitúan dos menús desplegables llamados *Data preview* y *Models*. Cada uno de ellos con las siguientes características.

Data preview permite la visualización de los datos contenidos en el fichero que se seleccione, con las opciones:

- *Content*: muestra el contenido en formato de tablas el tipo de variables y sus valores.

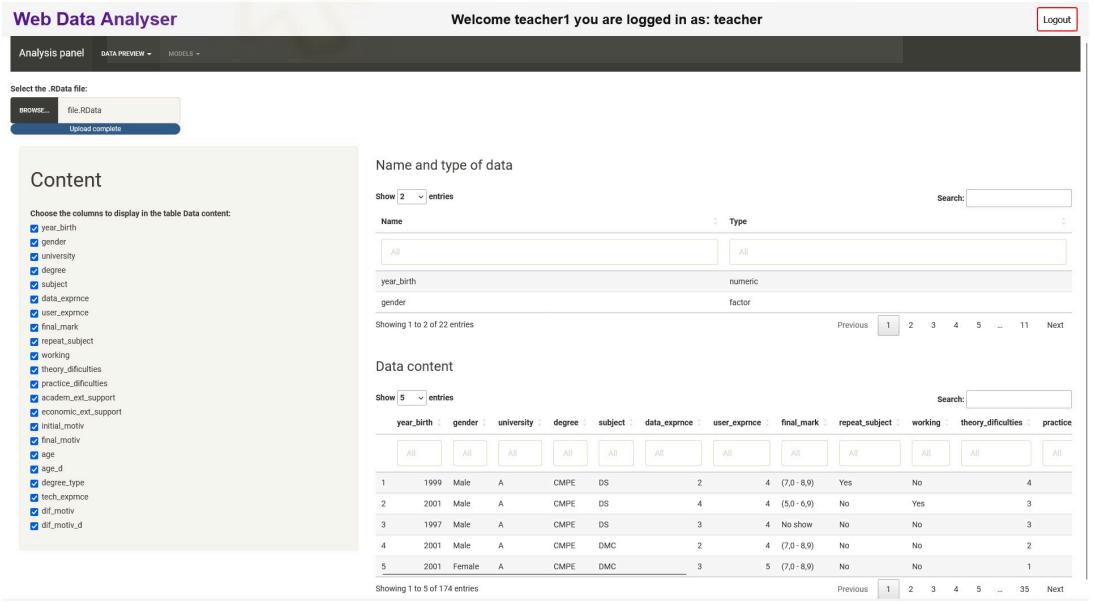


Figura 4.9: Contenido del archivo.

- *Scatter*: a través de un gráfico de dispersión se visualiza la relación entre dos variables.

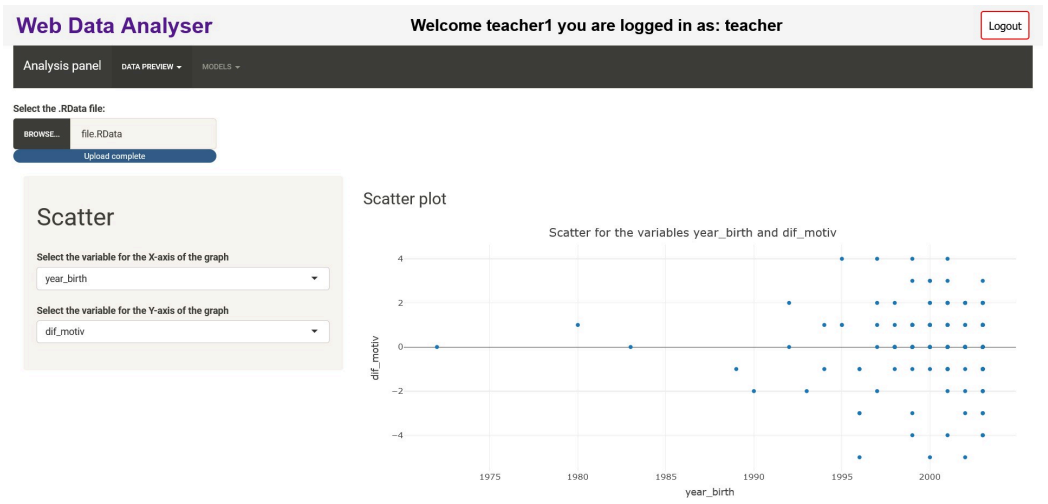


Figura 4.10: Gráfico de dispersión.

- *Histogram*: mediante la selección de una variable se genera su histograma.

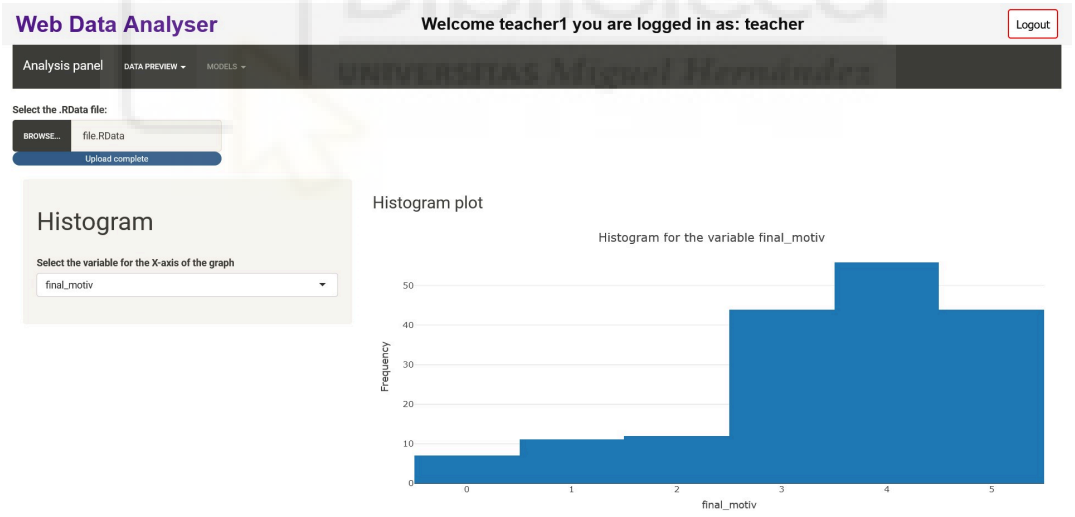


Figura 4.11: Histograma.

- *Overlapping histograms*: representa la distribución de dos variables en el mismo gráfico de histogramas.

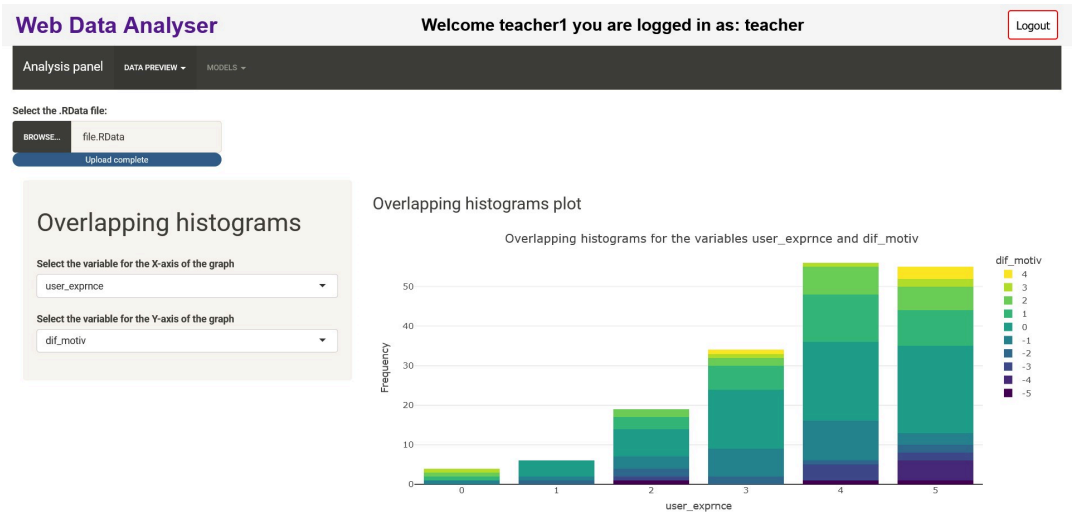


Figura 4.12: Histogramas superpuestos.

- *Box plot*: gráfico que contiene la relación de dos variables numéricas, representando la información en función de sus cuartiles.

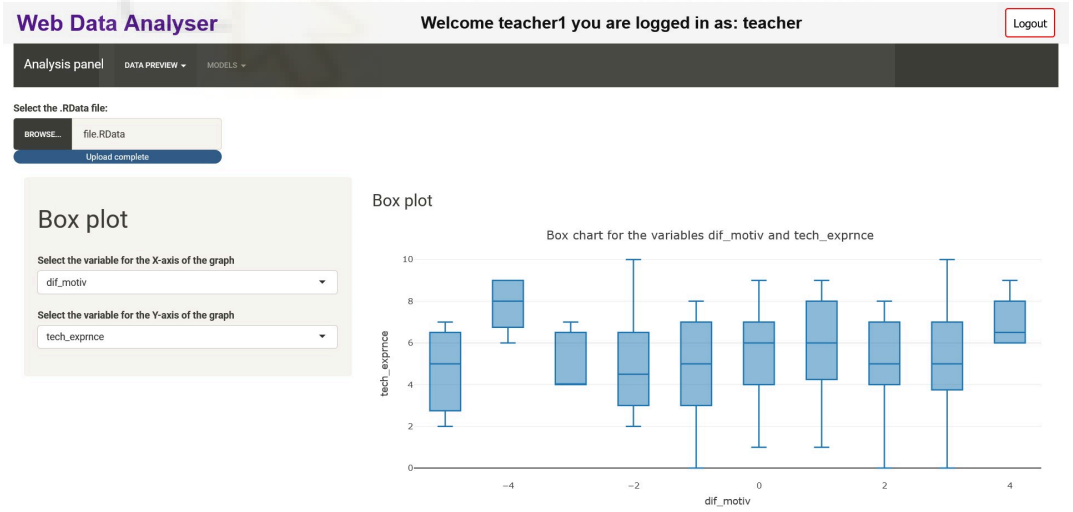


Figura 4.13: Gráfico de cajas.

- *Correlations*: dibuja una matriz escalonada de correlación entre las diferentes variables numéricas seleccionadas.

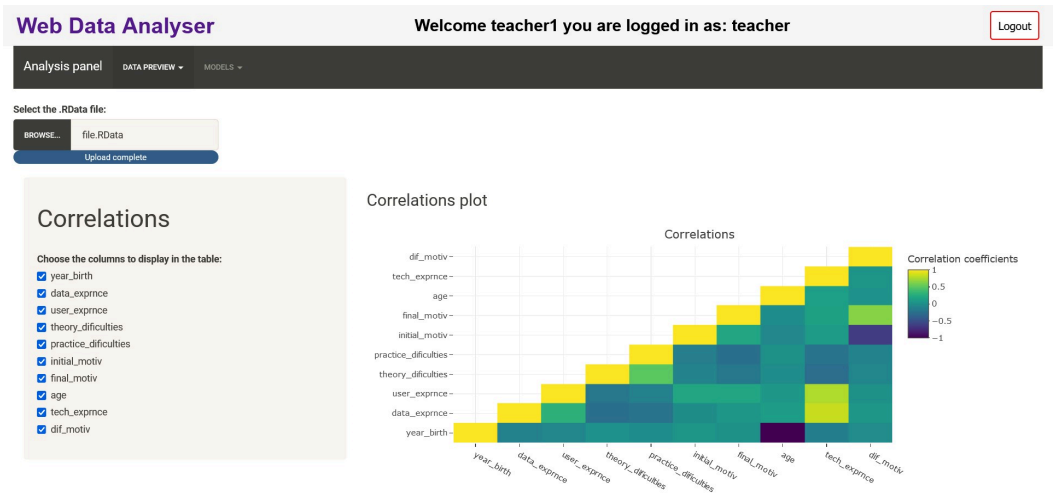


Figura 4.14: Gráfico de correlaciones.

Models ejecuta algoritmos de Machine Learning sobre las variables que se disponen en el fichero seleccionado, las opciones son:

- *Patterns - Apriori*: a través de la selección de al menos dos variables categóricas genera un gráfico en términos de soporte y confianza junto con las reglas de asociación.

Figura 4.15: Modelo de reglas de asociación.

- *Clustering - KMeans*: calcula y dibuja los grupos de pertenencia de dos variables numéricas dadas en función del número de clústers escogidos.

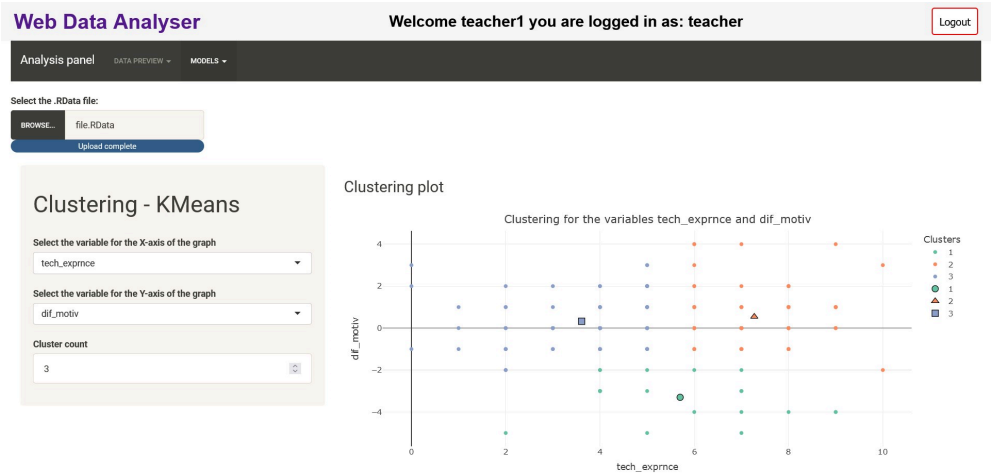


Figura 4.16: Modelo de clustering.

- *Classification - Decision tree*: genera el árbol de decisión seleccionando una variable objetivo categórica junto con las variables para los diferentes nodos del árbol. A su vez, calcula parámetros como la matriz de confusión, precisión global de la predicción, intervalo de confianza, coeficiente de Kappa, sensibilidad, etc.

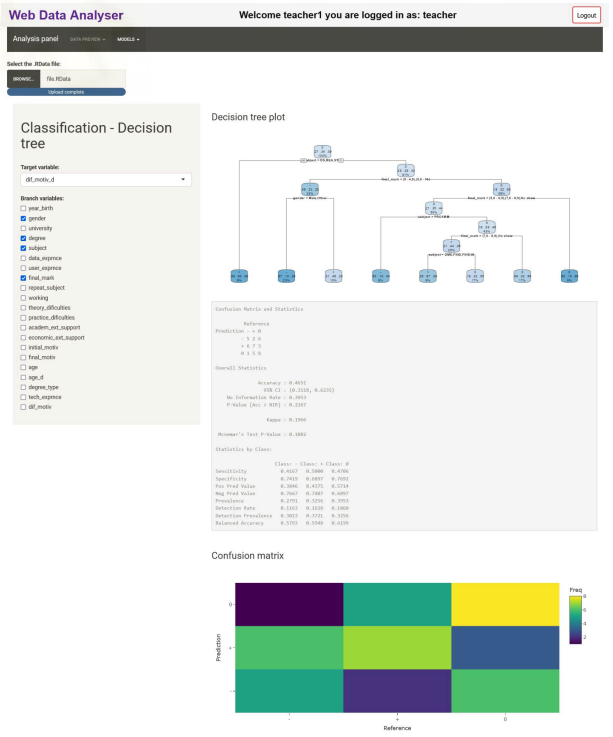


Figura 4.17: Modelo de clasificación.

Otra característica a destacar del prototipo desarrollado es la implementación de ayudas a lo largo de la aplicación para asistir al usuario. Para activarlas, el usuario puede dejar el puntero encima de los títulos de la aplicación, el puntero cambiará como se muestra en la siguiente figura.

Figura 4.18: Cursor para indicar un título de ayuda.

Al cabo de unos segundos, el texto de ayuda, se muestra y oculta automáticamente para no entorpecer la visualización con el resto de elementos de la aplicación.

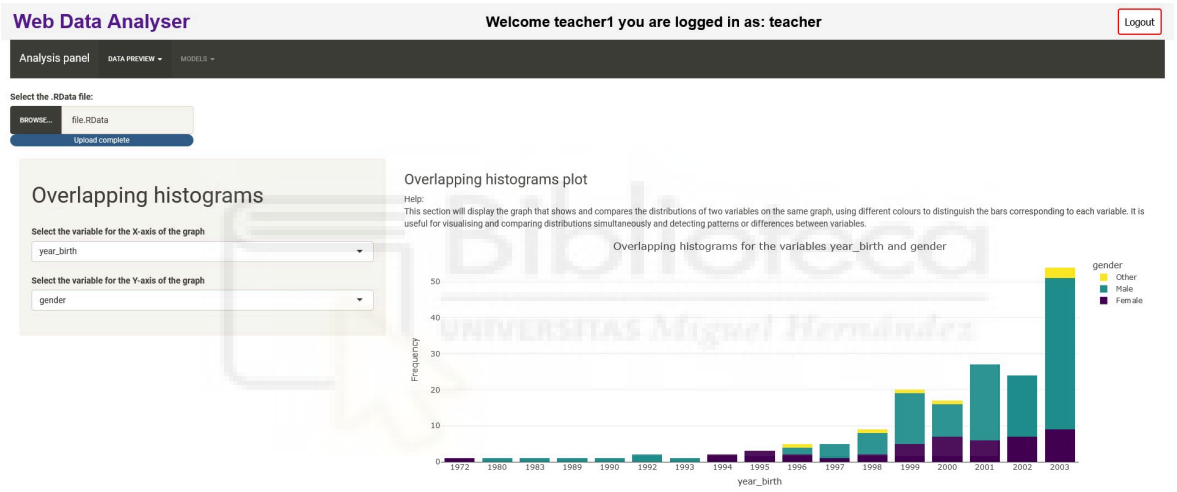


Figura 4.19: Mensaje de ayuda al usuario.

Como medida preventiva, si el usuario intenta seleccionar un tipo de archivo diferente al permitido, aparece un mensaje alertando al usuario.

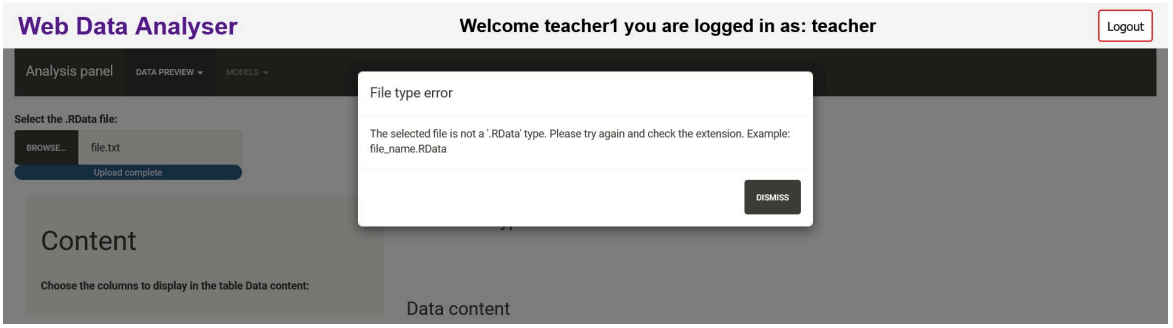


Figura 4.20: Alerta al usuario si selecciona un tipo de archivo incompatible.

4.5.- PRUEBAS

Un aspecto que se considera importante en el proyecto es el de la usabilidad y accesibilidad. Recordamos cada uno de estos conceptos mediante las definiciones formales de la *International Organization for Standardization* (ISO):

- **Usabilidad**: la medida en la que un sistema, producto o servicio se puede usar por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso especificado (ISO 9241-210:2010).

- **Accesibilidad**: usabilidad de un producto, servicio, entorno o instalación por personas con la más amplia gama de capacidades (ISO 9241-171:2008).

Para cubrir este apartado se han seguido buenas prácticas de desarrollo [4.2] como el uso de etiquetas semánticas y el diseño responsive [4.3], entre otras propiedades clave.

Si bien la accesibilidad y usabilidad son aspectos a tener en cuenta, también se reconoce la importancia de equilibrar estos objetivos con otras necesidades del proyecto. El objetivo no es alcanzar la más alta calificación en accesibilidad y usabilidad en cada sección de la aplicación, sino más bien encontrar un balance.

Es posible probar el comportamiento de la aplicación respecto a estos puntos mediante el uso de herramientas como *Siteimprove Accessibility Checker* o el lector de pantallas NVDA. Como se ha comentado anteriormente, a pesar de no obtener la mayor puntuación, en toda la aplicación se logra alcanzar el nivel A de la *Web Content Accessibility Guidelines* (WCAG).

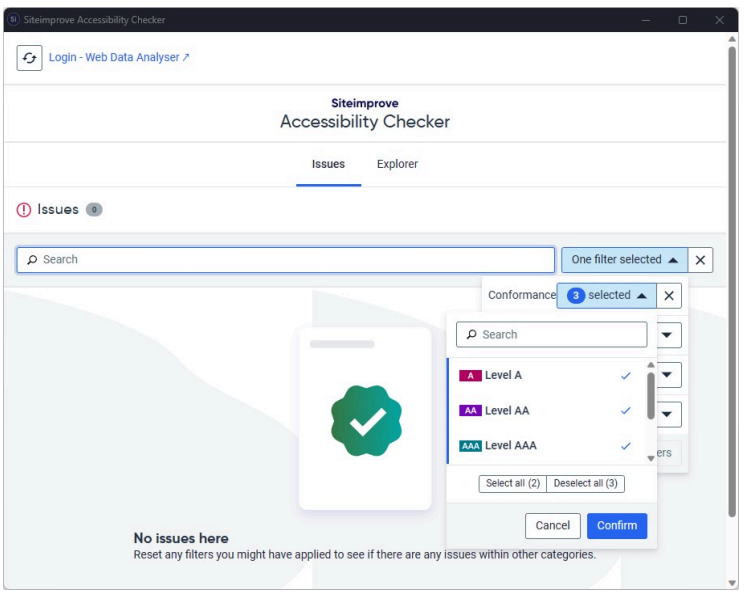


Figura 4.21: Puntuación WCAG AAA para la página de inicio de sesión.

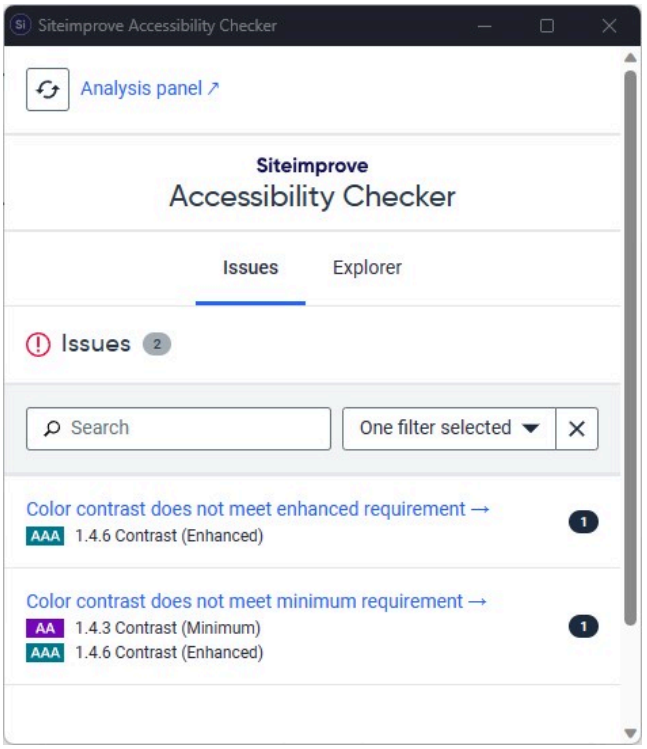


Figura 4.22: Puntuación WCAG A del contenido de análisis de datos de la aplicación.

Mencionar que, como se puede observar en la anterior figura, los errores arrojados son referidos en su totalidad a los colores y contrastes en dos partes específicas de la aplicación, esto se debe al hecho de utilizar el paquete “shinythemes” para el lenguaje R que establece por defecto una gama de colores prefijada.

La conclusión es que la accesibilidad y usabilidad de la aplicación son prácticamente óptimas en su totalidad y podrán ser mejoradas tras ser puesta en explotación para diferentes perfiles de uso.

Capítulo 5 Conclusiones y trabajo futuro

Después de haber finalizado el proyecto, en este capítulo, se exponen las conclusiones derivadas a raíz de los objetivos propuestos y el trabajo que se puede realizar cara al futuro para acabar de cumplimentar aquellas partes, que por limitaciones, no se han podido abarcar.

5.1.- CONCLUSIONES

El presente informe revela las conclusiones obtenidas a partir del trabajo de investigación centrado en el desarrollo de una aplicación web para el Análisis y Visualización de Datos. Se recuerda que, en aras de alcanzar este objetivo, se ha llevado a cabo una exploración de las diversas tecnologías disponibles para su implementación.

En primer lugar, se afirma que se han podido lograr los objetivos principales propuestos para el proyecto. Esto implica que la finalización de este trabajo ha dado lugar a la creación de una aplicación web funcional y eficiente. Uno de los logros radica en la interfaz amigable, con la

finalidad de facilitar la interacción del usuario con la plataforma. Y para garantizar esta facilidad de uso, se ha incorporado un sistema de ayudas intuitivas que proporciona orientación en cada sección de la aplicación.

Respecto a la parte referente a la seguridad, se ha implementado una página de inicio en la que, mediante la selección de un rol e introducción de unas claves, el usuario verifica su identidad para acceder a la aplicación web.

Asimismo, se ha logrado la generación de resultados visuales claros y concisos, permitiendo así una comprensión óptima de la información presentada. Conjuntamente, a través de gráficos interactivos y representaciones visuales dinámicas, los usuarios pueden explorar los datos de manera más efectiva y extraer conclusiones significativas.

En segundo lugar, se ha abordado el objetivo secundario, pero no por ellos menos importante, de realizar una investigación para obtener una vista panorámica de las tecnologías actualmente disponibles, con el propósito de evaluar su aplicabilidad en la Ciencia de Datos y la creación de una aplicación web destinada al Análisis y Visualización de Datos. Se escogieron entre todas las existentes: React, Django, Angular, Plotly Dash y Shiny R por ser herramientas conocidas y cada una con propiedades diferentes (véase Figura 2.1). Cabe destacar que todas estas tecnologías presentan características adecuadas para cumplir con los requisitos del proyecto, ya sea a través del uso de bibliotecas y paquetes específicos, o mediante sus capacidades nativas, como en el caso de Django, Plotly Dash y Shiny, desarrolladas en Python y R.

Durante el proceso de investigación, se ha constatado que todas las tecnologías analizadas cuentan con una sólida base y una amplia comunidad de desarrolladores. React, conocido por su capacidad para construir interfaces de usuario interactivas, ofrece una gran flexibilidad y un rendimiento eficiente. Django, al ser un framework de desarrollo web en Python, proporciona una estructura sólida y segura para la creación de aplicaciones robustas y escalables. Angular, por su parte, se destaca por su arquitectura modular y su enfoque en la construcción de aplicaciones de una sola página. Plotly Dash y Shiny R, al ser nativos de Python y R, respectivamente, ofrecen una integración directa con los lenguajes de programación utilizados en la Ciencia de Datos, lo que resulta altamente ventajoso para la manipulación y Visualización de Datos complejos.

Es importante tener en cuenta que la tecnología avanza rápidamente y lo que pueda ser adecuado en el presente, podría no serlo en un futuro no muy lejano. En este sentido, tanto Django como Shiny R se perfilan como herramientas que podrían resistir mejor los cambios y adaptarse a las evoluciones tecnológicas, debido a su arraigo en los lenguajes Python y R, que continúan siendo ampliamente utilizados en el ámbito de la Ciencia de Datos.

Dadas las limitaciones temporales del proyecto, se toma la decisión de utilizar ambas herramientas para el desarrollo del proyecto. Esta elección se basa en el hecho de que Shiny R es más fácil de aprender y utilizar en comparación con un framework completo como

Django. La curva de aprendizaje de Shiny R es más accesible, lo que permite una adopción más rápida y una mayor eficiencia en el desarrollo del proyecto en el tiempo asignado, proporcionando una ruta más directa hacia la implementación exitosa del proyecto, sin comprometer los objetivos y funcionalidad requerida.

De esta manera, Django en calidad de framework aporta la solidez y estructura que se espera en el desarrollo de un proyecto de una aplicación web, incluso permite solventar con eficacia ciertos requisitos de la aplicación, como el acceso identificado, ya que Shiny R presenta en este apartado cierta carencia. Por tanto, se delega en esta última tecnología íntegramente la parte de Análisis y Visualización de Datos ya que R es un lenguaje válido y potente para este propósito.

En tercer lugar, por los motivos planteados anteriormente, se puede afirmar la viabilidad del proyecto, logrando los objetivos que se habían planteado al inicio del trabajo.

Una vez realizado el proyecto, se identifica una particularidad en la tecnología Shiny R que merece ser comentada. Una de las pequeñas desventajas es la amplia cantidad de paquetes y bibliotecas existentes, algunos de los cuales son desarrollados por terceros. Esto puede plantear cierta incertidumbre en cuanto a su mantenimiento y compatibilidad a lo largo del tiempo. Sin embargo, es importante subrayar que siempre existe una alternativa viable y eficiente, aunque esta no sea tan directa.

Esta particularidad se ha identificado a la hora de implementar el sistema de inicio de sesión y al intentar mejorar la accesibilidad y usabilidad de la aplicación.

No obstante, la comunidad activa y colaborativa que rodea a Shiny es una fortaleza incuestionable. Ésta se esfuerza continuamente por mantener actualizadas las bibliotecas y paquetes, lo que asegura que las posibles limitaciones causadas por la obsolescencia sean abordadas de manera efectiva. Además, la amplia disponibilidad de recursos y documentación en línea aporta a los desarrolladores una guía sólida para enfrentar cualquier desafío.

La característica de Shiny R mencionada no representa un obstáculo insuperable, sino más bien un estímulo para la creatividad y el aprendizaje. La solidez y flexibilidad de ambas tecnologías permiten abordar cualquier dificultad con éxito, y la combinación de ambas se traduce en una aplicación web de Análisis y Visualización de Datos altamente eficiente y con un potencial ampliamente prometedor.

Tabla 5.1: Resumen de los objetivos logrados.

Objetivos	Logrado
Facilitar el uso de la aplicación	✓
Proteger la aplicación	✓
Disponer de un sistema de ayudas	✓
Obtener resultados visuales	✓
Interactuar con la aplicación	✓
Investigar las tecnologías disponibles	✓
Comprobar la viabilidad	✓
Poner en prácticas conocimientos	✓
Aprender nuevas tecnologías	✓
Adquirir experiencia	✓
Crear un prototipo de aplicación explotable	✓

5.2.- POSIBLES DESARROLLOS FUTUROS

Los posibles trabajos futuros para mejorar y enriquecer la aplicación adaptándola a los usuarios, asegurando que continúe siendo una herramienta eficiente y amigable para el Análisis y Visualización de Datos, son los siguientes:

- **Ampliar funcionalidades**: contemplar la posibilidad de añadir nuevas funcionalidades para aumentar la utilidad. Un enfoque importante sería implementar el preprocesamiento de datos, permitiendo una manipulación avanzada de la información antes de su análisis y visualización. Esta mejora contribuiría a optimizar la calidad y eficiencia del Análisis de Datos en la aplicación.

- **Desplegar en un entorno web real**: llevar a cabo el despliegue de la aplicación en un ámbito web real. Al hacerlo, se garantiza que los usuarios puedan acceder a la aplicación desde diversos dispositivos y ubicaciones, mejorando su accesibilidad y facilitando su uso en diferentes entornos.

- **Mejorar y ampliar características para los roles de los usuarios**: ofrecer una experiencia más personalizada, añadiendo características específicas para cada uno de los roles de usuarios (*System Operator*, *Manager*, *Teacher* y *Student*). De esta manera, se adaptará la aplicación a las necesidades particulares de cada grupo de usuarios, optimizando su interacción con la plataforma.

- **Pulir la accesibilidad y usabilidad**: seguir trabajando en mejorar la accesibilidad y usabilidad de la aplicación, a pesar de haberse demostrado una buena implementación en su mayoría. Este enfoque permitirá perfeccionar la experiencia de todos los usuarios, independientemente de sus capacidades, y asegurar una experiencia fluida y amigable en el uso de la aplicación.

- **Migrar paulatinamente a Django**: considerar migrar gradualmente todas funcionalidades que actualmente se encuentran distribuidas entre Shiny y Django hacia Django en su totalidad. Esta migración progresiva busca consolidar la aplicación en un solo entorno tecnológico, lo que podría facilitar su mantenimiento y escalabilidad a largo plazo.

Bibliografía

[2.1] Design Framework for the Development Dynamic Web Applications
Gustavo Martínez Villalobos, Germán Darío Camacho Sánchez, Daniel Alberto Biancha Gutiérrez
DISEÑO DE FRAMEWORK WEB PARA EL DESARROLLO DINÁMICO DE APLICACIONES (redalyc.org)
Scientia Et Technica (2010)
1 de marzo de 2023

[2.2] Modelo de procesos para el desarrollo del front-end de aplicaciones web
José Jesús Valdivia Caballero
Vista de Modelo de procesos para el desarrollo del front-end de aplicaciones web (ulima.edu.pe)
11 de marzo de 2023

[2.3] React
React – Una biblioteca de JavaScript para construir interfaces de usuario (reactjs.org)
4 de marzo de 2023

[2.4] ReactJS by Example - Building Modern Web Applications with React
Vipul A. M., Prathamesh Sonpatki
Packt Publishing Ltd. (2016)
4 de marzo de 2023

[2.5] Integrating React.js and Shiny
Integrating React.js and Shiny - RStudio
Alan Dipert
4 de marzo de 2023

[2.6] Análisis comparativo de tecnologías front end Angular.js vs React.js, en el modelo de procesos para el desarrollo de aplicaciones web
Luis Ali Anchundia Medrano
Babahoyo (2022)
9 de marzo de 2023

[2.7] Introducción a Django
Introducción a Django - Aprende sobre desarrollo web | MDN (mozilla.org)
6 de marzo de 2023

[2.8] Django Web Framework (Python)
Django Web Framework (Python) - Learn web development | MDN (mozilla.org)
7 de marzo de 2023

[2.9] Templates
Templates | Django documentation | Django (djangoproject.com)
7 de marzo de 2023

[2.10] Views
Writing views | Django documentation | Django (djangoproject.com)
12 de marzo de 2023

[2.11] Allow django to create R scripts and generate charts and other data output from those scripts
django-rpy2 · PyPI
6 de marzo de 2023

[2.12] Estudio comparativo entre los recursos Laravel 9 y Django para el desarrollo de aplicaciones web
Jorge Javier Villamar Gastesi
Babahoyo (2022)
9 de marzo de 2023

[2.13] ¿Qué es Angular? Características y ventajas
https://blog.hubspot.es/website/que-es-angular
7 de abril de 2023

[2.14] SPA (Single-page application)
SPA (Single-page application) - MDN Web Docs Glossary: Definitions of Web-related terms | MDN (mozilla.org)
7 de abril de 2023

[2.15] Introducción a aplicaciones web progresivas
Introducción a aplicaciones web progresivas - Aplicaciones Web Progresivas | MDN (mozilla.org)
7 de abril de 2023

[2.16] Template syntax
Angular - Template syntax
7 de abril de 2023

[2.17] Understanding dependency injection
Angular - Understanding dependency injection
7 de abril de 2023

[2.18] CLI Overview and Command Reference
Angular - CLI Overview and Command Reference
7 de abril de 2023

[2.19] Angular components overview
Angular - Angular components overview
7 de abril de 2023

[2.20] Component
Angular - Component
7 de abril de 2023

[2.21] Directive
Angular - Directive
7 de abril de 2023

[2.22] Work with Python and R modules
Work with Python and R modules - Azure Databricks | Microsoft Learn
7 de abril de 2023

[2.23] AngularJS in Action
Lukas Ruebbelke
Manning (2015)
7 de abril de 2023

[2.24] Explore Angular Resources
Angular - EXPLORE ANGULAR RESOURCES
7 de abril de 2023

[2.25] Dash Python User Guide
Dash Documentation & User Guide | Plotly
12 de abril de 2023

[2.26] Plotly Open Source Graphing Library for Python
Plotly Python Graphing Library
12 de abril de 2023

[2.27] Flask
Welcome to Flask — Flask Documentation (2.1.x) (palletsprojects.com)
12 de abril de 2023

[2.28] Dash R User Guide
Dash Documentation & User Guide | Plotly
12 de abril de 2023

[2.29] Dash Layout
Part 1. Layout | Dash for Python Documentation | Plotly
12 de abril de 2023

[2.30] Basic Dash Callbacks
Part 2. Basic Callbacks | Dash for Python Documentation | Plotly
12 de abril de 2023

[2.31] Dash Core Components
Dash Core Components | Dash for Python Documentation | Plotly
12 de abril de 2023

[2.32] Comunidad de Plotly
Plotly Community Forum - The world's largest online community for data science apps and data visualization.
12 de abril de 2023

[2.33] Shiny
Shiny (rstudio.com)
19 de abril de 2023

[2.34] R Interface to Python
Interface to Python • reticulate (rstudio.github.io)
19 de abril de 2023

[2.35] Sending data from client to server and back using shiny
Shiny - Getting help (rstudio.com)
19 de abril de 2023

[3.1] Python 3 al descubierto
Arturo Fernández Montoro
Alfaomega Grupo Editor, S.A. de C.V. (2013)
30 de marzo de 2023

[3.2] Models
Models | Django documentation | Django (djangoproject.com)
24 de abril de 2023

[3.3] Popular Python libraries and their applications domains
POPULAR PYTHON LIBRARIES AND THEIR APPLICATION DOMAINS (researchgate.net)
24 de abril de 2023

[3.4] Django Rest
Home - Django REST framework (django-rest-framework.org)
24 de abril de 2023

[3.5] FastAPI
FastAPI (tiangolo.com)
24 de abril de 2023

[3.6] What is R?
https://www.r-project.org/about.html
19 de junio de 2023

[3.7] Contributors
https://www.r-project.org/contributors.html
19 de junio de 2023

[3.8] Available CRAN Packages By Name
https://cran.r-project.org/web/packages/available_packages_by_name.html
22 de junio de 2023

[4.1] Ingeniería del software
Ian Sommerville
Pearson Educación, S.A., Madrid (2005)
4 de mayo de 2023

[4.2] Usable Usability: Simple Steps for Making Stuff Better
Eric Reiss
John Wiley & Sons, Inc (2012)
27 de julio de 2023

[4.3] Learning Responsive Web Design: A Beginner's Guide
Clarissa Peterson
O'Reilly Media, Inc (2014)
27 de julio de 2023