UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Diseño y desarrollo de una aplicación web configurable para el análisis de manifiestos electorales y campañas online.

Edwin Alejandro Maruri Tinajero

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito

para la obtención del título de

INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

Quito, 12 de diciembre de 2020

Universidad San Francisco de Quito USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN

DE TRABAJO DE FIN DE CARRERA

Diseño y desarrollo de una aplicación web configurable para el análisis de manifiestos electorales y campañas online

Edwin Alejandro Maruri Tinajero

Nombre del profesor, Título académico **Daniel Riofrío, Ph.D.**

Quito, 12 de diciembre de 2020

© DERECHOS DE AUTOR

Nombres y apellidos: Edwin Alejandro Maruri Tinajero

Código: 00136515

Cédula de identidad: 1750087007

Lugar y fecha: Quito, 12 de diciembre de 2020

ACLARACIÓN PARA PUBLICACIÓN

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

UNPUBLISHED DOCUMENT

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

RESUMEN

Actualmente, los sistemas democráticos dependen de la capacidad de los estados para promover la participación de partidos y movimientos políticos en función de propuestas de trabajo y no de ofertas populistas que luego de elecciones no llegan a cumplirse. Es así, como en la era de las redes sociales y su incursión en el terreno de la política, es necesario interpretar el lenguaje utilizado por los candidatos en la red social, durante una campaña política. Por lo que se plantea el análisis del contenido de la campaña digital realizada por los candidatos durante una elección. Es por esto que Riofrío et al. proponen un prototipo de aplicación para cumplir con este cometido. Así, cimentada la base del prototipo, surgen nuevas necesidades en búsqueda de aumentar su alcance y flexibilidad para la recolección de datos y ejecución de procesos de análisis. Dados estos antecedentes, se plantea como solución la construcción de una aplicación web que permita satisfacer estas necesidades. Se utiliza un proceso de diseño orientado a objetos en dónde se definen los casos de uso, diagramas de clases, diseño de base de datos y el diseño de la interfaz gráfica. Fundamentado en el diseño, se procede con el desarrollo de la aplicación, utilizando tecnologías como Django, Visual Studio Code, HTML, CSS y PostgreSQL. La aplicación desarrollada permite la creación de campañas, candidatos, asociar cuentas de redes sociales y manifiestos; además de la recolección de posts que permite el posterior análisis y muestra del resultado obtenido. Se espera que esta aplicación sirva de base para estudiar a mayor profundidad las campañas políticas y el impacto que tienen las redes sociales en éstas.

Palabras clave: análisis, manifiesto, métricas, aplicación web, diseño, desarrollo.

ABSTRACT

Currently, democracies depend on effective campaigns in which they can present their work proposals. Besides, social networks have been taking ground for this process. Due to this, questions are raised about the analysis of manifestos (government plans) and online campaigns carried out by candidates for elections. This is why Riofrío et al. make an application prototype as a proposal to fulfill this task. Thus, once the base of the prototype has been established, new needs arise in search of increasing its scope and flexibility for data collection and execution of the analysis process. Given these antecedents, the realization of a web application that allows meeting these needs is proposed as a solution. To comply with this, it is proposed to carry out both the design and the development of the application, to make it more scalable. Being a web application, the design should focus on both the Front-End and the Back-End. Thus, the design process is based on the type of object-oriented design (DOO), which allows the problem to be solved to be better modeled. The design consists of the definition of use cases, class identification, database design and UI design. Based on the design, we proceed with the development of the application, using technologies such as Django, Visual Studio Code, HTML, CSS and PostgreSQL. In the end, it was possible to create an application that allows the creation of campaigns, candidates, accounts and manifestos; in addition to the collection of posts that allows the subsequent analysis and display of the result obtained. This application is expected to serve as the basis for future work on campaign analysis and the impact of social media.

Keywords: analysis, manifest, metrics, web application, design, development.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Antecedentes

Los políticos de las democracias modernas dependen de realizar una campaña que comunique, lo mejor posible, el mensaje de su propuesta política, promoviendo y detallando sus manifiestos electorales (planes de gobierno). El proceso de difusión o divulgación de estos ha evolucionado en conjunto con las herramientas y medios utilizados para este fin. Por lo que, el Internet, y las redes sociales, se han convertido en un medio importante para que los políticos se comuniquen e interactúen con la comunidad. Sin embargo, se ha cuestionado el rol limitado de la tecnología sobre lo político; afirmando que los ciudadanos deberían poder ser capaces de participar más activa y directamente en las decisiones políticas.

Debido a esto, ha sido primordial el desarrollo de nuevos enfoques de análisis de datos que permitan comprender el contexto político, la toma de decisiones y el rol de la tecnología en este entorno. De esta manera, de acuerdo con Riofrío et al., surge un prototipo de aplicación, que consiste en una herramienta para el estudio y entendimiento del uso de las redes sociales para la compartición de mensajes alineados, o no, a los planes de gobierno de los candidatos de una elección. El prototipo se basa en la recopilación de datos provenientes de los "timelines" de Twitter de los candidatos y su correspondiente manifiesto electoral, durante la campaña electoral; siendo ambas fuentes sometidas a una prueba de correlación utilizando métricas de minería de texto (2019).

Descripción

Dado los antecedentes, y considerando el potencial que exhibe el prototipo de aplicación mencionado, se ha pretendido otorgar y ofrecer una mayor escalabilidad, funcionalidad y facilidad de uso y mantenimiento de este, a través de un sistema de más alto nivel, que

permita flexibilidad y ciertas configuraciones, sobre todo, en la recopilación de datos necesarios para el sistema. Así, para la elaboración de este proyecto, se plantea el desarrollo de una aplicación basada en el prototipo. Los nuevos requerimientos, necesidades y/o exigencias se han originado de uno de los autores del prototipo (principal interesado/cliente). Por lo que, para lograr este cometido, se propone el diseño y el desarrollo de una *aplicación web full-stack* como solución.

Para poder desarrollar un sistema informático, se requiere de un proceso iterativo, basado en un diseño (también iterativo). De este modo, el diseño de la aplicación consta de dos partes esenciales: diseño funcional/lógico y diseño visual. En cuanto al diseño funcional, se realiza un diseño basado en el paradigma de orientación a objetos y con uso del patrón de diseño MVC para obtener una arquitectura de software que permita mantener una separación clara de los datos, la interfaz gráfica y la lógica del proyecto (módulos independientes), que consta de la definición de los casos de uso para identificación de elementos clave, diagramas necesarios para la comprensión del sistema, mediante el uso de UML y diseño robusto de la base de datos relacional. Por otro lado, el diseño visual, basado en UI (Interfaz de Usuario), constituye la creación de wireframes y mockups que permiten la visualización de las pantallas (vistas) necesarias para la presentación de los datos, información, opciones, configuraciones, etc. Todo este diseño, brinda una idea más clara del problema antes de comenzar con la programación, facilitando, hasta cierto punto, este paso.

En cuanto al desarrollo de la aplicación, se marca como guía el prototipo visto. Este desarrollo constituye dos secciones primordiales: Back-End y Front-End. Por el lado del Back-End de la aplicación, se provee de diferentes servicios, esencialmente en el proceso de recopilación de datos, al Front-End. Además, existen otras funcionalidades extra que aporten al estudio del contexto político. Por su lado, el Front-End consta de una secuencia de páginas

(vistas) lo suficientemente flexibles que permiten la petición, parametrización, configuración y recepción de lo requerido al y por el sistema.

Objetivos

- **Objetivo General:**

  Diseñar, desarrollar e implementar una aplicación web configurable, flexible y funcional que trabaje sobre el prototipo de aplicación de análisis de manifiestos electorales y campañas online de candidatos, para facilitar y potenciar su uso.

- **Objetivos Específicos**
  - Ofrecer una herramienta libre en internet, que sea mayormente flexible para el análisis y estudio de datos durante campañas electorales.
  - Visualizar el impacto de las redes sociales y el panorama político en general para la toma de decisiones.
  - Permitir la configuración de los procesos de recopilación de datos del prototipo existente.

Requisitos

a. Funcionales

- Disponer de un **colector configurable** de “Timelines” y Manifiestos.
- Permitir la creación, lectura, actualización y borrado de las **entidades** necesarias para el sistema (uso de base de datos).
- Pantallas o Vistas web, con cierto nivel de UI, que permitan la configuración manual de la aplicación y ofrezca distintos servicios relacionados a los usuarios.
- Asociación de timelines y manifiestos electorales con los respectivos candidatos.

b. Técnicos

- Conocimiento de desarrollo web **Full-Stack** (Front-End y Back-End).
- Uso del **marco de trabajo** Django de Python para el desarrollo.
- Manejo de la arquitectura MVC.
- Diseño visual de la aplicación.
- Diseño y normalización de la Base de Datos a utilizar.
- Diseño de la lógica de la programación.
- Integración entre Back-End y Front-End desarrollado.
- Creación de la documentación del sistema.

Beneficios esperados

- Recopilación, mayormente flexible, de datos para el análisis de manifiestos electorales y campañas online.
- Ayuda a la comunidad votante y analistas políticos sobre el estudio, la interpretación y el análisis del contexto político y el uso de las redes sociales para la realización de campañas electorales.

Definiciones y terminología

- **Plan de gobierno/Manifiesto electoral:** documento que contiene la propuesta política, acciones, estrategias y metas que una organización política, que pretende participar activamente en un proceso de elecciones, elabora como plan de gobierno.
- **Timeline:** en redes sociales, conjunto de posts cronológicos asociados a la cuenta de una red social determinada.

- **Sistema:** estructura, unidad o totalidad compuesta de diversos componentes o partes que dependen entre sí y trabajan en conjunto para lograr un objetivo.
- **Aplicación web:** herramienta alojada en un servidor web que puede ser accedida a través de la red y visualizada mediante un navegador.
- **Desarrollo:** en informática, proceso de modelado de las ideas humanas en los computadores mediante programación.
- **Desarrollo Full-Stack:** proceso que gestiona los pasos y componentes lógicos y visuales relacionados con la creación y el mantenimiento de una aplicación web.
- **Framework:** o marco de trabajo, es un conjunto de prácticas estándares, herramientas y librerías para la resolución de problemas similares.
- **Diseño:** en informática, traza o bosquejo de un proyecto basado en su concepción original orientado a la producción de este.
- **UI:** del inglés *User Interface*, es la vista mostrada a un usuario la cual le permite interactuar con un sistema.
- **Análisis y diseño orientado a objetos (A/DOO):** proceso para interpretar y modelar software en términos de la *descripción* (características y comportamientos) de los objetos de la vida real, sus interrelaciones (herencia, composición, etc.) y la manera en que se comunican. Hasta cierto punto, este proceder se acerca a un modo más natural de pensar. Básicamente, con el DOO se ***encapsula*** (empaqueta) los atributos y comportamientos de un objeto en sí mismo "ocultando" su implementación (en una clase), permitiendo la abstracción de dicha implementación. Utiliza el lenguaje gráfico UML para la representación del sistema.
- **Abstracción:** en términos generales de informática, es la simplificación sistemática de un problema complejo a través del modelado de sus partes (puntos de interés) y

consiste en aislar un elemento de su contexto y hacer énfasis en el **qué hace** y no en el **cómo lo hace**.

- **Clase:** modo de empaquetar, a manera de plano, la definición de un objeto de la vida real. Permite la creación (instanciación) de varios objetos, permitiendo reutilización.
- **Lenguaje Unificado de Modelado (UML):** lenguaje gráfico que permite utilizar una notación estándar para la representación de software orientado a objetos en distintas fases, etapas o contextos del mismo.
- **Caso de Uso:** especificación que permite describir los distintos escenarios, en términos de actores y acciones, que un sistema puede presentar. Un caso de uso representa una funcionalidad o capacidad que el sistema provee.
- **Diagrama de casos de uso:** gráfico que modela las interacciones entre los actores relacionados a un sistema y el sistema en sí, sin lujo de detalle.
- **Diagrama de clases:** gráfico que modela bloques de construcción de software reutilizables usados en un sistema; permite especificar las relaciones estructurales de las partes del sistema.
- **Arquitectura de la información:** proceso de análisis, selección y estructuración de los datos e información de un sistema en términos de las interfaces gráficas.
- **Wireframe:** esquema o plantilla, a modo de boceto, que permite tener una guía visual de una página o pantalla que se quiere mostrar a un usuario.
- **Sitemap:** o árbol de navegación, es una manera de representar visualmente la estructura de navegación en un sitio web.
- **Mockup:** conjunto de fotomontajes que muestran la estructura visual final de un diseño. Permite visualizar el resultado final antes de programarlo.
- **Base de Datos:** colección estructurada (o no) de datos interrelacionados o pertenecientes a un mismo contexto.

- **Modelo Relacional:** en bases de datos, cúmulo de tablas conformada por un nombre identificador, atributos con sus respectivos dominios y tuplas (filas o conjunto de relaciones de valores).
- **Normalización:** en base de datos, proceso basado en la descomposición o subdivisión iterativa (sin pérdida) que ayuda a garantizar que la información almacenada se encuentre sin redundancias innecesarias, se conserve las dependencias y que, además, sea fácilmente recuperable.
- **Diagrama de Entidad-Relación** (**DER):** gráfico de flujo que permite observar cómo las entidades de una base de datos relacional se relacionan entre sí.
- **Servidor Web:** programa que procesa aplicaciones en el lado del servidor, establece las conexiones con los clientes y da una respuesta oportuna.

DESARROLLO DEL TEMA

Materiales

Con la intención de crear una aplicación web funcional y escalable, se utilizó herramientas que faciliten varios de los procesos relacionados con el diseño y el desarrollo de la misma. Por lo tanto, esta sección se encarga de describir tanto las herramientas como las tecnologías y la arquitectura base utilizada para la realización de la aplicación web.

Herramientas de diseño.

Creately.

Interfaz web que permite el modelado interactivo de diagramas UML. En el presente proyecto se utilizó para la creación de los diagramas de clases, diagrama de casos de uso, y diagrama de entidad-relación (DER).

Figma.

Aplicación web que permite el modelado y diseño de interfaces. La herramienta se utilizó para la creación de wireframes y mockups del proyecto.

WriteMaps.

Herramienta web para la diagramación de sitemaps o árboles de navegación.

Tecnologías de desarrollo.

PostgreSQL.

Herramienta de código abierto, utilizada como Sistema Gestor de Base de Datos (SGBD). Permite la gestión de bases de datos relacionales orientado a objetos.

Django.

Es un framework o marco de trabajo web de alto nivel, gratuito y de código abierto escrito en Python. Permite el desarrollo de sitios web de manera sencilla y rápida, simplificando tareas repetitivas o comunes. Este framework se basa en el patrón MVC. El módulo Modelo gestiona los datos de la aplicación o sitio, la Vista es el módulo que muestra la información al usuario y le permite interacción y el Controlador es el módulo de gestionar las comunicaciones existentes entre la Vista y el Modelo.

Visual Studio Code.

Editor de texto de código abierto utilizado para el proceso de desarrollo: programación del código fuente, estructuración de directorios del proyecto, depuración y pruebas del proyecto.

Arquitectura base.

A continuación, se presenta la Figura 1, que exhibe la arquitectura utilizada para la creación del prototipo de aplicación de análisis de manifiestos y campañas online (timelines de Twitter). Dicha arquitectura, funciona como base para el diseño y desarrollo del proyecto en cuestión.

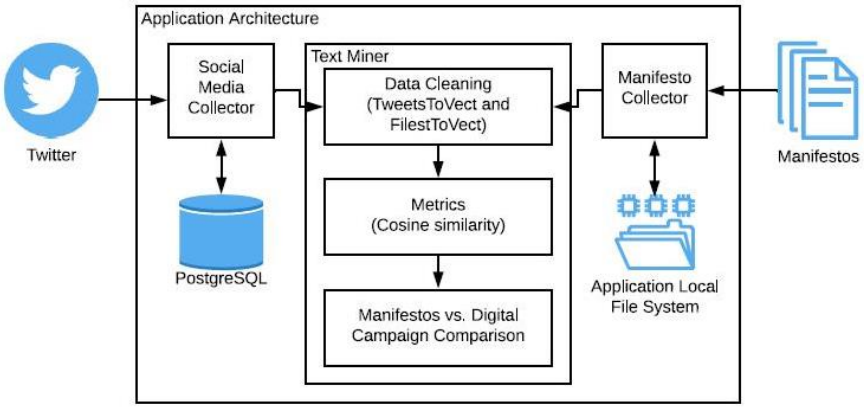

Figura 1. Arquitectura del prototipo de aplicación.

Metodología

Dado que la concepción de un proyecto puede evolucionar y cambiar con el tiempo, se incluyen etapas esenciales para el diseño y desarrollo de este: análisis, diseño, implementación, prueba y depuración, despliegue y mantenimiento. Todo este proceso tiene un fuerte enfoque en el paradigma de orientación a objetos.

Proceso iterativo.

- Recopilar los requerimientos para el desarrollo del proyecto.
- Analizar detalladamente los requerimientos, para comprender el **qué** tiene que hacer el sistema (especificación del diseño).
- Diseñar las soluciones a los distintos requerimientos recopilados, para identificar el **cómo** hacer lo que se supone que debe hacer el sistema. Este es un paso esencial para construcción y codificación de un sistema.
- Implementar y programar el sistema, basado en el diseño del mismo.
- Probar y depurar el sistema implementado y diseñado.
- Corregir los pasos previos, en caso de ser necesario.
- Iterar nuevamente, si hace falta.

Descripción general.

El desarrollo del proyecto pretende ser la solución al planteamiento de una necesidad: sistema de análisis de correlación entre manifiestos electorales (planes de gobierno) y campañas online, realizadas en redes sociales. Por lo que, se planea diseñar y desarrollar software (aplicación web). Se pretende realizar un diseño sustancial y una

implementación completa del sistema. Para cumplir con esta idea, se realiza un análisis y diseño orientado a objetos (A/DOO), diseño UI y diseño de bases de datos.

Especificación de hardware objetivo.

La interfaz de usuario contiene los siguientes componentes de hardware:

- Pantalla para visualizar la información correspondiente, asociada al análisis de manifiestos y campañas online.
- Teclado para suministrar datos alfanuméricos. Requerido para el ingreso de datos y llenado de campos necesarios para el sistema.
- Conectividad física a Internet, para la obtención de datos e información mediante la web.

Descripción de requerimientos generales.

Un grupo de investigación y desarrollo construyó un prototipo de aplicación, para el análisis de manifiestos electorales y campañas online que permita a interesados (usuarios) comprender la correlación entre ambas partes. El grupo busca crear una herramienta web y aumentar el alcance del prototipo. Los tipos de usuarios que pueden interactuar con el sistema pueden ser *lectores* (visualizadores) o *configuradores* del sistema. El usuario lector puede visualizar la relación de los posts (en este caso tweets) con el timeline del candidato y las menciones o reacciones en respuesta al *screen name* de este, un contador de frecuencia de texto con el número de veces que una palabra se repite en un documento y/o un post (tweet) y la similitud de los manifiestos y un timeline (conjunto de posts asociados a una cuenta de una red social) determinado (este se

considera un requerimiento más primordial). Por otro lado, el usuario configurador (que también es lector) puede configurar la alimentación de los colectores que requiere el sistema (timelines y documentos), crear entidades, como *campañas, candidatos*, *cuenta/red social* y/o *manifiestos* usados por la base de datos del sistema y para el posterior análisis y asociar manifiestos (documentos) con su respectivo candidato.

Con este precedente, se busca desarrollar una primera versión de la aplicación web que cumpla con las funcionalidades mencionadas y que se ejecute en computadoras personales (posteriormente, en dispositivos móviles).

Sesión con la aplicación.

Una **sesión** con la aplicación depende del tipo de usuario, permitiendo mostrar y proceder con las funcionalidades pertinentes a cada uno. Así, un usuario al acceder a la pantalla debe experimentar la siguiente secuencia de eventos:

- **i.** La pantalla "da" la bienvenida al sistema y muestra la página que contiene a las distintas opciones y funcionalidades.
- **ii.** Para un usuario del tipo configurador, se solicita autenticación para acceder a sus funcionalidades extra. Mientras que, para un usuario tipo lector no hace falta autenticación, pero se mostrarán las funciones limitadas para ese tipo de usuario.
- **iii.** Una vez autenticado, se mostrarán en pantalla las opciones acordes al tipo de usuario de una manera ordenada y con sentido.
    - **1.** Opciones que usuario LECTOR dispone (no hay necesidad de autenticación):
        - **a.** Visualizar resultados de análisis de una campaña determinada (caso de uso). Los resultados pueden contener lo siguiente:
            - **i.** Similitud de los manifiestos y timeline.
            - **ii.** Posts directamente relacionados con el timeline de un candidato.

iii. Contador de frecuencia de texto.

b. Solicitud de registro o promoción a usuario configurador.

2. Opciones que usuario CONFIGURADOR:

a. Posee las mismas opciones que un usuario tipo lector.

b. Configurar una campaña (caso de estudio). Consiste en:

i. Ver, crear, actualizar y eliminar entidades: campaña, candidato, cuenta/red social, manifiesto para la base de datos del sistema.

ii. Proveer, enlazar y/o desligar manifiestos de un candidato.

c. Configurar, inicializar y/o pausar el proceso de recopilación de timelines.

d. Procesar una sesión de análisis de una campaña existente dada una métrica (en combobox).

e. Salir/Cerrar sesión.

f. Eliminar/Configurar cuenta.

iv. El usuario podrá elegir entre las opciones disponibles y obtendrá una respuesta correspondiente. Luego, el usuario podrá regresar y elegir otra opción disponible.

v. En general, si se realiza una acción de manera exitosa, los datos e información solicitada serán desplegados en pantalla acompañados, a veces, de texto complementario. Si llegase a ocurrir algún error, se mostrará un mensaje respectivo.

Por motivos de legibilidad y facilidad, se denominará simplemente *usuario* a un usuario lector y *super-usuario* a un usuario configurador a partir de ahora.

Procedimiento

Proceso de diseño.

El diseño de la aplicación consta de dos partes fundamentales: diseño funcional/lógico y diseño visual (UI). Ambos tienen sustento en el análisis de requerimientos de la aplicación, suministradas por el cliente. Teniendo así, una estrecha relación entre ambas partes. En cuanto al diseño funcional, se aspira realizar un *diseño orientado a objetos* (DOO) basado en el análisis del mismo paradigma para el modelado robusto de la base de datos a utilizar, los servicios que se proveerá al Front-End y los elementos lógicos esenciales para la solución. Por otro lado, el diseño visual comprende la *arquitectura de la información* y la creación de *wireframes* y *mockups* que permitan componer y organizar la secuencia de páginas (vistas), lo suficientemente flexibles e interactivas para el uso de los servicios ofrecidos por el Back-End.

Definición de casos de uso.

En general, los actores, al realizar una respectiva acción o solicitud, esperan por una respuesta apropiada por parte del sistema.

i. Autenticación (sign in):

Curso normal: El sistema recibe al **usuario** con una pantalla de bienvenida, le muestra un formulario y le solicita que se autentique (que se identifique). Para poder ser autenticado, el usuario debe estar registrado como **super-usuario** y debe introducir sus credenciales (nombre de usuario y pin/password), mediante teclado. El usuario introduce correctamente sus credenciales (coinciden en la base de datos) y, posteriormente, se despliega la pantalla de inicio con las

funcionalidades respectivas, además de las disponibles como **usuario**: configurar una campaña, configurar e inicializar proceso de recopilación de timelines, procesar una sesión de análisis de una campaña existente dada una métrica, salir/cerrar sesión y eliminar cuenta.

Curso alternativo: *Campo(s) Erróneo(s)*. (revisar sección "Cursos Alternativos en Común").

Curso alternativo: *Demitir*. (revisar sección "Cursos Alternativos en Común").

Curso alternativo: *Error interno*. (revisar sección "Cursos Alternativos en Común").

ii. Registro-Promoción (sign up):

Curso normal: Si un **usuario** desea ser considerado como **super-usuario**, debe ser registrado en la base de datos (para mantener control de sus acciones). Así, el usuario elige la opción "registro/promoción" que le permita llenar un formulario con información pertinente, como nombre, correo, número de identificación, etc., creando una solicitud para su promoción. El usuario introduce correctamente todos los campos necesarios y será redirigido a la pantalla de inicio. Se le notificará del éxito de la solicitud presentada. El usuario esperará por una respuesta vía correo electrónico.

Curso alternativo: *Campo(s) Erróneo(s)*.

Curso alternativo: *Demitir*.

Curso alternativo: *Error interno*.

iii. Configurar una campaña

Curso normal: Un **super-usuario** indica al sistema que desea configurar una campaña. El sistema le permite realizar varias opciones, como ver, crear, actualizar y eliminar entidades: campaña, candidato, cuenta/red social, manifiesto para la base de datos del sistema y colocar las relaciones necesarias entre ellas. Cada entidad posee campos distintos (acorde al diseño de la base de datos), por lo que se presentarán los formularios adecuados. El usuario ingresa correctamente los campos requeridos y "acepta" los cambios (si hubiesen); se solicita confirmación por parte del **super-usuario**. La respuesta es positiva y el sistema responde comunicando que la operación ha sido exitosa.

Curso alternativo: *Demitir*

Curso alternativo: *Intento de Demitir.*

Curso alternativo: *Campo(s) Erróneo(s).*

Curso alternativo: *Error Interno.*

iv. Crear entidad

Curso normal: Un **super-usuario** puede realizar la creación de una entidad (campaña, candidato, red social/cuenta o manifiesto). De acuerdo con la entidad seleccionada, el sistema mostrará el formulario pertinente, que permita crear dicha entidad y almacenarla en la base de datos (depende del diseño de la base de datos). El usuario introduce correctamente los campos y atributos requeridos en el formulario y el sistema responde comunicando que la operación ha sido exitosa.

Curso alternativo: *Demitir*.

Curso alternativo: *Intento de Demitir*.

Curso alternativo: *Campo(s) Erróneo(s)*.

Curso alternativo: *Error Interno*.

v. Configurar recolección de datos

Curso normal: El **super-usuario** solicita al sistema "configurar la recolección de timelines". Así, el usuario indicará los factores asociados a la recolección de datos de un caso de estudio, como el periodo, el horario y otros parámetros asociados a este proceso. El sistema solicita confirmación. Si a respuesta es positiva, se procede a configurar y programar la tarea para luego ser ejecutada. Si todo ha sido completado correctamente, se puede cumplir con la petición e indicar que ha sido exitoso, actualizando la página, mostrando un mensaje y procediendo con la ejecución de la recolección de datos (si fuese el momento).

Curso alternativo: *Demitir.*

Curso alternativo: *Intento de Demitir.*

Curso alternativo: Error Interno.

vi. Comenzar/Pausar recolección de timelines

Curso normal: Un **super-usuario** podrá comenzar y pausar arbitrariamente un proceso de recolección programado. El Sistema solicita confirmación. Si la respuesta es positiva, el sistema procede con la petición e indica que ha sido satisfactorio el proceso solicitado.

Curso alternativo: *Demitir.*

Curso alternativo: *Intento de Demitir.*

Curso alternativo: *Error Interno*.

vii. Proveer manifiesto

Curso normal: El **super-usuario** solicita "suministrar" manifiesto mediante la opción correspondiente. El sistema permitirá la carga de un documento del tipo PDF. Dicho documento debe ser asociado a un candidato preexistente en la base de datos (de un caso de estudio en específico). El sistema asiste al requerimiento y, si ha sido exitoso, muestra un mensaje de "éxito".

Curso alternativo: Demitir.

Curso alternativo: *Intento de Demitir.*

Curso alternativo: *Error Interno*.

viii. Desligar manifiesto

Curso normal: El **super-usuario** elige desligar un manifiesto asociado a un candidato, lo que culmina con la eliminación de dicho manifiesto. El sistema solicita confirmación para proceder. Si la respuesta es positiva se procede y el sistema muestra mensaje de culminación satisfactoria del proceso.

Curso alternativo: *Demitir*

Curso alternativo: *Intento de Demitir.*

Curso alternativo: *Error Interno*.

ix. Procesar sesión de análisis de caso de estudio dada una métrica

Curso normal: Un **super-usuario** puede comenzar el análisis de una campaña, basada en una métrica indicada por él mismo (de varias disponibles). El sistema responde acorde a la solicitud y muestra los resultados apropiados.

Curso alternativo: *Demitir.*

Curso alternativo: *Intento de Demitir.*

Curso alternativo: *Error Interno.*

x. Configurar cuenta propia

Curso normal: Un **super-usuario** solicita configurar su cuenta registrada en el sistema. El usuario puede acceder a cierta información asociada a su cuenta, como nombres o correo electrónico, con la intención de modificarla. Dicho usuario generará los cambios, acepta y retorna a la página inicial con las funcionalidades de visualización respectivas y los cambias realizados.

Curso alternativo: *Intento de Demitir.*

Curso alternativo: *Demitir*.

Curso alternativo: *Error interno*.

xi. Salir/Cerrar sesión

Curso normal: El **super-usuario** desea cerrar su sesión actual con el sistema. El sistema recarga la página y vuelve a la página inicial con, únicamente, las funcionalidades de visualización de datos.

Cursos alternativos en común.

- **Demitir:** El usuario demite y retorna a la página inicial.
- **Error Interno:** Un error interno, relacionado al servidor o la base de datos ha ocurrido. Se notifica al usuario, si es posible.
- **Error Sincronización (Interno):** Por cualquier motivo, se realiza una solicitud sobre una entidad u opción que parece estar disponible, pero en realidad no lo está. El sistema reaccionará notificando el problema, si es posible, y se regresará a la página inicial.

- **Intento de Demitir:** El **super-usuario** intenta demitir, pero se le solicita confirmación. Si la respuesta es positiva, se regresa a la página inicial. Caso contrario, se continúa en la página y se puede continuar.
- **Campo(s) Erróneo(s):** El **super-usuario** ingresa de manera errónea al menos un campo requerido. El sistema comunica el error y solicita nuevamente la entrada de los campos.

Diagrama de casos de uso.

En base a la descripción de los casos de estudio, la Figura 2 muestra el diagrama de casos de uso como resultado.

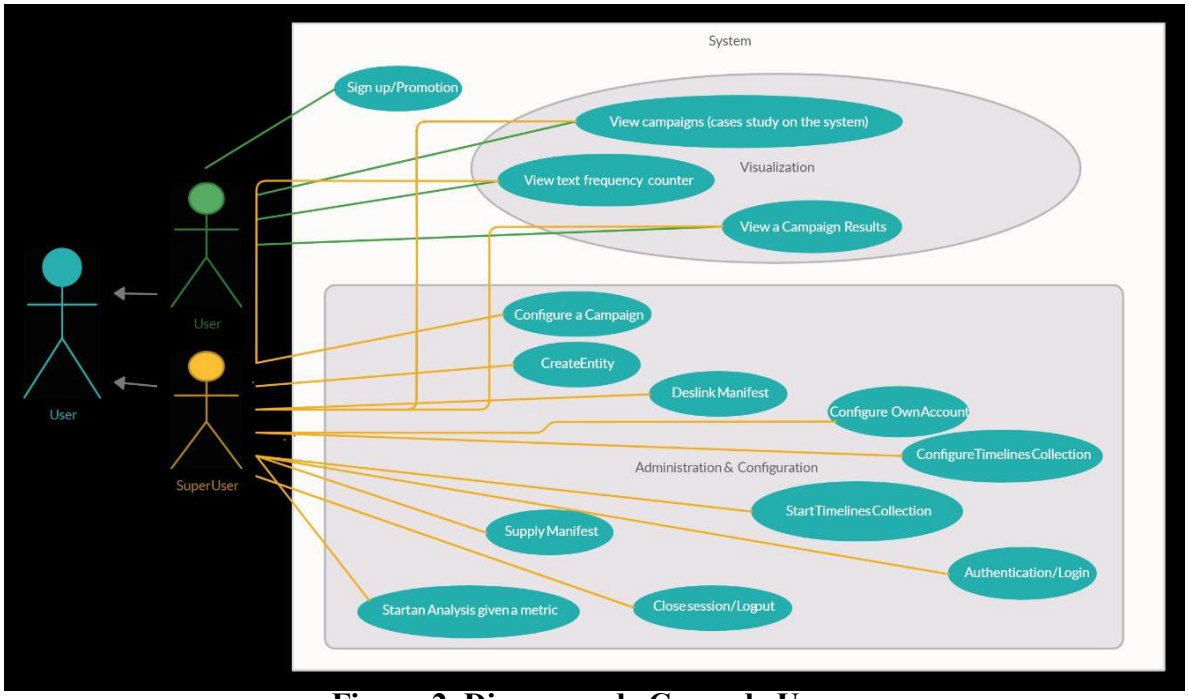

Figura 2. Diagrama de Casos de Uso

Identificación de clases.

Para proceder con la identificación de posibles clases relacionadas al sistema, se analizan y recopilan los sustantivos y frases nominales especificadas en los requerimientos y casos de uso.

Dominio del sistema.

La Tabla 1 muestra los sustantivos y frases nominales asociadas a las especificaciones de la aplicación

Prototipo de aplicación.	Manifiesto electoral	Campañas Online	Usuario
Correlación	Usuario lector y visualizador	Usuario configurador de información	Post
Timeline	Candidato	Menciones y reacciones	Screen name
Documento	Plan de gobierno	Colector	Red social
Campaña	Elección	Base de datos	Administrador
Pantalla	Teclado	Internet	Web
Computador personal	Dispositivo móvil	Sesión con la aplicación	Proceso de registro
Opción acorde al usuario	Texto	Entidad de BD.	Mensaje de éxito/error
Sistema	Usuario registrado	Credenciales	Pin/password
Cuenta propia	Página web	Menú Principal	Campos
Notificación	Posts recopilados	Usuario disponible	Notificación
Post/Tweet	Servidor	Formulario	Petición
Fuente de recopilación de datos	Identificador de usuario	API	Candidato Preexistente
PDF	Requerimiento	Tarea a realizar	Respuesta

Tabla 1. Dominio del Sistema (sustantivos y frases nominales)

Depuración del dominio del sistema.

A continuación, se procede con la depuración del dominio del sistema, tomando en cuenta los sustantivos relevantes, necesarios y relacionados con el objetivo del desarrollo

de la aplicación. En la Tabla 2 se muestran los sustantivos que se consideran posibles candidatos para ser una clase del sistema y que pueden ser modelados posteriormente.

Manifiesto Electoral (Plan)	Campaña	Usuario	Usuario lector/visualizador
Usuario configurador de información	Timeline (conjunto de posts)	Candidato	Documento
Colector timelines y documentos	Elección	Usuario administrador	Notificador de mensajes
Prototipo de aplicación	Base de datos	Cuenta de usuario (red social)	

Tabla 2. Dominio del Sistema depurado (potenciales clases)

Diagrama de Clases.

Para mayor legibilidad, la Figura 3 muestra un diagrama de clases con *elementos omitidos* (atributos y métodos). Este diagrama muestra las clases del sistema y sus relaciones estructurales.

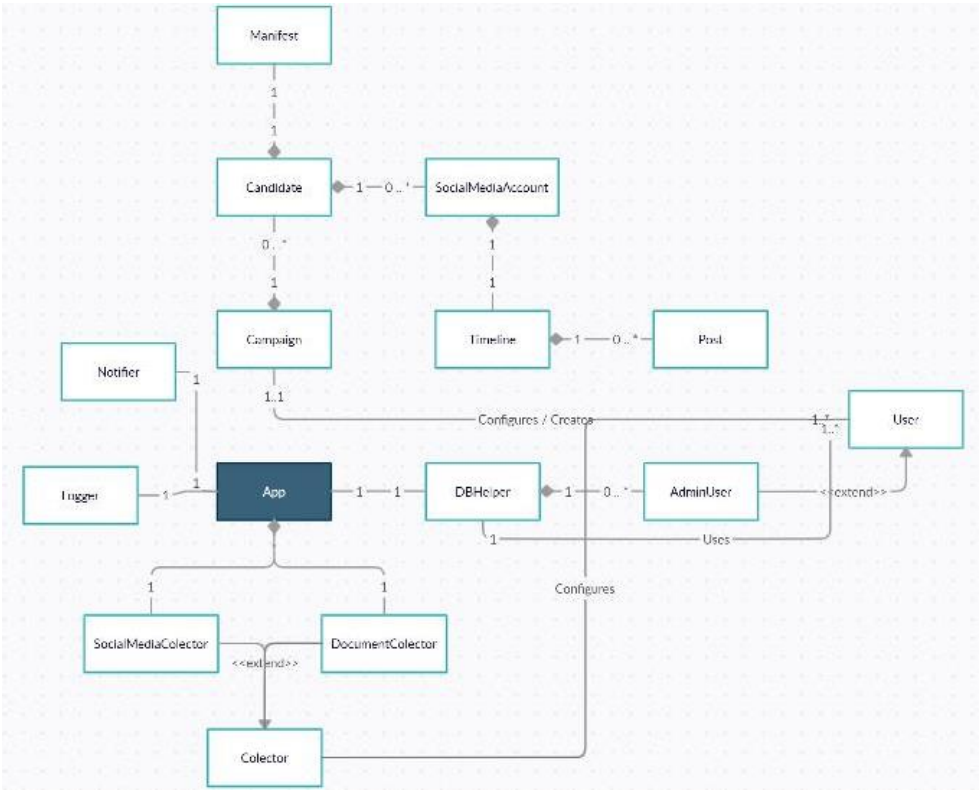

Figura 3. Diagrama de Clases con elementos omitidos (atributos y métodos).

Diagrama de clases (omisión de interrelaciones).

A continuación, en la Figura 4, se muestra un diagrama de clases con omisión de interrelaciones que muestra algunos atributos y comportamientos de las clases mostradas anteriormente (es posible que se hayan omitido varios atributos o comportamientos, los cuales pueden ser finiquitados en pasos posteriores y avance del proyecto). En este diagrama se omite las interrelaciones para no redundar con el diagrama anterior (Figura 3). Además, por este mismo motivo, no se coloca la *composición* de clases como un atributo. Por ejemplo, sabemos que una campaña puede poseer uno o varios candidatos; sin embargo, como esta relación está descrita en la Figura 3, no hace falta colocarla nuevamente como uno de los atributos de la clase.

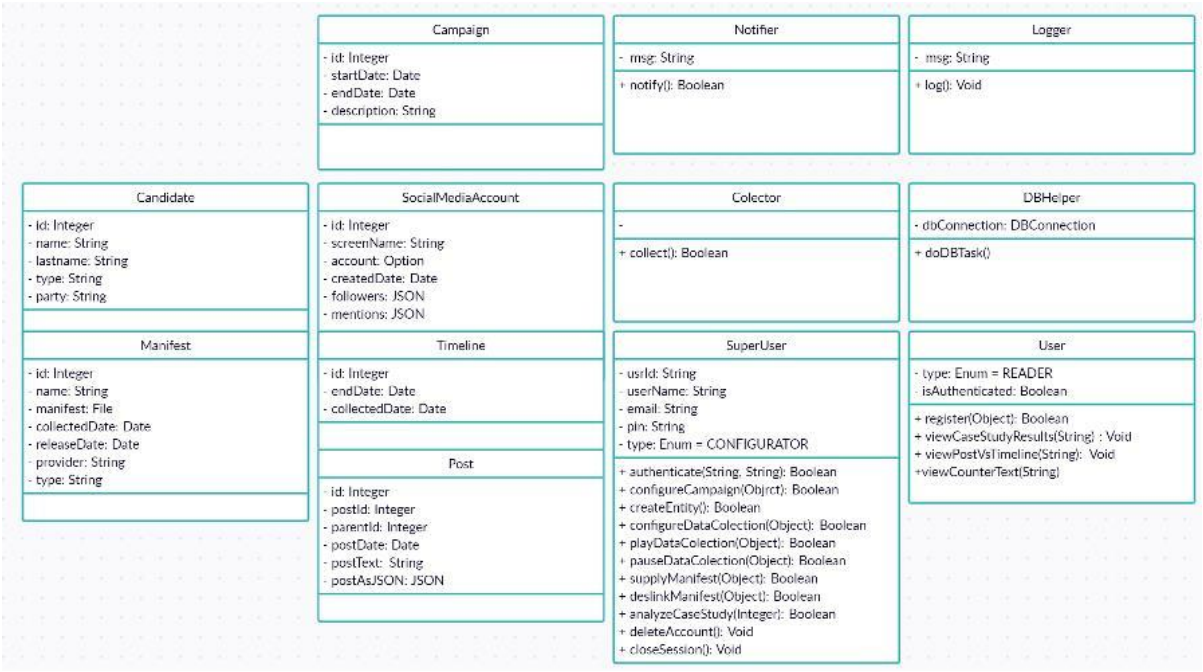

Figura 4. Diagrama de clases omitiendo interrelaciones.

Diseño de base de datos.

Descripción.

El sistema requiere de persistencia de varios datos. Además, se pretende que el almacenamiento sea de manera estructurada y relacional. En base a los requerimientos, el sistema especifica varias entidades que requieren ser almacenadas, ya que son partes esenciales o sujetos de estudio. Varias han sido identificadas como clases del sistema (ver Figuras 2 y 3).

Aunque el diseño y el análisis esté basado en la orientación a objetos, no se debe confundir con el modelo orientado a objetos para el diseño de base de datos. En la figura 5 se muestra el diagrama de Entidad-Relación (DER).

Entidades.

Las entidades que requieren ser modeladas en la base de datos son:

- **Caso de estudio:** entidad que dispone de una campaña en una relación uno a uno. Permita mayor abstracción para resolución del proyecto
- **Campaña:** entidad que dispone de fechas de inicio y final. Además, posee un nombre, una descripción y puede contener de uno a varios candidatos.
- **Candidato:** esta entidad posee uno o varios manifiestos (documentos) asociados con él. Además, tiene una asociación de uno a varias cuentas en redes sociales. Los atributos que lo definen son su nombre, su apellido, su ciudad, tipo y el partido al que pertenece.
- **Cuenta/Red Social:** una cuenta de una red social es una entidad que se refiere a un candidato a través de un identificador de la cuenta (screen name), una descripción, una fecha de creación (o recopilada) y un conjunto de posts, es decir, un timeline. Además, se busca una manera de almacenar a los

seguidores de la cuenta (posiblemente un JSON para luego ser transformado a grafo) y las menciones asociadas al nombre de usuario.

- **Timeline:** esta entidad corresponde a una cuenta en específico derivada de una red social, posee una relación de uno a muchos con los posts y una fecha del día en que fue recopilada y del final de la campaña. No corresponde literalmente al timeline de una cuenta, sino, más bien, a un conjunto de posts recopilados en un determinado tiempo.
- **Post:** una entidad específica de un posta de un correspondiente Timeline. Los atributos que lo identifican son la fecha de posteo, el id del "padre", el texto, el post en formato JSON.
- **Documento/Manifiesto:** esta entidad modela un plan de gobierno asociado a un candidato. Los campos que lo identifican son su fecha de recolección, la fecha de realización, el origen o proveedor y un tipo.

DER (Diagrama Entidad-Relación).

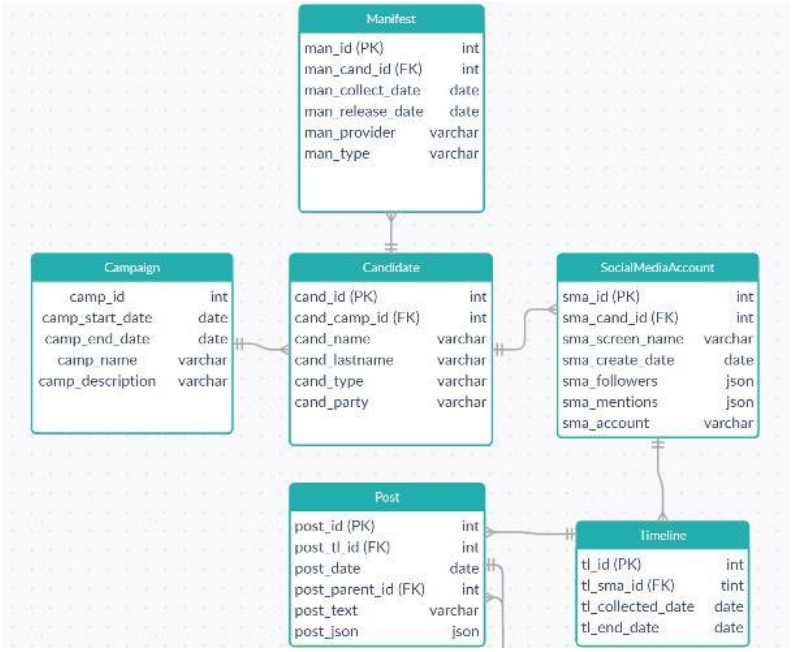

Figura 5. DER de la base de datos.

Diseño UI.

Arquitectura de la información.

La intención de la arquitectura de la información es promover tanto la usabilidad y funcionalidad, como la "encontrabilidad" (findability) de los puntos de interés del *espacio de la información* (básicamente una interfaz gráfica). Además, se procura maximizar la asimilación de la información mostrada y facilitar la ejecución de tareas por parte de los usuarios.

Así que, en base al análisis de requerimientos, y en búsqueda de que el usuario pueda percibir la información de la mejor manera, se determinan las siguiente "vistas" y componentes clave:

- Pantalla de inicio con opciones para registro/promoción, inicio de sesión y elección de un caso de estudio para ver funcionalidades subyacentes.
- Pantalla que presenta la información correspondiente a un caso de estudio y las opciones pertinentes asociadas a ese caso: similitud entre manifiestos y timelines recolectados, contador de frecuencias de palabras y porcentaje de posts directamente relacionados a un timeline.
- Formulario de registro, para **usuario** (tipo lector).
- Formulario de "sign in", para usuarios registrados (tipo configurador).
- Opciones extra de administración y configuración asociadas al **super-usuario**: configurar campaña, crear entidad, proveer y asociar/desligar manifiesto, configurar colección de datos, iniciar análisis dada una métrica, cerrar sesión y configurar cuenta.

Sitemap.

La Figura 6 muestra el sitemap (árbol de navegación), con la intención de mostrar visualmente la estructura de navegación que posee la aplicación.

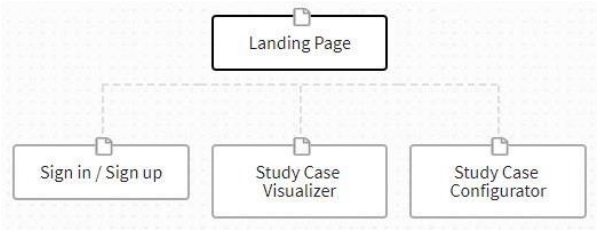

Figura 6. Árbol de Navegación de la Aplicación Web

Wireframes.

De la Figura 7 a la Figura 10 se muestran los wireframes creados que sirvieron como plantillas para disponer de una guía visual para el desarrollo de una vista.

Landing Page.

Figura 7. Landing Page Wireframe

Sign in / Sign up Page.

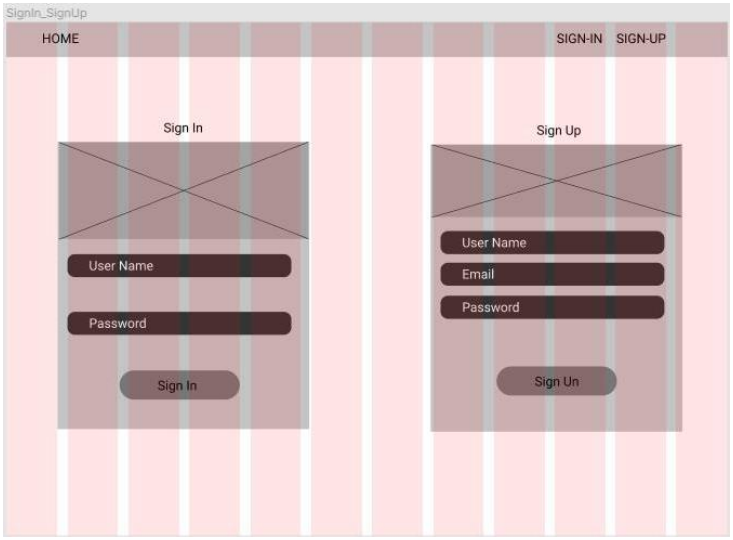

Figura 8. Sign In – Sign Up Wireframe

Study Case Visualizer.

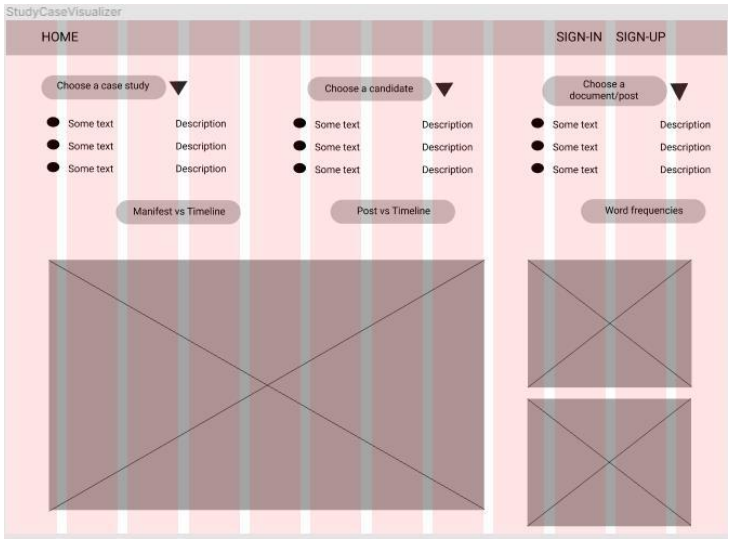

Figura 9. Study Case Visualizer Wireframe

Study Case Configurator.

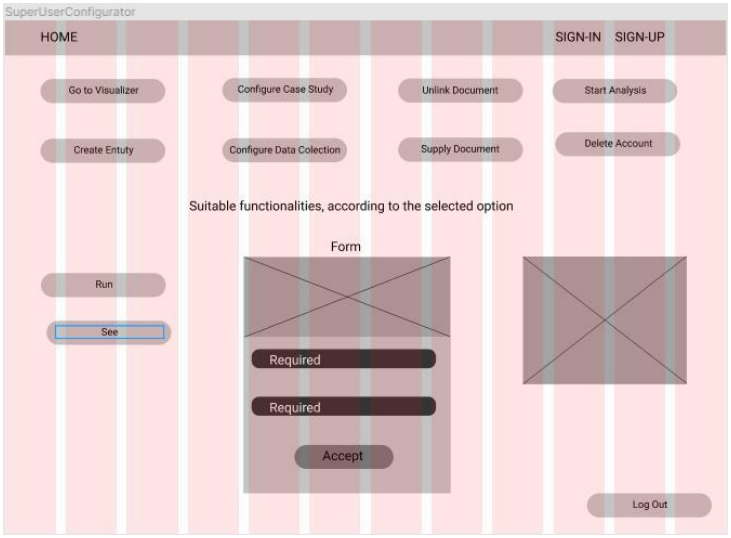

Figura 10. Study Case Configurator Wireframe

Mockup.

Enlace o anexo al mockup realizado en Figma:

https://www.figma.com/file/9dIgNw4SHSwe03UyOaOQhw/Mockup_Data_Analysis?node-id=0%3A1

Las imágenes utilizadas en el mockup son de libre uso y/o provenientes del paper *Electoral Manifestos and Online Campaign Analysis: Case Study – The 2019 Ecuadorian Sectional Elections*

Nota de diseño.

Todo proceso de diseño es iterativo, por ende, es posible que varios de los esquemas y pasos realizados, hasta el momento, tengan que ser revisitados con la intención de actualizarlos o ajustarlos a las necesidades que irán apareciendo con el tiempo. Es decir, se espera que, con el avanzar del tiempo, se descubran nuevas necesidades, se consideren algunas obsoletas, etc.

Proceso de desarrollo.

Resumen de versiones de tecnologías utilizadas.

La Tabla 3 muestra algunas de las tecnologías utilizadas (las más importantes) para la realización del proyecto y sus respectivas versiones.

Tecnología	V
Ubuntu	18.04.5
Python	3.8.3
Django	3.1.3
Matplotlib	3.2.2
Psycopg2	2.8.6
PyPDF2	1.26
Tweepy	3.9.0

Tabla 3. Resumen de versiones de tecnologías utilizadas

Django.

Django es un marco de trabajo para el desarrollo de aplicaciones web muy robusto. Debido a esto, varios aspectos de desarrollo son simplificados, con la intención de enfocarse en otras actividades menos comunes o específicas del trabajo a tratar. De acuerdo con la documentación del sitio oficial de Django, el framework maneja un conjunto de normas, pautas y capas (layers) que proveen un nivel de abstracción que encapsula varias de las responsabilidades más comunes en una aplicación web. Las capas son la capa de modelo (model), de vista (view), plantilla (template).

La capa de modelo ayuda en la estructuración y manipulación de los datos. De esta manera, Django ofrece una API automática de acceso y creación de bases de datos; donde cada entidad puede ser representada mediante una clase y cada campo de la

entidad corresponde a un atributo de dicha clase. Por su lado, la capa de vista es la responsable de procesar y devolver una respuesta a la solicitud de un usuario. Finalmente, la capa de plantilla proporciona una sintaxis sencilla para la representación de la información.

Por otro lado, Django también proporciona otras utilidades que facilitan el proceso de ciertos trabajos repetitivos, como la creación de formularios y la manipulación de los datos asociados al mismo o una sección automatizada y configurable de administración para acceso rápido a distintas configuraciones globales y a los modelos creados.

Proceso.

Para la realización del proyecto se utilizó la versión 3.1 de Django. Luego de instalar, se procedió con la creación del proyecto y las aplicaciones. Django trabaja con una estructura que facilita la reutilización del código, a través de la modularización. Para lograr este cometido, el framework permite la creación de un proyecto y aplicaciones. Un proyecto hace referencia a la aplicación completa y cada una de sus partes; mientras que una aplicación es un sub-módulo del proyecto. Cada aplicación, de cierta manera, es autosuficiente. Así, se procedió a crear la aplicación y dos aplicaciones con las siguientes sentencias, dentro del directorio que contiene el proyecto:

```
python manage.py startproject Manifests_and_Online_Campaigns_Analysis
python manage.py startapp StudyCasesConfApp
python manage.py startapp StudyCasesManage
```

Dentro del directorio del proyecto, se utilizaron los siguientes scripts: settings.py, urls.py y views.py. El primero permite indicar las distintas configuraciones del proyecto, como las aplicaciones a utilizar, los hosts permitidos para iniciar la aplicación, ubicación

de las plantillas o archivos estáticos, idioma y de más. El segundo script ayuda al mapeo directo de una vista con un patrón url correspondiente. El último tiene la intención de crear el controlador de las vistas (mediante funciones de python).

Las Figuras 11, 12 y 13 muestran ejemplos de cada uno de los scripts explicados anteriormente. La Figura 11 indica una parte de las configuraciones definidas en settings.py; específicamente, la configuración para el uso de plantillas. Por otro lado, la Figura 12 muestra la definición de algunas URLs en el script urls.py del proyecto, con la intención de señalar la manera en la que estas son determinadas. Finalmente, en la Figura 13 se exhibe un ejemplo sencillo acerca de un controlador definido en el script views.py del proyecto.

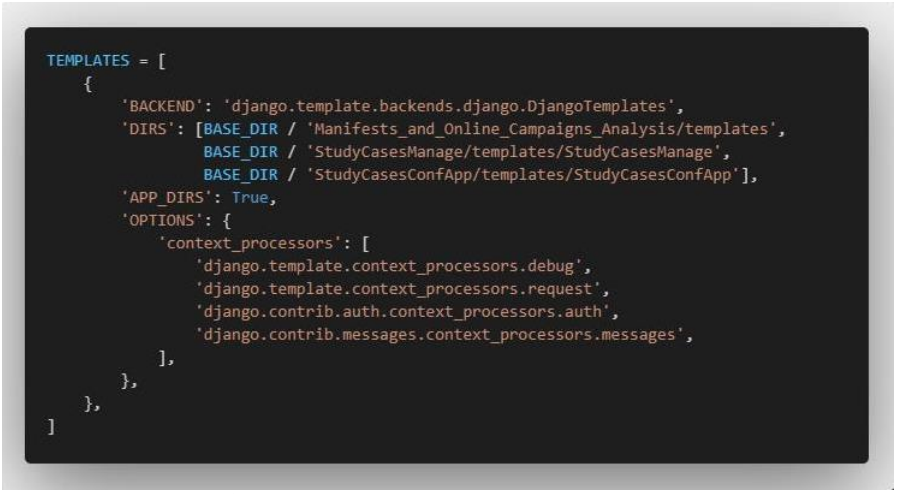

Figura 11. Configuración de las plantillas en settings.py

Figura 12. Uniform Resource Locators (URLs) definidos en urls.py

```
def edit_profile(request):
  if request.method == 'POST':
    form = UserChangeForm(request.POST, instance=request.user)

    if form.is_valid():
      form.save()
      return redirect(to='home')

  else:
    form = UserChangeForm(instance=request.user)
    args = {'form': form}
    return render(request, 'registration/edit-profile.html', args)
```

Figura 13. Ejemplo de un controlador dentro de views.py

En el caso de las aplicaciones, cada una está orientada a una parte específica del proyecto. La aplicación StudyCasesManage posee parte de la lógica del proyecto dentro del subdirectorio “logic”. En dicho subdirectorio se encuentran scripts con funciones para la recolección de datos, utilidades de la base de datos como obtener candidatos de una campaña dada y el procesamiento de los datos. Además, esta aplicación posee la definición de los modelos utilizados en la aplicación y sus propias urls, que luego son mapeados por urls.py del proyecto. Por su lado, la aplicación StudyCasesConfApp provee las plantillas utilizadas por la aplicación. Básicamente, se ofrecen plantillas orientadas a la configuración de distintas etapas del proyecto como las pantallas para la recolección de datos o para la muestra de resultados.

Formularios.

Django proporciona un marco que facilita la creación de formularios y la manipulación de datos de formularios. Por ejemplo, la Figura 14 presenta un formulario para la creación de campañas dentro de la aplicación. Además, es posible crear formularios a partir de los modelos creados, lo que facilita el trabajo en algunas ocasiones y permite reducir el código utilizado.

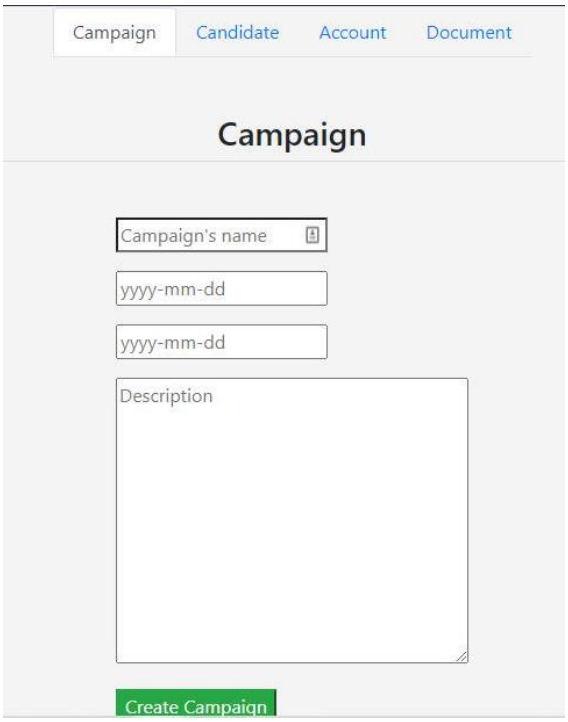

Figura 14. Formulario para crear una campaña

Estructura del Proyecto.

A continuación, en la Figura 15 se muestra la estructura de directorios principales que conforman la aplicación. Se puede apreciar el directorio del proyecto, las dos aplicaciones creadas y dos directorios para el almacenamiento de algunos de los recursos estáticos y recolectados (manifiestos).

Figura 15. Estructura del Proyecto

El directorio Manifests_and_Online_Campaigns_Analysis corresponde al contenido principal del proyecto, posee una vista, varias plantillas utilizadas en el resto del proyecto y las configuraciones globales. Por otro lado, los directorios StudyCasesConfApp y StudyCasesManage son las aplicaciones del proyecto, los

cuales poseen subdirectorios con sus propias plantillas, vistas y urls. En el caso de StudyCasesManage también existe un script (admin.py), que configura y registra la visualización, en la sección de administración, de los modelos creados y posee, además, el subdirectorio "migrations" y "logic", que es la manera en la que Django indica los cambios realizados sobre los modelos, como agregar campos, eliminar modelos, etc. para poder actualizarlos en el esquema de la base de datos posteriormente con las instrucciones **python manage.py makemigrations** y **python manage.py migrate** y varios scripts asociados a la lógica y el Backend de la aplicación, respectivamente**.** Por último, los directorios *media* y *static* son orientados al almacenamiento de manifiestos/documentos y de archivos estáticos, respectivamente.

Subdirectorio "logic".

El subdirectorio *logic*, perteneciente al directorio y aplicación *StudyCasesManage*, posee scripts que ofrecen distintos servicios y utilidades necesarias para el proyecto. En específico, el script *ea_data_process.py* posee funciones orientadas al análisis de los datos, como la recolección del contenido de los posts y de los manifiestos para su posterior análisis; es por esto que este script también realiza el cálculo de similaridad entre ambos, a partir de los vectores de frecuencias por palabras. Por su parte, el script *ea_db_utilities.py* posee varias funciones que abstraen, aún más, aspectos relacionados con el modelo de la aplicación, especialmente, en la recolección de datos. Finalmente, el script *ea_get_time_lines.py* posee el enlace directo con la API de twitter para la recolección de posts de los candidatos del modelo (de acuerdo con la campaña elegida para la recolección).

Plantillas.

Para la creación de las plantillas se utilizaron el lenguaje de marcado HTML y las hojas de estilo CSS. Además, se utilizó Bootstrap 4 como framework open source, que permite realizar de manera más sencilla el desarrollo del front-end y que, a la vez, este sea más responsivo, es decir, con apariencia para múltiples tamaños de pantalla. Gracias a Django, es posible realizar herencia e incrustación de plantillas. Esto último permite reutilización de código y cierto nivel de abstracción en cuanto a las diferentes partes que conforman una plantilla. Por ejemplo, la barra de navegación y el pie de página son plantillas reutilizadas en distintas plantillas, como el *home* o el *configurator*.

Por otra parte, existen varias plantillas completas (o secciones) que disponen una restricción. Dicha restricción consiste en que el usuario debe estar registrado y haber pasado un proceso de login (sign-in en la vista). Esto es realizado con la intención de proveer, al menos, una capa de seguridad mínima, con la intención de salvaguardar la integridad de la base de datos y otras operaciones ligadas con la aplicación.

Administración.

Django provee un sitio de Administración que es una interfaz muy versátil. Esta interfaz lee la metadata contenida en los modelos creados para mostrarlos y permitir la manipulación y administración de dichos modelos. Además, la interfaz se convierte en una herramienta para la gestión interna del sitio creado. Por otra parte, esta interfaz puede ser configurada desde un script (por convención, admin.py) para mostrar los modelos y atributos deseados. Incluso, se permite indicar el orden en el que los atributos son exhibidos y los métodos de búsqueda y filtrado de los modelos. Para acceder a este sitio se puede usar el siguiente enlace http://xterm.mynetgear.com:9191/admin/

Caso de estudio.

Con la aplicación en ejecución, se procedió a crear el caso de estudio de ejemplo para examinar el funcionamiento de la aplicación. A continuación, se muestran las entidades creadas, desde el Front-End de la aplicación. Las Figuras 16, 17, 18 y 19 muestran las entidades de tipo Campaña, Candidato, Cuenta y Documento creadas, respectivamente. Las imágenes corresponden a visualizaciones del panel de administración.

NAME	START DATE	END DATE	DESCRIPTION
Elecciones Presidenciales ECU. 2021	Nov. 12, 2020	Feb. 7, 2021	Elecciones Presidenciales Ecuador: Próximo proceso electoral ecuatoriano a realizarse el 7 de febrero de 2021 para elegir al 81° Presidente Constitucional y 52° Vicepresidente Constitucional de la República del Ecuador para el período 2021-2025.

1 campaign

Figura 16. Campaña del caso de estudio

NAME	LASTNAME	CAMPAIGN	TYPE	PARTY
Ximena	Peña	Elecciones Presidenciales ECU. 2021	Presidente	Alianza PAIS
Guillermo	Celi	Elecciones Presidenciales ECU. 2021	Presidente	SUMA
Isidro	Romero	Elecciones Presidenciales ECU. 2021	Presidente	Partido Avanza
Gustavo	Larrea	Elecciones Presidenciales ECU. 2021	Presidente	Movimiento Democracia SI
Xavier	Hervas	Elecciones Presidenciales ECU. 2021	Presidente	Partido Izquierda Democrática
Andrés	Araúz	Elecciones Presidenciales ECU. 2021	Presidente	Unión por la Esperanza
Yaku	Pérez	Elecciones Presidenciales ECU. 2021	Presidente	Pachakutik
Juan	Velasco	Elecciones Presidenciales ECU. 2021	Presidente	Movimiento Construye
César	Montúfar	Elecciones Presidenciales ECU. 2021	Presidente	Partido Socialista Ecuatoriano - Concentración
Lucio	Gutiérrez	Elecciones Presidenciales ECU. 2021	Presidente	Partido Sociedad Patriótica
Guillermo	Lasso	Elecciones Presidenciales ECU. 2021	Presidente	Alianza CREO-PSC

Figura 17. Candidatos dentro del caso de estudio

CANDIDATE	SCREEN NAME	CREATED DATE	ACCOUNT
Ximena Peña. (Elecciones Presidenciales ECU. 2021)	@XimenaPenaP	-	Twitter
Guillermo Celi. (Elecciones Presidenciales ECU. 2021)	@GuillermoCeli	-	Twitter
Isidro Romero. (Elecciones Presidenciales ECU. 2021)	@isidroromero_c	-	Twitter
Gustavo Larrea. (Elecciones Presidenciales ECU. 2021)	@GustavoLarreaSi	-	Twitter
Xavier Hervas. (Elecciones Presidenciales ECU. 2021)	@xhervas	-	Twitter
Andrés Araúz. (Elecciones Presidenciales ECU. 2021)	@ecuarauz	-	Twitter
Yaku Pérez. (Elecciones Presidenciales ECU. 2021)	@yakuperezg	-	Twitter
Juan Velasco. (Elecciones Presidenciales ECU. 2021)	@juanfervelasco	-	Twitter
César Montúfar. (Elecciones Presidenciales ECU. 2021)	@CesarMontufar51	-	Twitter
Lucio Gutiérrez. (Elecciones Presidenciales ECU. 2021)	@LucioGutierrez3	-	Twitter
Guillermo Lasso. (Elecciones Presidenciales ECU. 2021)	@LassoGuillermo	-	Twitter

Figura 18. Cuentas asociadas a los candidatos del caso de estudio

CANDIDATE	MANIFEST
Guillermo Celi. (Elecciones Presidenciales ECU. 2021)	StudyCasesManage/manifestos/plan_trabajo_celi_y_sevilla_compressed.p
Isidro Romero. (Elecciones Presidenciales ECU. 2021)	StudyCasesManage/manifestos/Propuesta_de_Plan_de_Trabajo_AVANZA
Gustavo Larrea. (Elecciones Presidenciales ECU. 2021)	StudyCasesManage/manifestos/PLAN_DE_GOBIERNO_DEMOCRACIA_SI.p
Xavier Hervas. (Elecciones Presidenciales ECU. 2021)	StudyCasesManage/manifestos/PLAN_DE_GOBIERNO_2021_-_2025_IZQUIERDA_DEMOCRATICA.pdf
Juan Velasco. (Elecciones Presidenciales ECU. 2021)	StudyCasesManage/manifestos/Plan_de_Gobierno_CONSTRUYE_25.pdf
César Montúfar. (Elecciones Presidenciales ECU. 2021)	StudyCasesManage/manifestos/PLAN_DE_GOBIERNO_Presidencial_17-51

Figura 19. Documentos (Manifiestos) del caso de estudio

Despliegue.

El despliegue de la aplicación web se realizó en un servidor Ubuntu versión 18.04.5 LTS. Actualmente, la aplicación es ejecutada, en background, por el propio web-server de Django. Se utiliza la versión DEBUG del proyecto (lo cual se indica en el archivo settings.py del proyecto).

Url.

La URL actual, para acceder a la aplicación, es la siguiente:

http://xterm.mynetgear.com:9191/StudyCasesManage/

Recolección.

El proceso para la recolección de datos fue configurado desde el 10 de noviembre del 2020 hasta el 31 de diciembre del mismo año. En la Figura 20 se puede visualizar la manera en la que se configuró la recolección de datos (posts).

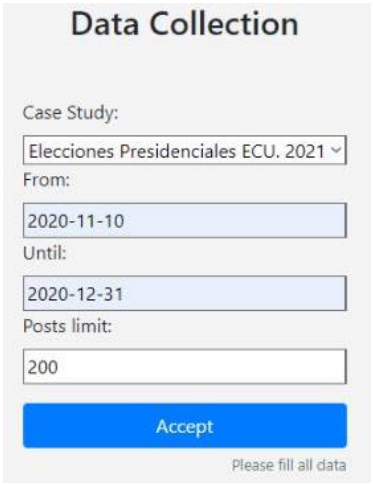

Figura 20. Evidencia de configuración para la recolección de datos

RESULTADOS Y DISCUSIÓN

Resultados

Pantallas-Vistas.

En cuanto al Front-End respecta, y basado en el diseño realizado, se obtuvieron las siguientes pantallas principales: Home, Configurator y Sign-In.

La pantalla principal, al igual que varias otras, posee la barra de navegación (enlaces a la página principal, *Configurator* y de *Sign-in*), el pie de página y contenido que indica un resumen de los casos creados (campañas/casos de uso) que posee el sistema hasta el momento. Esto puede verse en la Figura 21.

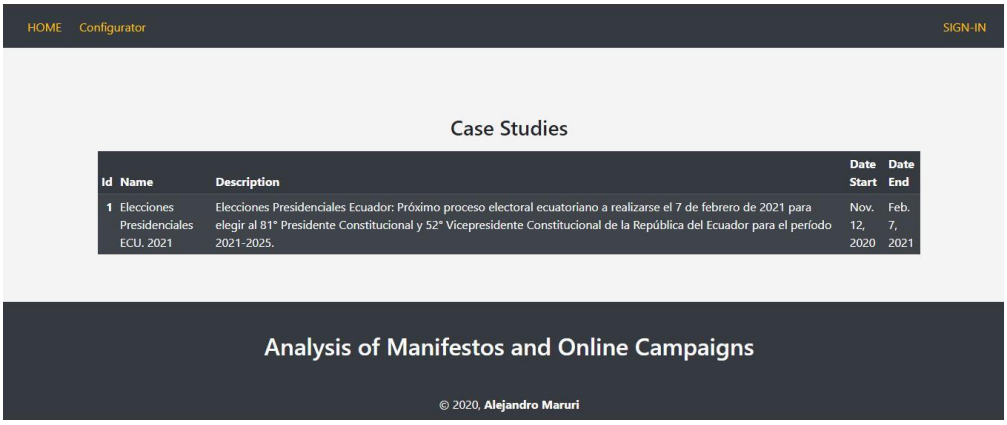

Figura 21. Vista de la página principal (Home)

La vista *Configurator* posee un menú con distintas opciones: “Campaign Configuration”, “Data Collection”, “Analysis”, “Account”, tal y como se puede apreciar en la Figura 22.

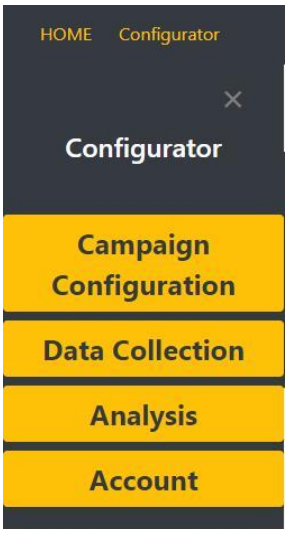

Figura 22. Menú de la página "Configurator"

Campaign Configurator.

Esta sección permite al usuario autenticado crear entidades como campañas y candidatos; además de permitir la asociación de cuentas de redes sociales y manifiestos con un respectivo candidato. Por ejemplo, la Figura 23 muestra el formulario que permite asociar una cuenta de red social con un candidato.

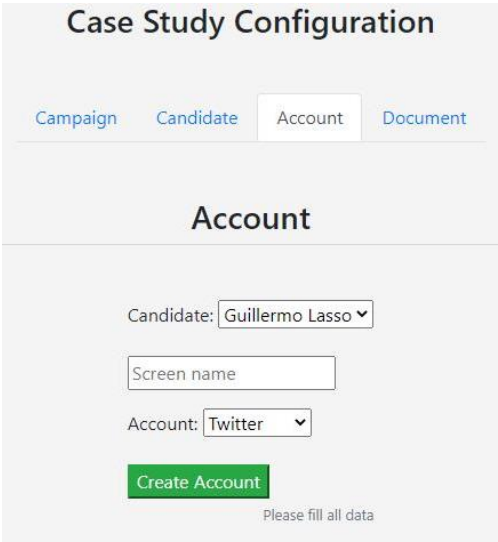

Figura 23. Formulario para la asociación de cuenta de red social con un candidato

Data Collection.

Data Collection corresponde a una opción del menú de *Configurator* que permite configurar la recolección de datos de una campaña en específico (de los candidatos asociados a dicha campaña) dada una fecha de inicio y una de fin. El formulario que permite configurar dicha recolección se muestra en la Figura 24.

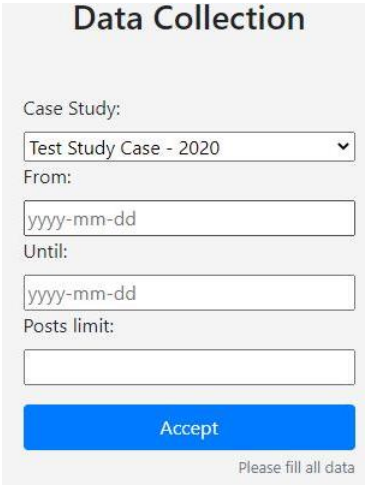

Figura 24. Formulario para configuración de la recolección de datos

Analysis.

Analysis es una opción del menú *Configurator* que permite comenzar un proceso de análisis con los datos recolectados hasta el momento. El proceso para generar el resultado es sencillo y puede apreciarse mediante la Figura 25.

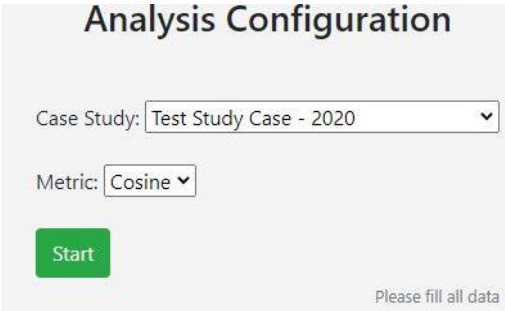

Figura 25. Formulario para iniciar un análisis

Account.

La opción Account del menú *Configurator* permite al usuario parametrizar (configurar) algunos campos relacionados con su cuenta de usuario como nombres,

contraseña y correo electrónico. Así, la Figura 26 muestra parte de la página con información de un usuario y los campos que pueden ser modificados por él/ella.

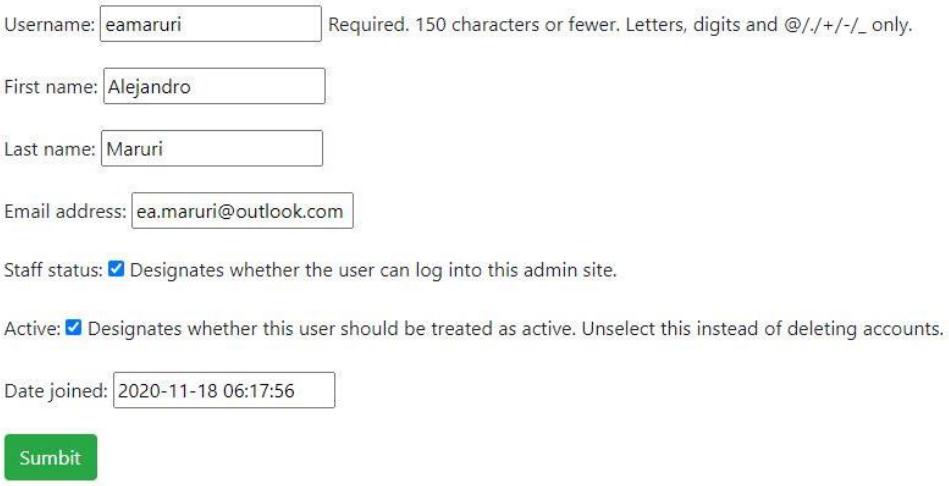

Figura 26. Ejemplo de configuración de cuenta de usuario

Por su lado, la pantalla de Sign-In, mostrada en la Figura 27, permite a los usuarios, previamente registrados, ingresar al sistema y, por ende, a ciertas vistas o secciones restringidas para un usuario no autenticado, como la página de Configuración o secciones de la página de inicio.

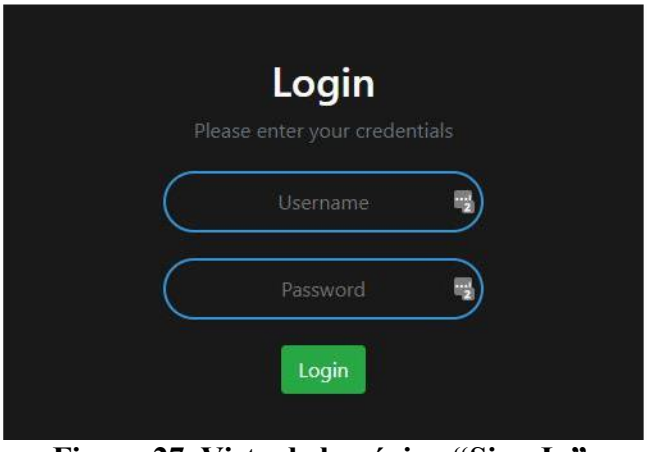

Figura 27. Vista de la página "Sign-In"

Diferencias con diseño planteado.

A simple vista puede parecer que no hay relación entre el diseño realizado y el resultado final del Front-End. Sin embargo, hay que tomar en cuenta que el diseño es una base y un proceso iterativo que puede ir cambiando y evolucionando durante el mismo proceso de diseño o el proceso de desarrollo. Esto se debe a que el proyecto se torna más tangible y varias cosas pueden ser cambiadas, mezcladas, eliminadas, etc. con el tiempo. Es por esto por lo que, por ejemplo, la página *Configurator* no es similar al diseño, ya que varias funcionalidades se juntaron en una única opción y además se deseaba presentar de mejor manera el menú de opciones y los resultados de cada uno.

Resultados Caso de Uso.

Para mostrar resultados del análisis de posts y manifiestos, se utiliza un plot de barras que muestra la similaridad entre ambos basado en la métrica del coseno. Un resultado parcial obtenido se puede ver en la Figura 28.

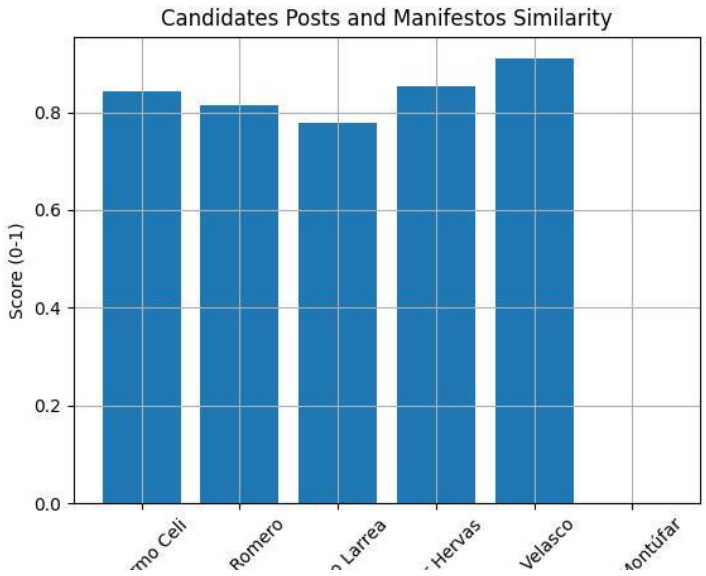

Figura 28. Resultado del análisis del 2 de diciembre del 2020

Trabajo Futuro

El trabajo actual presenta una idea y resolución más concreta y, hasta cierto punto, más flexible y avanzada sobre el prototipo planteado por Riofrío et al, en el estudio *Electoral Manifestos and Online Campaign Analysis: Case Study – The 2019 Ecuadorian Sectional Elections*. Como se indica en dicho documento, la intención es permitir la comparación entre las campañas online de los candidatos con sus respectivos manifiestos electorales. Sin embargo, el presente proyecto tiene una finalidad más grande, en cuanto a flexibilidad de recolección de datos y ejecución de ciertos procesos respecta. Así, aunque el resultado del proyecto es satisfactorio, surgen distintas actividades que pretenden mejorar y/o complementar lo realizado hasta el momento.

Entre las principales actividades futuras tenemos las siguientes:

- Implementación de otras métricas, distintas a las del coseno, para realizar análisis sobre los datos recolectados y reportar oportunamente el resultado.
- Mejoramiento de la presentación de las tablas en el Front-End y que permita ordenamiento por columna, filtrado y búsqueda.
- Implementación de recolección de datos (posts) provenientes de otras redes sociales como Facebook e Instagram.
- Asociación de entidades existentes a entidades superiores. Por ejemplo, un candidato creado puede ser utilizado a una nueva campaña.

En general, se desea estandarizar, aún más, la aplicación, con la intención de generar una herramienta que sea publicable en internet, para el uso y estudio de las campañas online de los candidatos a próximas elecciones. Además, es importante la intención de esta aplicación, ya que pretende extenderse para realizar distintos tipos de análisis

mediante métricas de minería de texto para evaluar de diferentes maneras la política y el uso de redes sociales.

Problemas conocidos.

Despliegue.

Actualmente, el proyecto se encuentra desplegado y en ejecución sobre el propio servidor web de Django; sin embargo, este no es lo suficientemente eficiente ni recomendado para una versión en etapa de producción. El problema surge debido a que hubo problemas al momento de desplegar la aplicación en un servidor web como NGINX o Apache. Puede deberse a falta de conocimiento o experiencia.

Actualización de manifiestos.

El formulario de creación y asignación de documentos a un candidato determinado permite asociar los manifiestos una única vez. A partir de la primera asignación, ya no es posible cambiar de documento, a menos que se realice el proceso desde la sección de administración.

Levantamiento automático procesos.

Debido a que el servidor web que despliega la aplicación requiere de una instrucción explícita para que sea ejecutado, los procesos, como la recolección de datos, se ven afectados. Por ejemplo, en el momento en el que, por cualquier motivo, el servidor se apague o se suspenda, dichos procesos también lo harían. Es por esto por lo que se plantea, como trabajo futuro, la creación de tareas en background que se almacenen en la base de datos, con la intención de que puedan ser levantadas y/o ejecutadas automáticamente, en caso de ser necesario, cuando el servidor se reinicie.

Propuestas de Solución:

Existen varias soluciones y/o perspectivas para solventar este problema. Básicamente lo que se busca es generar **scheduled tasks** (tareas programadas). Es posible crear tareas programadas de distintas maneras y que sean ejecutadas automáticamente dado ciertos tiempos o intervalos. A continuación, se presentarán dos maneras posibles.

1. **Uso de Celery**

   Celery es una herramienta de Python que permite ejecutar código de manera pospuesta a manera de un proceso separado. Esta librería permite indicar que ciertas funciones de Python sean manejadas como tareas mediante una instancia de Celery. Dependiendo la versión de Django, se requerirá una versión de Celery. En este caso, es recomendable utilizar la versión 4.0 de Celery (última versión en el momento en que se escribe este documento) que soporta versiones de Django de 1.8 en adelante. Para más información y tutoriales de cómo implementar las tareas programadas con Celery, se puede revisar el siguiente enlace: https://docs.celeryproject.org/en/latest/getting-started/first-steps-with-celery.html#installing-celery

2. **Django-Background-Tasks**

   Una solución considerada más oportuna es utilizar las tareas en background de Django. Esta solución proporciona una cola de tareas respaldadas por una base de datos. Cada tarea de **Django-Background-Tasks** corresponde a una función de Python.

> Básicamente, se debe seguir dos pasos esenciales. Primero, se crea la tarea (función) y se la registra con el planificador (scheduler). Segundo, configurar una tarea del tipo **cron** que permita ejecutar las tareas del planificador. Para más información de este método se puede revisar el siguiente enlace: https://django-background-tasks.readthedocs.io/en/latest/

Verificar cuenta para recolección de timeline.

Aunque la aplicación web permita indicar la red social a la que pertenece la cuenta del candidato, ya sea Facebook, Twitter o Instagram; la aplicación únicamente actúa con la API de Twitter. Además, no se verifica que sea del tipo Twitter para proceder con la recolección. Por lo tanto, sería importante poder corroborar y verificar esta información para proceder con la recolección. Además, es importante considerar la implementación de las recolecciones correspondientes a las otras redes sociales (Facebook e Instagram) para futuros trabajos.

CONCLUSIONES

El presente trabajo exhibe una propuesta interesante para el proceso de análisis sobre el impacto de las campañas online y sus relaciones con los planes de gobierno respectivos, asociados a un candidato. Debido a que se trata de un proyecto Full-Stack, el diseño del mismo se orientó tanto al Front-End, como al Back-End. Tal proceso de diseño contribuyó a la etapa de desarrollo; ya que, además de brindar una idea general del camino a tomar para la realización del proyecto, permite tener una trazabilidad sobre lo creado y que, de cierta manera, permita escalar el proyecto mediante trabajos futuros.

En general, se obtuvo un desenlace satisfactorio, pues se alcanzó lo planteado. La intención era conseguir una aplicación web que permita la recolección más flexible de datos y un proceso más sencillo para la ejecución de procesos de análisis. Aunque se tiene una aplicación web funcional, que cumple con lo propuesto, el proyecto no está totalmente consumado. Esto se debe a que existe más trabajo por realizar en cuanto a la implementación y el aprovisionamiento de otras funcionalidades, como el resultado de análisis con nuevas métricas o la recolección de datos provenientes de otras redes sociales como Facebook o Instagram. Por otra parte, es importante mencionar que hubo varias dificultades como el despliegue o el levantamiento automático de tareas. Sin embargo, esto es tratado brevemente, proponiendo como trabajo futuro en la sección pertinente de este texto.

REFERENCIAS BIBLIOGRÁFICAS

A. (2019, 2 mayo). *Qué es un Framework*. Lenguajes de programación. Recuperado el 16 de octubre de 2020, de https://lenguajesdeprogramacion.net/diccionario/que-es-un-framework/

Cantú, A. (2020, 25 julio). *Qué es: UX y UI*. Andrea Cantú. Recuperado el 3 de noviembre de 2020, de https://blog.acantu.com/que-es-ux-y-ui/

Chart, Diagram & Visual Canvas Software. (2020). Creately. https://creately.com/

Content, R. R. (2019, 7 junio). *Glosario de Redes Sociales: 176 términos que necesitas conocer*. Rock Content. Recuperado el 28 de octubre de 2020, de https://rockcontent.com/es/blog/glosario-de-redes-sociales/

Deitel, P. J., & Deitel, H. M. (2007). *C++ How to Program* (6.ª ed.). Prentice Hall.

Documentación de Django. (2020). Recuperado el 20 de octubre de 2020, de Django Project website: https://docs.djangoproject.com/es/3.1/

Figma: the collaborative interface design tool. (2020). Figma. https://www.figma.com/

Informática Básica: ¿Qué son las aplicaciones web? (s. f.). GCFGlobal.org. Recuperado el 26 de octubre de 2020, de https://edu.gcfglobal.org/es/informatica-basica/que-son-las-aplicaciones-web/1/

Krug, S. (2005). *Don't Make Me Think: A Common Sense Approach To The Web Usability* (2.ª ed.). New Riders Pub.

M. Kroenke, D., & J. Auer, D. (2014). *Database Processing, International Edition* (13.ª ed.). Pearson.

PostgreSQL: Documentation. (s. f.). The PostgreSQL Global Development Group. Recuperado el 3 de noviembre de 2020, de https://www.postgresql.org/docs/

Riofrío, D, Almeida, P., Dávalos, J. Flores Moyano, R., Pérez, N., Benitez, D. S., & Medina-Pérez, P. (2019). *Electoral Manifestos and Online Campaign Analysis: Case Study – The 2019 Ecuadorian Sectional Elections* (Pregrado). Universidad San Francisco de Quito USFQ.

R&C Consulting. (2019, 9 julio). *¿Cómo elaborar un Plan de Gobierno exitoso?* Blog R&C Consulting. Recuperado el 25 de octubre de 2020, de https://rc-consulting.org/blog/2018/03/como-elaborar-un-plan-de-gobierno/

writemaps.com. (2020). *WriteMaps | Create Sitemaps Online*. WriteMaps. https://writemaps.com/