**PROYECTO FINAL:**

Noti – Escuela

Sebastian Londoño Medina

**Arquitectura en la nube**

2025

**Docente:** Rómulo Betancourt Hortua

**Documentación:**

**Estructura y contenido:**

1. **Definiciones y especificación de requerimientos:**

**Definición general del proyecto:**

Noti Escuela es una plataforma web diseñada para optimizar la comunicación bidireccional entre instituciones educativas y los padres de familia. A través de un sistema centralizado, permite el envió ágil de avisos, recordatorios, noticias institucionales y actividades académicas, garantizando una difusión oportuna y organizada de la información.

La plataforma incorpora funcionalidades interactivas que habilitan a los padres a comentar y responder en tiempo real, fomentando la participación activa en la comunidad educativa. Además, al reemplazar los canales tradicionales basados en papel, Noti Escuela contribuye a la reducción del consumo de recursos físicos, promoviendo el uso responsable de tecnologías digitales y aplicando practicas sostenibles.

**Definiciones técnicas:**

El desarrollo de la aplicación web se realizo utilizando un microframework de Python minimalista ideal para construir aplicaciones web ligeras y escalables, Flask. El cual nos proporcionó la lógica del backend, manejo de rutas y solicitudes HTTP (GET, POST, etc.)

La capa de presentación fue implementada mediante HTML5 y CSS3, con una estructura sencilla y amigable con los usuarios para que pueda ser visto desde varios dispositivos. Para la persistencia de datos, se empleó PostgreSQL ya que su utilidad de gestión de bases de datos relacionales es robusta y soporta tipos de datos avanzados. La conexión entre Flask y PostgreSQL se gestionó con SQLAlchemy, que nos facilito la manipulación de datos mediante clases y objetos de Python, mejorando la mantenibilidad y seguridad del código.

Finalmente, la aplicación fue desplegada con Render, una plataforma tipo PaaS que nos permitió alojar la aplicación de forma sencilla y gratuita.

**Procedimientos de instalación y prueba:**

Para que todo surja de manera correcta se tienen que instalar lo siguiente:

* Python 3.9
* Flask

(*pip install Flask*

*pip install psycopg2-binary # Para conectar con PostgreSQL*

*pip install SQLAlchemy # ORM para manejar la base de datos*

*pip install Flask-SQLAlchemy # Extensión de Flask para SQLAlchemy* )

* PostgreSQL
* Editor de código o IDE (Visual Studio Code)
* Preparar despliegue en Render
* Repositorio en GitHub
* Descargar Git, psycopg2-binary, venv

1. **Arquitectura del sistema**

[Usuario (Padre / Docente / Administrador)]

↓

[HTTPS / CDN]

↓

[Aplicación Flask (Backend)]

↓

[Base de Datos PostgreSQL]

↓

[Almacenamiento de archivos (Opcional)]

**Frontend (Cliente):** Acceso mediante navegadores web desde cualquier dispositivo buscando una buena experiencia de usuario y uso fácil de la aplicación web.

**Backend (Aplicación con Flask):** Desplegada con un servicio web en una PaaS (Render) con dependencias gestionadas, variables de entorno para una configuración segura (Clave secreta, URL de BD) y finalmente PostgreSQL.

* **Esquema de BD que incluye tablas para:**
* Roles
* Comentarios
* Historial de comentarios
* Sistema de login con usuario y contraseña
* Almacenamiento de archivos
* Repositorio de GitHub para despliegue en render

Navegador

(usuario)

Load Balancer

(HTTPS /CDN)

PostgreSQL (BD gestionado)

(usuarios, avisos, comentarios, permisos)

Servidor Flask (Render)

(Rutas, lógica, autenticaciones)

**Dependencias externas:**

Archivo requirements.txt con todas las dependencias de la aplicación web:

flask

gunicorn

flask\_sqlalchemy

flask\_admin

psycopg2-binary

pillow

**Librerías Flask:**

Librerías contenidas en el archivo app.py en donde se encuentran todas las librerías del microframework Flask para la conexión de BD, admin, registros, renderizar los templates, etc.

from flask import Flask, request, render\_template, redirect, url\_for, session, jsonify

from markupsafe import Markup

from flask\_sqlalchemy import SQLAlchemy

from flask\_admin import Admin

from flask\_admin.contrib.sqla import ModelView

from flask\_admin import AdminIndexView

from flask\_admin.menu import MenuLink

from flask\_admin.form import ImageUploadField

import os

from wtforms.fields import SelectField

from datetime import datetime

from models import db, Comentario

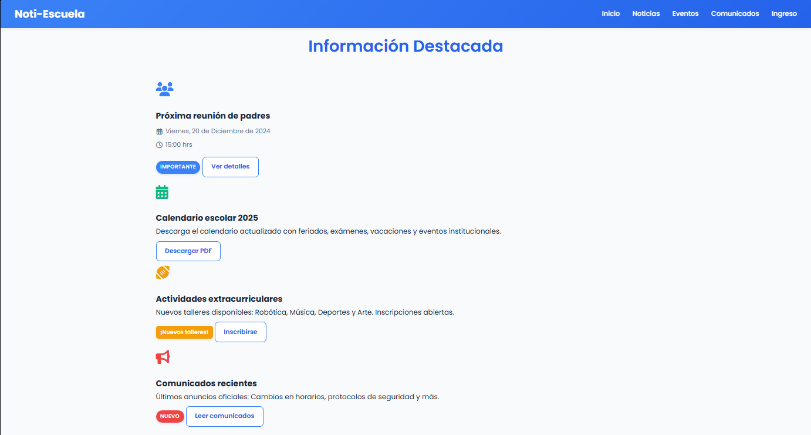
import time

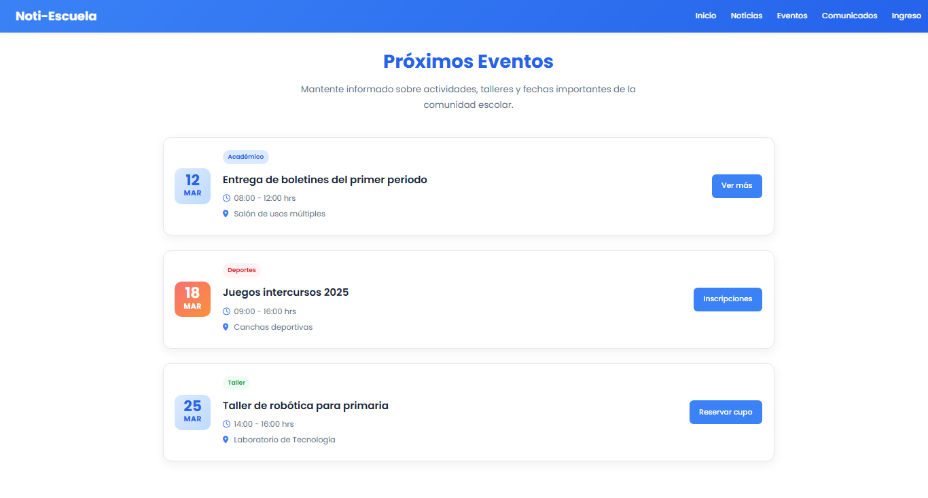
import psycopg2

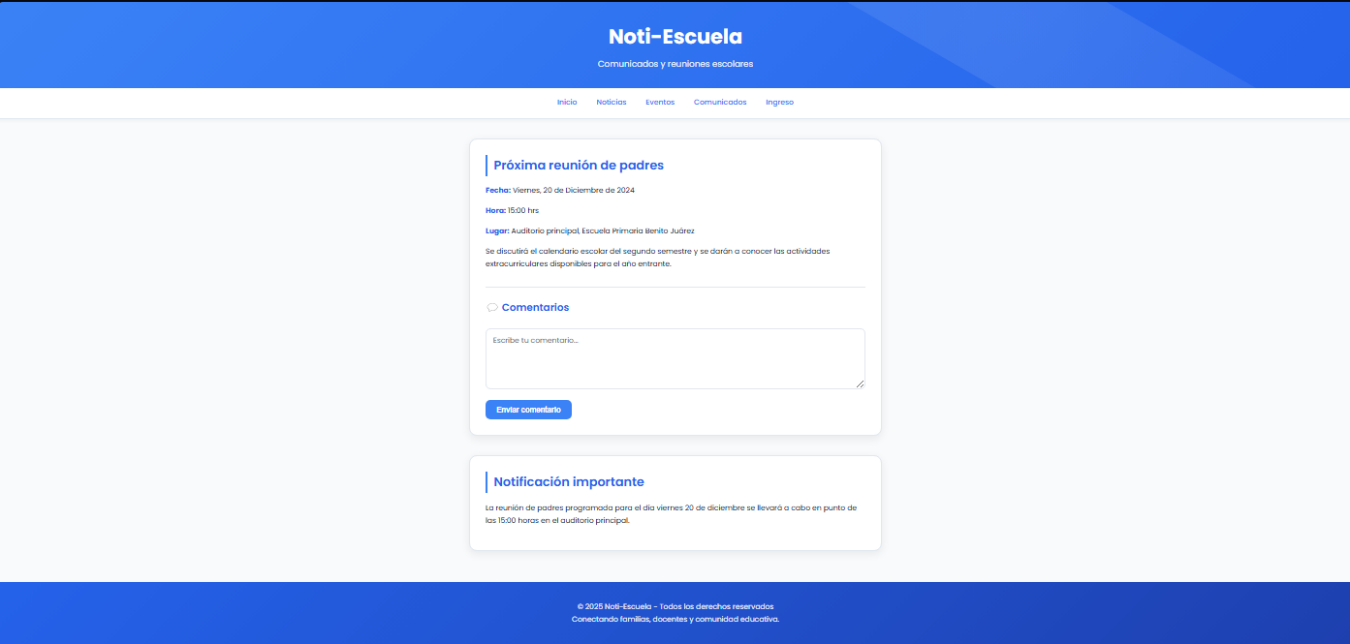
**Templates de la aplicación web:**

Nos enfocamos en un diseño intuitivo, accesible y visualmente sencillo con un toque atractivo, priorizando la experiencia de usuario. La interfaz combina una estética limpia con una jerarquía de contenido clara, utilizando buenas paletas de colores y componentes UI funcionales.

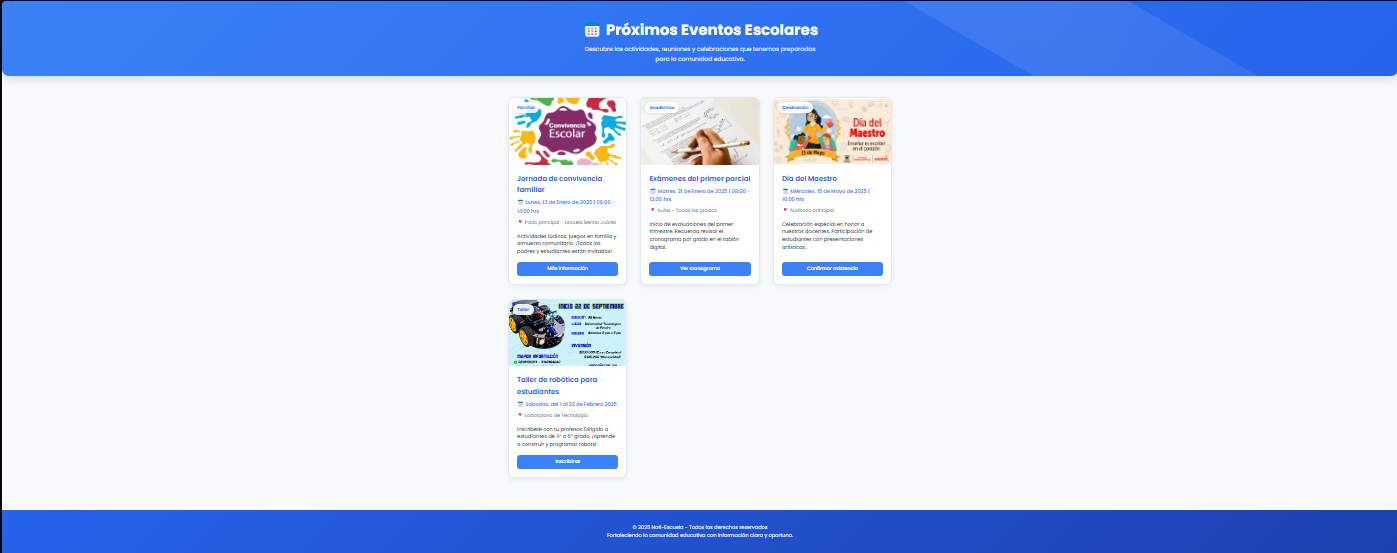
**Página principal (index.html)**

****

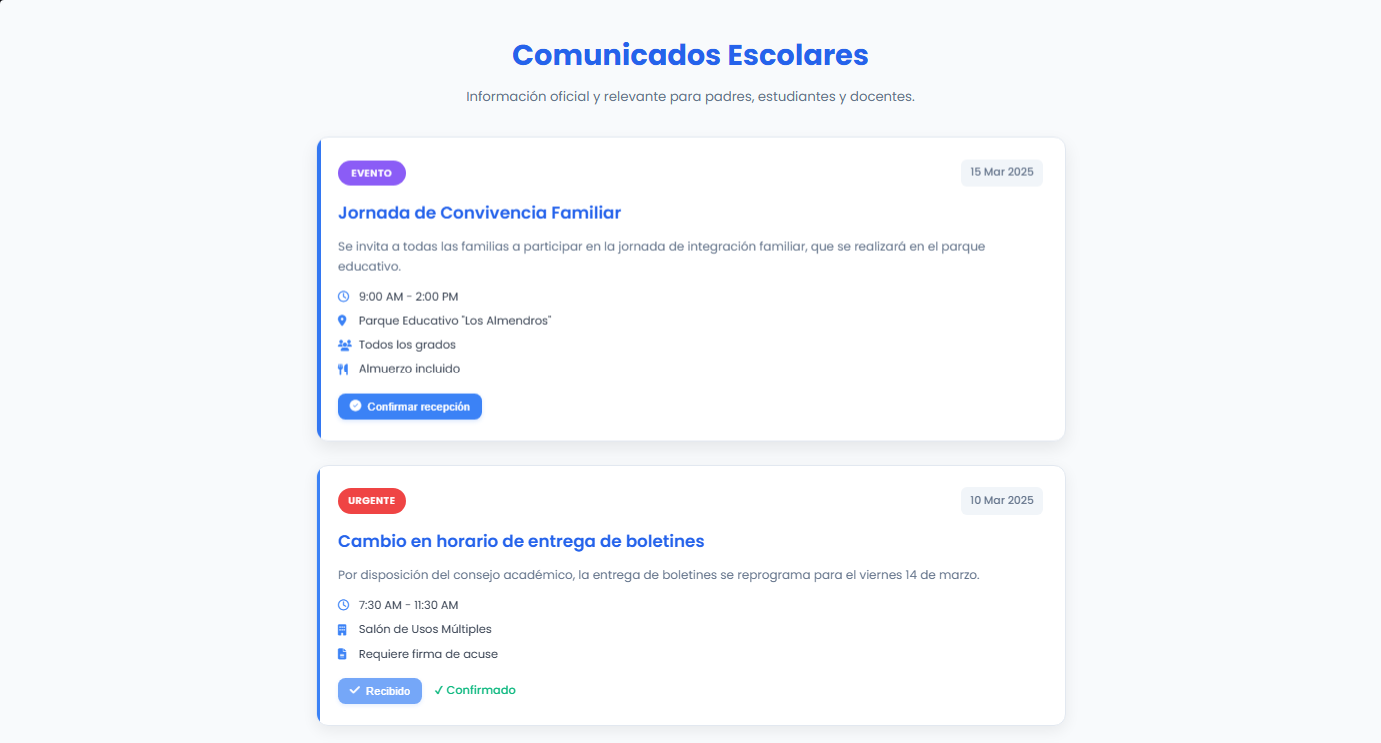


**Noticias (Noticias.html)**

**Eventos (Eventos.html):** Los botones redirigen a distintos lugares, pdfs, formularios dependiendo el evento.

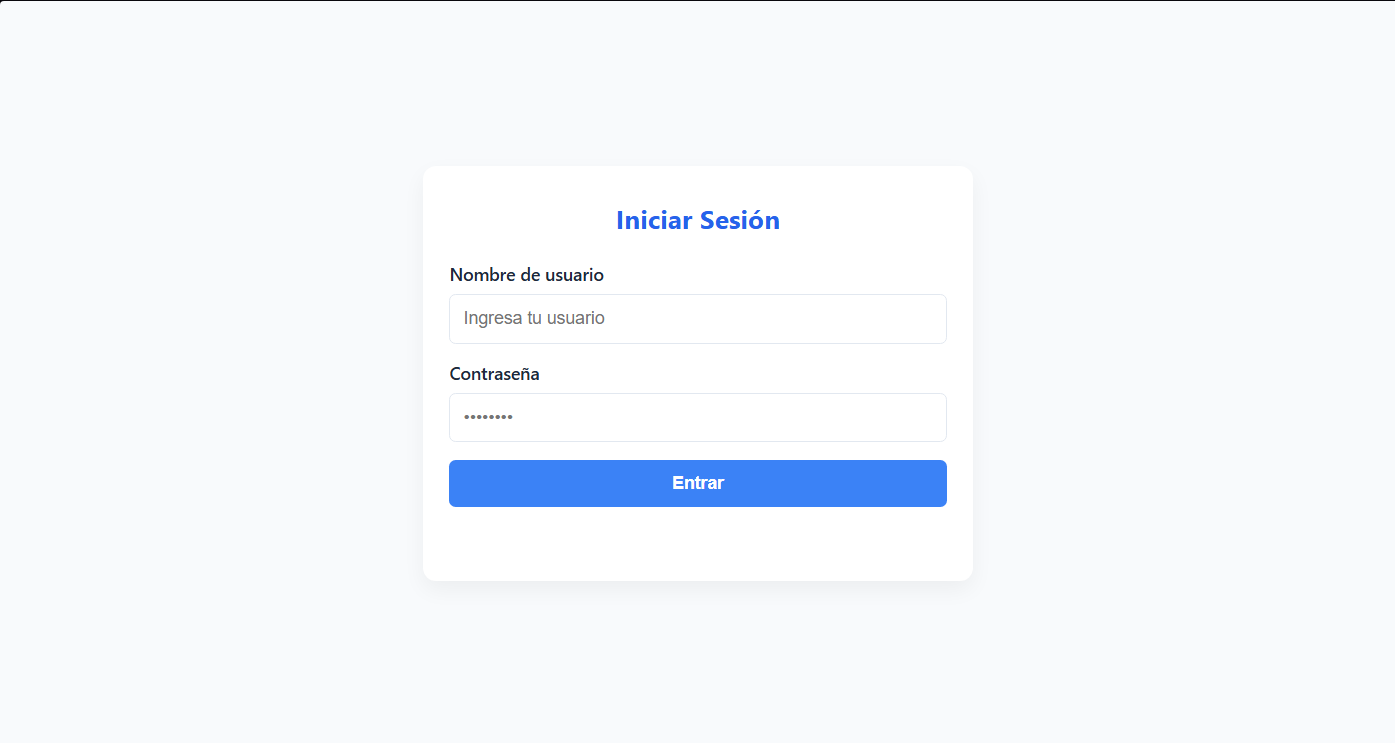


**Comunicados Institucionales (Comunicados.html)**

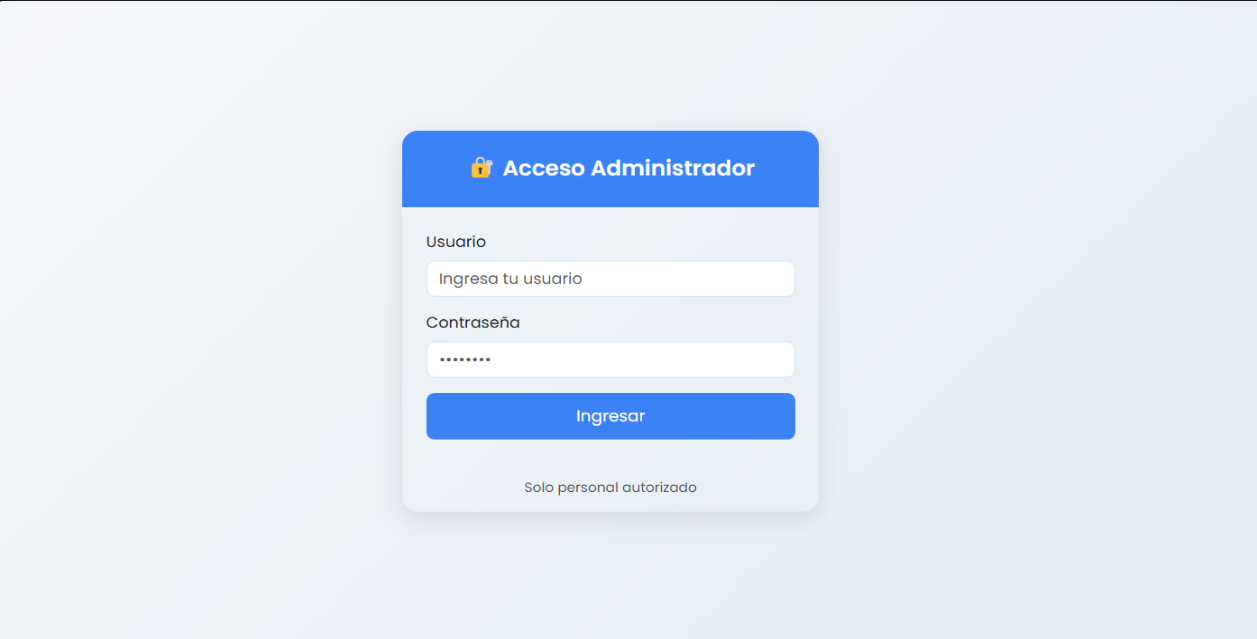
****

Al clickear, confirmas recepción como se ve en la imagen

**Ingreso (Login del padre, estudiante, profesor)**

En este caso, sabemos que se puede mejorar muchísimo este apartado para que se pueda hacer un registro, elegir el rol, etc. En este caso nos centramos con credenciales ya hechas y simuladas.

**Login (Admin)**

****

**Como ejecutar el proyecto:**

Para la ejecución de este proyecto hay dos métodos. Desplegar en render o inicial de manera local, en esta documentación se explicarán los dos métodos:

**Instalación y configuración local:**

**1. Clonar el repositorio:**

git clone https://github/noti-escuela.git

**2. Crear entorno virtual:**

python -m venv venv

source venv/bin/activate # Linux/Mac

venv\Scripts\activate # Windows

**3. Instalar dependencias:**

pip install -r requirements.txt

**4. Configurar variables de entorno:**

Copiar `.env.example` a `.env` y completar las variables.

**5. Iniciar la base de datos (PostgreSQL local o Docker).**

**6. Ejecutar la aplicación:**

flask run

**Actualizar archivos desde render:**

* Para esto primero tenemos que hacer las distintas modificaciones que veamos necesarias en nuestra aplicación web, ya sea estilos en las imágenes, agregar o eliminar cosas de nuestra aplicación Flask. Luego abriremos la consola o terminar del entorno en donde estemos trabajando el proyecto: En mi caso, Visual Studio Code.
* Luego ejecutaremos los siguientes comandos:

git add . 🡪 Para añadir en este caso todo el proyecto

git add (carpeta del archivo/archivo) 🡪 Este caso es para un archivo en especifico

git commit -m “nombre de actualización en repositorio” 🡪 Para capturar el seguimiento de cambios del estado actual y guardarla como confirmación en el repositorio Git

git push origin main “nombre de actualización en repositorio” 🡪 Este es el comando mas importante debido a que envía los cambios que realizamos en local (main) al repositorio (origin)

* Finalmente, solo queda esperar a que render despliegue esos nuevos cambios y se puedan visualizar mediante el link, en este caso https://noti-escuela.onrender.com

**Aspectos relevantes:**

* Fue un proyecto bastante complejo a nivel personal debido a la integración de la base de datos, muchas veces fallaba la conexión del sistema flask con PostgreSQL lo cual llegaba a estresarme, pero de igual manera se trato de hacer lo mejor posible
* En temas de escalabilidad, hay muchas cosas que se pueden integrar como mejora en los login, mejoras visuales en el frontend, integrar mas funcionalidades extras como tableros virtuales u otros.
* En últimos momentos, mi grupo no se notifico para concretar la idea y empezar a codificar (No había comunicación entre nosotros). Mi reacción a esto fue entonces por pasión, ganas de aprender y sacar el proyecto adelante realizarlo solo. Lo cual me dejo en una situación bastante compleja ya que tendría que codificar backend y frontend. Sin embargo, pude realizar bajo mi punto de vista un conjunto de acciones de lo realizado en clase con mi proyecto y estoy muy feliz de lo aprendido en este bootcamp.

**Detalles finales:**

Todo el resto de información como modelos de las tablas de las bases de datos, credenciales y códigos fuentes se encuentra en mi GitHub y en los demás archivos de esta carpeta comprimida. Agradecerle al profesor Rómulo por lo aprendido en este Bootcamp y su pedagogía excepcional la cual me ayudo a entender muchos de los aspectos necesarios para que el proyecto surja de la mejor manera.