Introducción

Este documento detalla el desarrollo del sistema GESDE (Gestor de Excusas Escolares), una aplicación web creada para facilitar la gestión y el control de excusas escolares por parte de padres, profesores y administradores. A lo largo de varias semanas de trabajo, se implementaron funcionalidades clave como autenticación de usuarios, envío y validación de excusas, asignación de profesores por grado, generación de reportes en PDF y notificaciones automáticas mediante SMS. La aplicación fue desarrollada utilizando Flask para el backend, MySQL como sistema gestor de base de datos, y principios de Programación Orientada a Objetos (POO) para una mejor estructura y mantenibilidad del código. Esta documentación resume el proceso de desarrollo semana por semana, destacando los principales avances técnicos y decisiones tomadas.

Tecnologías Implementadas en el Proyecto

Tecnologías base:

- **HTML**
- **CSS**
- **JavaScript**
- **Python**

Herramientas y APIs Externas

- **Twilio (envío de mensajes SMS)**
- **Cohere AI (procesamiento de lenguaje natural)**

Requisitos del Proyecto

A continuación, se detallan las librerías y dependencias necesarias para ejecutar correctamente la aplicación:

a. **Flask==3.1.0**
b. **Flask-MySQL==1.6.0**
c. **flask-mysql-connector==1.1.0**
d. **Flask-MySQLdb==2.0.0**
e. **Flask-SQLAlchemy==3.1.1**
f. **Flask-WTF==1.2.2**
g. **fpdf==1.7.2**
h. **Jinja2==3.1.4**
i. **PyMySQL==1.1.1**
j. **pypng==0.20220715.0**
k. **SQLAlchemy==2.0.37**
l. **twilio==9.4.6**
m. **cohere==5.14.0**

Semana 1

(20–26 de enero de 2025)

1. **Elección de nombre y logo:**
   Busqué un nombre adecuado y me quedé con GESDE. En Canva diseñé el primer logo, ajustando tipografía y colores para que fuera claro y reconocible.

2. **Diseño en Miro:**
   Descargué Miro y, en un tablero, bosquejé la estructura del homepage: cabeceras, secciones de contenido y navegación básica, también seleccione los colores que utilizaría mi página.

3. **Implementación del homepage en HTML:**
   El jueves 23 trasladé ese boceto a código HTML, creando la página de inicio.

4. **Diseño y maquetación del login:**
   También el jueves diseñé la pantalla de login en Miro; el domingo 26, convertí ese diseño a HTML y lo añadí a html.

5. **Preparación del backend:**
   Limpié el repositorio y centralizar todo en PYTHON. Configuré la conexión a MySQL (GESDE en el puerto 3307) y probé un método inicial get_cursor() en Backend/PYTHON/archivos.py.

6. **Restricción de acceso por rol**
   El martes 6 implementé un sistema de control de acceso por roles, asegurándome de que solo los usuarios con rol de profesor pudieran entrar al nuevo apartado. Validé esta funcionalidad tanto en backend como en frontend para evitar accesos no autorizados.

7. **Limpieza de la estructura:**
   El lunes 27 eliminé la carpeta auth para simplificar el repositorio y dejé solo PYTHON como punto central de la lógica de backend.

8. **Inicialización de Flask:**
   Ese mismo día creé mi archivo app.py, instalé y configuré Flask, definiendo la aplicación básica y el routing inicial.

9. **Definición de la tabla user:**
   Creé en la base de datos la tabla user con las columnas:
   - **id (PK, auto-incremental)**
   - **name**
   - **rol**
   - **phone**
   - **idcard**
   - **password**
   - **last_update (timestamp de la última modificación)**

2. **Conexión a la base de datos:**
   El miércoles 29 configuré en Backend/PYTHON/archivos.py la conexión a MySQL, apuntando a la base de datos GESDE en el puerto 3307, y verifiqué que podía abrir un cursor y ejecutar consultas básicas.
3. **Método get_cursor() inicial:**
   Implementé el método get_cursor() en mi clase de conexión, el cual devolvía únicamente el cursor de la base de datos para facilitar las operaciones de consulta.

Semana 2

(27 de enero – 2 de febrero de 2025)

1. **Login funcionar:**
   Implementé el formulario de login en el frontend y conecté la lógica en login.py para autenticar usuarios contra la base de datos, dejando el acceso funcionando correctamente.

2. **Revisión de la tabla excuses:**
   El lunes 31 revisé la estructura de la tabla excuses en la base de datos, confirmando que incluyera los campos necesarios: texto, estado (pending, accepted, rejected), fecha_envio, y id_card del padre.

3. **Creación del formulario de excusas:**
   El martes 1 de abril desarrollé el apartado para que el padre pudiera enviar una excusa desde la plataforma. Diseñé el formulario en HTML y, al mismo tiempo, creé su respectiva clase en form.py, encargada de gestionar y validar los datos enviados. Conecté todo al backend para que al enviarse, los datos se insertaran correctamente en la tabla excuses.

4. **Pruebas con distintos tipos de excusas:**
   El jueves 3 realicé pruebas ingresando diferentes tipos de excusas para verificar que el formulario funcionara bien y que los datos se almacenaran correctamente: textos cortos, largos y con errores comunes.

5. **Verificación en MySQL Workbench:**
   Validé desde MySQL Workbench que las excusas se estuvieran almacenando con su estado en pending por defecto, y que cada una estuviera bien asociada a la cédula (id_card) del padre correspondiente.

Semana 3

(31 de marzo – 6 de abril de 2025)

1. **Diseño del panel de administración:**
   Diseñe un panel de administración para que los administradores pudieran ver todas las excusas en espera, aceptadas o rechazadas. Este panel incluye estado de excusa, todo los detalles de la excusa, cédula del padre y una prueba en imagen o pdf si se ha enviado.

2. **Implementación de la lógica para actualizar el estado de las excusas:**
   Empecé a trabajar en la lógica para cambiar el estado de las excusas (de pending a accepted o rejected). Añadí un sistema para que los administradores pudieran revisar las excusas y actualizarlas según corresponda desde la interfaz de administración.

3. **Pruebas de actualización de estado:**
   Realicé pruebas con el panel de administración, asegurándome de que las excusas pudieran ser modificadas correctamente desde el backend. Probé varios escenarios de cambio de estado y validé que los cambios se reflejaran correctamente en la base de datos.

4. **Refactorización del código de backend:**
   Refactoricé aplicando el código de mi backend a POO para mejorar la estructura y optimizar el manejo de las excusas. Separé la lógica de las excusas en funciones más pequeñas y manejables, lo que hizo que el código fuera más claro y fácil de mantener.

5. **Notificación con Twilio al aceptar o rechazar excusas:**
   En esta semana integraste Twilio a tu proyecto para enviar notificaciones por SMS a los padres cuando una excusa era aceptada o rechazada. Configuraste la lógica en el backend para que, al cambiar el estado de una excusa en la base de datos, automáticamente se disparara un mensaje personalizado informando la decisión al número de teléfono del padre asociado.

6. **Revisión final y validación:**
   Revisé todo el flujo del sistema, desde el envío de las excusas hasta la actualización de su estado. Hice algunas correcciones menores y aseguré que todo estuviera funcionando correctamente antes de seguir con las siguientes funcionalidades del proyecto.

Semana 4

(7 – 14 de abril de 2025)

1. **Inicio del CRUD de usuarios:**
   Comencé a trabajar en el CRUD de usuarios, implementando las funcionalidades para agregar, editar, eliminar y listar usuarios. Creé las rutas necesarias en Flask y los formularios correspondientes en HTML para permitir a los administradores gestionar la información de los usuarios.

2. **Diseño del formulario de registro de usuario:**
   Diseñé el formulario de registro de usuario, donde los administradores pueden ingresar los datos básicos de un nuevo usuario (nombre, rol, teléfono, cédula, contraseña). Implementé la lógica en Flask para validar y almacenar esos datos en la base de datos.

3. **Desarrollo de la funcionalidad de edición de usuario:**
   Empecé a trabajar en la funcionalidad de edición de usuarios. Implementé un formulario para que los administradores pudieran editar los datos de los usuarios (como nombre, teléfono, rol y contraseña). Aseguré que los cambios se reflejaran correctamente en la base de datos.

4. **Implementación de la visualización de usuarios:**
   Creé una vista para mostrar la lista de usuarios registrados en el sistema. Esta vista incluía opciones para editar o eliminar a cada usuario, y se conectó con la base de datos para mostrar los registros actualizados.

5. **Reorganización del proyecto y estructura modular**
   Antes de comenzar con los reportes, me dediqué a mejorar la organización de mis archivos Python. Reduje la cantidad de archivos .py dividiendo el proyecto en módulos bien definidos. Dejé solamente cinco archivos principales: user.py, crud.py, administration.py, teacher.py, y el archivo principal app.py. Esta estructura modular facilitó la mantenibilidad y legibilidad del proyecto.

6. **Refactorización a POO:**
   Comencé a refactorizar el código utilizando Programación Orientada a Objetos (POO). Separé la lógica relacionada con los usuarios en clases, haciendo que el código fuera más organizado y fácil de mantener. También encapsulé la funcionalidad de acceso a la base de datos en métodos de clases, mejorando la legibilidad del proyecto.

7. **Pruebas y validación del CRUD de usuarios:**
   Realicé pruebas para asegurarme de que el CRUD de usuarios estuviera funcionando correctamente: verifiqué que los usuarios pudieran ser agregados, editados, eliminados y listados sin problemas. También validé que la información se guardara correctamente en la base de datos y que el sistema manejara los errores de forma adecuada.

Semana 5

(14 – 21 de abril de 2025)

1. **Revisión de la seguridad del sistema:**
   El miércoles revisé la seguridad del sistema, asegurándome de que las contraseñas se almacenaran de manera segura utilizando hashing (con una librería como werkzeug.security en Flask). Realicé pruebas para verificar que los usuarios solo pudieran acceder a sus propios datos.

2. **Pruebas de edición y carga de foto de perfil:**
   El jueves realicé pruebas exhaustivas con la funcionalidad de edición de usuario, incluyendo la actualización de todos los campos, la carga de la foto de perfil y la validación de que los cambios se guardaran correctamente en la base de datos.

3. **Optimización de la interfaz de usuario:**
   El sábado trabajé en la optimización de la interfaz para la edición de usuarios, mejorando la estética y la usabilidad del formulario, y asegurándome de que fuera fácil para los administradores modificar los datos y las fotos de perfil.

4. **Pruebas finales y validación:**
   El domingo realicé pruebas finales para verificar que el sistema de edición de usuario estuviera funcionando correctamente, incluyendo el registro de fotos de perfil y el manejo adecuado de contraseñas. Aseguré que todo estuviera listo para la siguiente etapa del proyecto.

5. **Creación del apartado para profesores:**
   El lunes desarrollé una nueva sección dedicada exclusivamente a los profesores dentro del sistema GESDE. En esta vista, los docentes pueden consultar las excusas aprobadas enviadas por los padres. Filtré las excusas desde la base de datos utilizando el campo estado = 'aceptada' y mostré únicamente la información relevante para ellos.

1. **Implementación del asistente virtual con Cohere AI:**
   Esta semana comencé la integración de Cohere AI en GESDE, pero no para analizar excusas, sino como asistente virtual conversacional. La idea fue crear un chatbot que ayudara al usuario, especialmente a los padres, a llenar el formulario de excusas con mayor facilidad.
2. **Conexión con la API de Cohere y pruebas iniciales**
   Configuré correctamente la API de Cohere en el backend y realicé pruebas de interacción para asegurarme de que respondiera de forma coherente y útil. Usé ejemplos básicos para validar su comportamiento como asistente.
3. **Diseño del flujo conversacional:**
   Definí una estructura de conversación donde el asistente guía al usuario con preguntas como: "¿Cuál es el motivo de la excusa?", "¿Qué fecha corresponde?" o "¿Quieres que te sugiera una redacción?". Aunque aún no se autocompletan textos, el asistente ayuda con orientación.
4. **Integración del asistente en el frontend:**
   Agregué una interfaz básica donde los usuarios pueden chatear con el asistente. Esto se integró en la sección del formulario de excusas, pero como una ayuda interactiva opcional.
5. **Postergación del análisis y autocompletado de excusas:**
   Aunque originalmente consideré implementar funciones como análisis de lenguaje o autocompletado inteligente, decidí no incluirlas por el momento debido a su complejidad. El enfoque de esta semana fue únicamente el asistente conversacional.

Semana 6

(21 de abril – 28 de abril de 2025)

6. **Generación de reportes PDF de usuarios:**
   El lunes inicié la implementación del módulo de reportes PDF, empezando con un reporte general que listaba todos los usuarios. Utilicé la librería reportlab para generar el PDF a partir de los datos extraídos de la base de datos.

7. **Reporte de usuarios deshabilitados:**
   El martes desarrollé un segundo tipo de reporte que muestra únicamente los usuarios deshabilitados. Configuré los filtros en la consulta SQL y diseñé el PDF para presentar la información de manera clara.

8. **Reporte de trabajadores (no padres):**
   El miércoles implementé otro reporte enfocado en trabajadores, es decir, usuarios con roles distintos al de padre. Esto permitió generar un listado más específico y útil para fines administrativos.

9. **Refactorización del sistema de reportes:**
   El jueves de mayo refactoricé toda la lógica de generación de PDFs para evitar duplicación. Convertí las partes comunes en funciones reutilizables y mantuve todo organizado aplicando principios de POO dentro de un módulo dedicado a reportes.

10. **Pruebas funcionales de reportes:**
    El sábado probé la descarga y visualización de cada uno de los reportes, verificando tanto los datos como el formato de presentación. Me aseguré de que los botones en la interfaz funcionaran correctamente.

11. **Revisión y validación final:**
    El domingo hice una validación general de todo el módulo de reportes, confirmando que cada tipo respondiera correctamente a los filtros y que los documentos generados tuvieran un diseño limpio y profesional. Dejé todo listo para seguir avanzando con nuevas funcionalidades.

12. **Implementación de la carga de foto de perfil:**
    Añadí la opción para que los usuarios pudieran cargar una foto de perfil al editar su información. Implementé un campo de archivo en el formulario de edición y configuré el backend para guardar la imagen correctamente en el servidor y asociarla con el usuario correspondiente.

13. **Mejoras al asistente virtual con Cohere AI**
    Esta semana continué mejorando el asistente virtual que había integrado la semana anterior usando Cohere AI. Refiné las preguntas que hace el chatbot para que fueran más útiles, claras y enfocadas en asistir al usuario al momento de llenar el formulario de excusas.

14. **Ajustes en la interfaz del chat**
    Rediseñé la interfaz del asistente para hacerla más intuitiva. Agregué estilos para que el chat pareciera más natural, con burbujas de mensajes, scroll automático y mejor separación entre respuestas del bot y entradas del usuario.

15. **Visualización de excusas por grado para profesores:**

    Implementé una funcionalidad que permitió a los profesores ver las excusas aprobadas de los estudiantes de su grado asignado. Esto organizó las excusas por grado, asegurando que los profesores solo pudieran acceder a las relevantes para su sección.

Semana 7

(29 de abril – 5 de mayo de 2025)

1. **CRUD de Estudiantes:**
   Implementé una interfaz para registrar, editar y eliminar estudiantes. Cada estudiante quedó vinculado a un grado específico, permitiendo mantener organizada la base de datos académica y facilitar la gestión por parte de la dirección.

2. **CRUD de Profesores:**
   Desarrollé el módulo para gestionar a los profesores, permitiendo agregar nuevos, actualizar su información o eliminarlos. Esta sección fue clave para asociarlos correctamente a los grados correspondientes.

3. **CRUD de Grados:**
   Agregué la funcionalidad para crear, modificar y borrar grados escolares. Esta sección sirvió de base para asignar tanto estudiantes como profesores a sus respectivas secciones.

4. **Asignación de Profesores a Grados:**
   Configuré una funcionalidad que permite asociar un profesor con uno o varios grados. Esta relación facilitó que más adelante, al iniciar sesión, cada profesor pudiera ver solo las excusas de los estudiantes de sus grados asignados.

5. **Reporte de estudiantes:**
   Durante esta semana, desarrollaste la funcionalidad para generar un reporte PDF de todos los estudiantes registrados. Este informe incluyó datos relevantes como nombre, cédula, grado y teléfono, organizado de forma clara y exportable para uso administrativo.

6. **Reporte de excusas (aprobadas, pendientes y rechazadas):**
   También implementaste la generación de reportes PDF separados para excusas según su estado. Esto permitió filtrar y exportar únicamente las excusas aceptadas, rechazadas o en espera, facilitando la revisión y documentación por parte del personal académico.

7. **Diseño e integración de la interfaz de profesor:**
   El miércoles diseñé una interfaz clara y sencilla para los profesores, mostrando una tabla con las excusas aprobadas, incluyendo campos

como el nombre del estudiante, la fecha de la excusa y el motivo. Integré esta vista con la navegación general de la plataforma.

8. **Pruebas de funcionalidad y visualización:**
   El viernesrealicé pruebas para asegurarme de que solo los profesores accedieran a la vista y de que las excusas se mostraran correctamente. También verifiqué que los filtros por estado funcionaran bien en la consulta SQL y que la interfaz se mantuviera limpia y funcional.
9. **Revisión final del módulo de profesores:**
   El domingo finalicé revisando el flujo completo de este módulo, desde el login del profesor hasta la consulta de excusas. Me aseguré de que los permisos, la visualización y la lógica de filtrado estuvieran correctamente implementados y listos para producción.