UNIVERSIDAD CATÓLICA DE TEMUCO

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA
FACULTAD DE INGENIERÍA

ClaveForte

Curso: Desarrollo de Aplicaciones Empresariales

Integrantes: Ignacio De Celis y Oscar Barra

Fecha: 01/06/2025

Proyecto y Objetivo

Proyecto: Gestión de Contraseñas Empresariales

Este proyecto tiene como objetivo desarrollar una plataforma web segura para el almacenamiento y gestión de credenciales empresariales. Surge como respuesta a una problemática común en las organizaciones: **la falta de mecanismos seguros y centralizados para gestionar contraseñas**, lo que puede derivar en filtraciones, accesos no autorizados y dificultades para compartir credenciales de forma controlada.

Para abordar este problema, el proyecto permitirá que cada usuario acceda exclusivamente a sus propias credenciales, mientras que un Jefe o Administrador Principal pueda visualizar y administrar todas las contraseñas del sistema.

Además, el sistema permite que los usuarios compartan credenciales de manera **controlada** y **voluntaria**, mediante el uso de una **contraseña adicional personal**, la cual se requiere para autorizar el acceso temporal o específico a determinada información. Esta funcionalidad garantiza que el intercambio de datos sensibles sólo se produce bajo consentimiento explícito y con medidas de seguridad adicionales.

La plataforma será construida utilizando **Flask** para la administración de backend, **SQLite** como un sistema de base de datos liviano y **Tailwind CSS** para construir una interfaz web moderna, limpia y responsiva.

Diagrama Entidad-Relación (Imagen en la sección de Diagramas)

A continuación se presenta el **diagrama entidad-relación** que representa la estructura lógica de la base de datos del sistema. Este diagrama permite visualizar cómo se organizan y relacionan las distintas entidades involucradas en la gestión de credenciales.

Tabla Users:

Esta tabla almacena la **información principal de cada usuario**, incluyendo credenciales de acceso y su rol dentro del sistema.

- **id_usr (PK)[Int]:** Identificador único de cada usuario.
- **usr_name[Str]:** Nombre del usuario. Puede ser usado para mostrar en interfaces o perfiles.
- **usr_mail[Str]:** Correo electrónico único, usado como identificador para inicio de sesión.

- **usr_pass[Str]:** Contraseña principal hasheada. Se utiliza exclusivamente para la autenticación.
- **secret_pass[Str]:** Segunda contraseña hasheada, usada para acceder a información o acciones sensibles dentro del sistema (por ejemplo: ver claves, compartir datos, etc).
- **last_login[Date]:** Fecha y hora del último inicio de sesión exitoso.
- **id_rol(FK)[Int]:** Relaciona al usuario con su rol en la tabla Roles. Determina el tipo de permisos o acceso que tiene.

Tabla Roles:

Esta tabla define los **tipos de roles** disponibles dentro del sistema, como administrador, usuario, invitado, etc.

- **id_rol(PK)[Int]:** Identificador único de cada rol.
- **rol_type[Str]:** Descripción del rol (por ejemplo: 'admin', 'usuario', 'invitado', etc.). Esto permite personalizar los niveles de acceso.

Tabla Accesos:

Registra cuándo y por qué un usuario accede a una credencial guardada. Útil para auditoría y seguridad.

- **id_acceso(PK)[Int]:** Identificador del acceso.
- **id_usr(FK)[Int]:** Usuario que accedió a la credencial.
- **id_credential(FK)[Int]:** Credencial a la que se accedió.
- **timestamp[Date]:** Fecha y hora del acceso.

Table Credenciales:

Almacena las contraseñas que el usuario desea guardar, asociadas a distintos servicios

- **id_credential(PK)[Int]:** Identificador de la credencial.
- **service_name[Str]:** Nombre del servicio (ej. 'Netflix', 'Outlook').
- **service_pass[Str]:** Contraseña del servicio cifrada o hasheada.
- **access_level[Str]:** Nivel de acceso (ej: 'privado', 'compartido').
- **id_usr(FK)[Int]:** Propietario de la credencial.

Relación entre las tablas:

La relación de Usuarios ---> Roles es **muchos a uno**: Un rol puede estar asignado a muchos usuarios, pero cada usuario solo puede tener un rol.

La relación de Credenciales ---> Usuarios es **muchos a uno**: Un usuario puede guardar múltiples contraseñas, pero cada contraseña tiene un único dueño original.

La relación de Accesos ---> Usuarios es **muchos a uno**: Cada vez que alguien consulta o accede a una credencial, se registra quién lo hizo y cuándo.

La relación de Accesos ---> Credenciales es **muchos a uno**: Porque varios accesos pueden revisar la misma credencial en distintos periodos de tiempo.

Diagrama Casos de Uso (Imagen en la sección de Diagramas)

A continuación se muestra el **diagrama de casos de uso** del sistema, el cual describe las principales interacciones entre los usuarios y la plataforma.

Actor Usuario:

1. **Registrarse**: El usuario crea una cuenta proporcionando su nombre, correo, contraseña principal y contraseña secreta.

2. **Iniciar sesión**: El usuario se autentica en el sistema con su correo y contraseña principal.
    2.1. **Include** Validar datos del usuario: Asegura que el correo y la contraseña correspondan con un usuario registrado.

3. **Ver perfil**: El usuario consulta su información personal registrada (nombre, correo, rol)
    3.1. **Extend** Actualizar perfil: Permite modificar ciertos campos del perfil.
        3.1.1. **Extend:** Cambiar correo: El usuario puede modificar su dirección de correo electrónico.
        3.1.2. **Extend:** Cambiar nombre de usuario El usuario actualiza su nombre o alias visible.
        3.1.3. **Extend:** Cambiar contraseña de autenticación Cambia la contraseña principal del usuario.

4. **Guardar credencial**: Permite registrar en el sistema una nueva credencial de un servicio externo (nombre del servicio, usuario y contraseña cifrada).

5. **Eliminar credencial**: Elimina una credencial previamente almacenada por el usuario.

6. **Ver credencial**: El usuario accede a los detalles de una credencial guardada.
    6.1. **Extend:** Acceder a credencial compartida Permite ver una credencial que otro usuario haya compartido con él.
    6.2. **Extend:** Compartir credencial Brinda acceso a una credencial propia a otro usuario.
        6.2.1. **Include:** Validar envío con la contraseña secreta Para autorizar el acto de compartir, el sistema requiere la validación con la contraseña secundaria (secret_pass).

Actor Administrador:

1. **Administrar usuarios**: El administrador puede ver, modificar o eliminar usuarios existentes. También puede restablecer contraseñas o cambiar roles si es necesario.

2. **Administrar roles**: Permite crear, editar o eliminar tipos de rol (admin, usuario, etc.) y asociarlos a usuarios según el nivel de acceso.

Arquitectura del Sistema:

El sistema sigue una arquitectura de tipo **cliente-servidor**, dividida en tres componentes principales: el **cliente (frontend)**, el **servidor (backend con Flask)** y la **base de datos (SQLite)**. A esto se suma una capa de **seguridad** encargada de garantizar la protección de los datos sensibles y el control de acceso.

1. Cliente (Frontend)

El cliente consiste en una interfaz web construida con **HTML** y **Tailwind CSS**, orientada a ser moderna, minimalista y responsiva. Desde esta interfaz, los usuarios pueden interactuar con el sistema mediante diversos formularios y acciones clave.

Características:

- Formulario de inicio de sesión y registro.
- Interfaz para la gestión de credenciales personales.
- Solicitudes de acceso a credenciales compartidas.
- Comunicación con el backend mediante peticiones HTTP (fetch/AJAX)

2. Servidor (Backend con Flask)

El backend está implementado utilizando **Flask**, un microframework de Python, encargado de manejar la lógica de negocio, el control de flujo y la validación de usuarios y permisos.

Responsabilidades principales:

- Manejo de rutas para operaciones como login, registro, gestión de credenciales y solicitudes de acceso.
- Cifrado y descifrado de contraseñas utilizando bibliotecas como bcrypt o pynacl.

3. Base de Datos (SQLite)

Se utiliza **SQLite** como sistema de gestión de base de datos por su ligereza y simplicidad en entornos locales o de pequeña escala. El diseño de la base de datos permite mantener relaciones entre usuarios, credenciales y accesos con control detallado de permisos.

Tablas principales:

- Usuarios (ID, nombre, rol, contraseña cifrada)
- Credenciales (ID, nombre del servicio, contraseña cifrada, nivel de acceso)
- Accesos (ID, ID usuario, ID credencial, timestamp, motivo)

4. Seguridad

La seguridad es un aspecto central del sistema, que se aborda desde distintas capas de la aplicación.

Medidas implementadas:

- Autenticación mediante sesiones gestionadas con **Flask-Login** o tokens **JWT**.
- Cifrado robusto de contraseñas y credenciales sensibles antes de almacenarlas.
- Validación estricta de permisos antes de autorizar la visualización o edición de credenciales.
- Registro de actividad (logs de accesos).

Consideraciones Técnicas o Desafíos Anticipados

Durante el desarrollo e implementación del sistema, se identifican una serie de aspectos técnicos críticos y desafíos que deben abordarse cuidadosamente para garantizar la seguridad, funcionalidad y escalabilidad de la plataforma.

1. Almacenamiento Seguro de Contraseñas

Uno de los retos fundamentales es proteger la confidencialidad de las credenciales almacenadas. Para ello se aplicarán los siguientes principios:

- Cifrado fuerte para las contraseñas de servicios guardadas por los usuarios.
- Hashing seguro para las contraseñas de acceso de los usuarios.
- Prohibición absoluta del almacenamiento de contraseñas en texto plano, incluso durante procesos temporales como validaciones.

2. Compartición Segura de Credenciales

Una funcionalidad clave del sistema es permitir que los usuarios compartan credenciales de forma controlada. Esto implica:

- La implementación de una **clave personal secundaria** o **contraseña temporal** para autorizar cada acto de compartición.
- Posible expiración del acceso compartido, o límites por tiempo o intentos.

3. Gestión de Sesiones y Protección contra Amenazas Comunes

Será fundamental implementar mecanismos de defensa contra ataques web comunes, incluyendo:

- **CSRF (Cross-Site Request Forgery)**: mediante tokens anti-CSRF.
- **XSS (Cross-Site Scripting)**: sanitizando entradas y controlando los contenidos dinámicos.
- **Secuestro de sesión**: mediante tokens seguros, expiración automática, e invalidación activa.

El uso de herramientas como **Flask-Login** o **JWT** será clave para el manejo de sesiones seguras.

4. Usabilidad y Experiencia de Usuario (UX)

Para garantizar la adopción y el uso efectivo del sistema, se busca una interfaz clara y accesible, incluso para usuarios sin conocimientos técnicos avanzados. Esto incluye:

- Formularios intuitivos y con validaciones en tiempo real.
- Mensajes de error y confirmación claros y amigables.
- Diseño responsivo y limpio utilizando **Tailwind CSS**.

5. Mantenimiento y Auditoría de Registros

Será necesario implementar mecanismos de registro de actividad para:

- Rastrear accesos a credenciales, especialmente cuando son compartidas.
- Detectar comportamientos inusuales o posibles brechas de seguridad.

Plan de trabajo y distribución de tareas

El desarrollo del proyecto se realizará en equipo, siguiendo una planificación semanal que permita avanzar de manera progresiva desde un prototipo básico hasta una versión funcional del sistema.

Semana 1 – Desarrollo del Prototipo Mínimo Navegable (PMN)

Objetivo: Diseñar y construir una interfaz funcional que permita la navegación básica por la aplicación.

Oscar Barra:

- Diseñar la estructura de la interfaz principal (home, login, registro) utilizando Tailwind CSS.
- Implementar el enrutamiento básico de las vistas en el frontend (HTML o usando Flask templates).

Ignacio De Celis:

- Configurar el servidor Flask con las rutas base (/login, /signup, /dashboard).
- Crear una base de datos inicial en SQLite con las tablas Usuarios y Credenciales (sin lógica compleja aún).

Semana 2 – Desarrollo del Prototipo Mínimo Viable (PMV)

Objetivo: Incorporar funcionalidades esenciales y establecer comunicación entre frontend y backend.

Oscar Barra:

- Implementar formularios funcionales de registro e inicio de sesión conectados al backend.
- Diseñar la vista de gestión de credenciales (listar, crear, eliminar).

Ignacio De Celis:

- Programar la lógica de autenticación en Flask (hash de contraseña, sesiones o JWT).
- Conectar la base de datos con la lógica de creación y consulta de credenciales.

Semana 3 – Desarrollo general y mejoras del sistema

Objetivo: Consolidar las funcionalidades, aplicar medidas de seguridad y mejorar la experiencia de usuario.

Oscar Barra:

- Mejorar la interfaz (feedback visual, validaciones de formularios, navegación clara).
- Implementar la funcionalidad de solicitudes de acceso a credenciales compartidas.

Ignacio De Celis:

- Añadir cifrado real de contraseñas.
- Implementar registros de acceso y validación de permisos al acceder a datos.

Diagramas del Sistema

Diagrama Entidad Relación

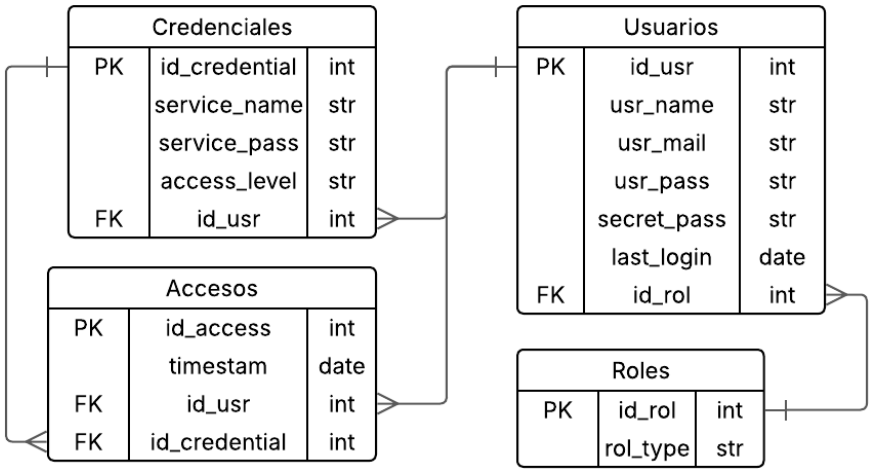


Diagrama Casos de Uso

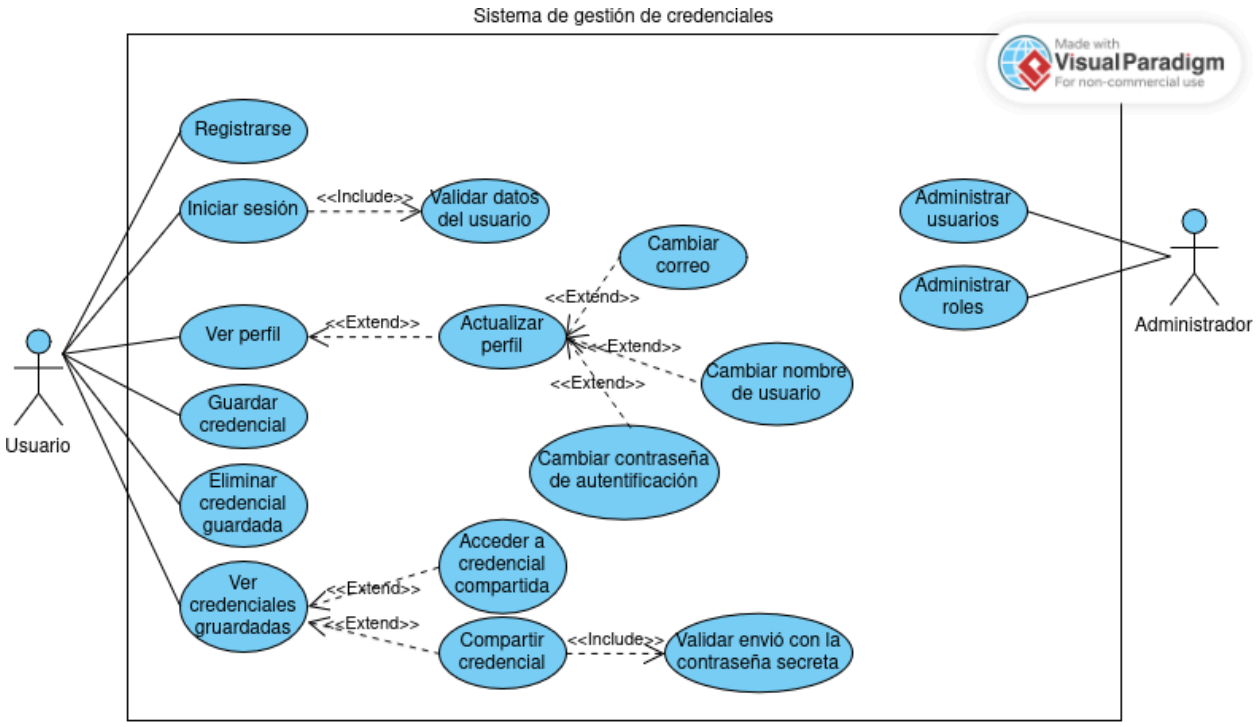
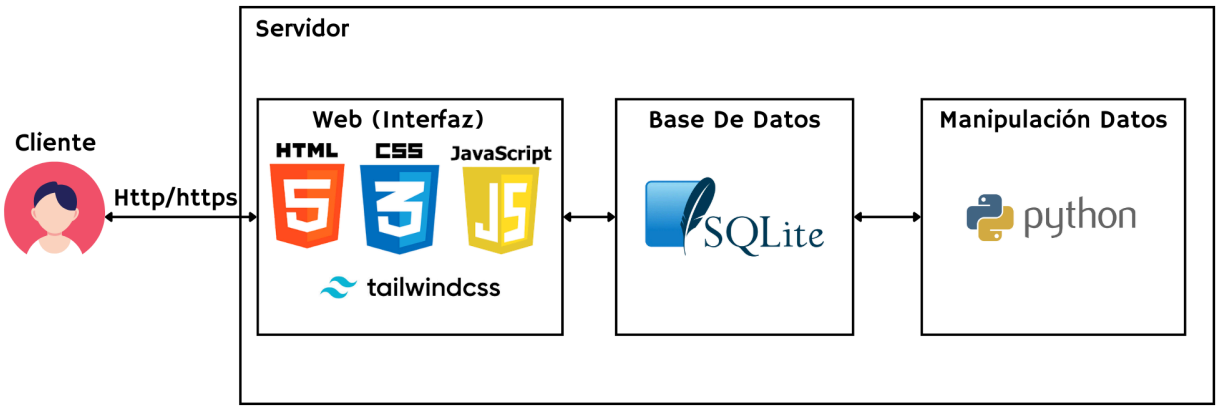


Diagrama Esquemático



Enlaces de Interés

Github: https://github.com/oscarbarra/DAE_Project2.git