Trabajo Obligatorio segundo semestre 2024 - BD I

Introducción

El presente informe describe el desarrollo del proyecto de gestión de deportes de nieve para la universidad. Este proyecto tiene como objetivo principal la administración de las actividades relacionadas con deportes de nieve. Utilizando SQL Server, Python, React y Flask, logramos crear un sistema de gestión que incluye funcionalidades como alta, baja y modificación de instructores, gestión de turnos, modificación de actividades, administración de alumnos, así como un sistema de registro e inicio de sesión a nivel de administrador.

Desarrollo del proyecto

El proceso de desarrollo comenzó con la creación de una base de datos relacional en SQL Server, elegida por su facilidad de uso, ya que habíamos trabajado previamente con ella en el curso, además de su alta compatibilidad con Python. En la base de datos se encuentran las tablas predeterminadas para administrar los distintos aspectos del sistema, como actividades, alumnos, clases, instructores, turnos y equipamiento. Estas tablas fueron diseñadas para reflejar las necesidades específicas del proyecto, pero durante el desarrollo realizamos ajustes en el modelo de datos para mejorar su funcionalidad y optimizar la forma en que se almacenaba la información. Se modificaron ciertos tipos de datos y relaciones para facilitar las consultas y reducir redundancias.

Una de las decisiones clave en el diseño de la base de datos fue el uso del atributo IDENTITY en tablas como actividades, login, turnos y clase. Esto permitió generar identificadores únicos de manera automática, asegurando la integridad de las relaciones entre tablas y facilitando la gestión de claves primarias y externas. Además, este enfoque optimiza el rendimiento de las inserciones y simplifica el manejo de datos al evitar colisiones de claves manuales.

Para interactuar con esta base de datos SQL, se utilizó la librería **pyodbc** en Python, la cual permitió establecer la conexión y realizar todas las operaciones necesarias, incluyendo consultas (SELECT), inserciones (INSERT), eliminaciones (DELETE) y actualizaciones (UPDATE). Inicialmente, los resultados de las consultas eran visualizados mediante impresiones en la consola. Sin embargo, fue necesario adaptar estas salidas a objetos estructurados para una futura implementación de interfaz web.

Una parte fundamental del desarrollo fue la implementación de un sistema de registro e inicio de sesión seguro. Para ello, las contraseñas de los usuarios se almacenaron cifradas utilizando la librería **bcrypt**. Esto asegura que las contraseñas no se guarden en texto plano, sino como hashes cifrados, lo que protege los datos sensibles incluso si la base de datos es comprometida mediante inyección SQL.

Una vez configurada la base de datos y las operaciones necesarias en cada tabla para la administración de todos los aspectos del proyecto con Python, utilizamos Flask para crear los endpoints que conectaran nuestra base de datos con la interfaz web. Flask fue elegido por su simplicidad y flexibilidad para crear APIs, lo que permitió una integración eficaz entre el backend y el frontend.

Intentamos implementar un sistema de caché para mejorar el tiempo de respuesta en las solicitudes que consultan frecuentemente la base de datos, utilizando la librería **Flask-Caching**. Este sistema permitiría almacenar en memoria las respuestas de operaciones recurrentes, como el inicio de sesión, evitando realizar consultas repetitivas a la base de datos. Sin embargo, no logramos obtener el resultado esperado debido a problemas en la configuración y la integración con el resto del proyecto, pero a lo mejor con un poco más de tiempo lo hubiéramos conseguido.

En cuanto al frontend, seleccionamos React debido a sus beneficios para crear interfaces de usuario dinámicas y reactivas, lo que garantiza una experiencia intuitiva y ágil para el usuario. React, combinado con los endpoints creados en Flask, permitió la visualización y el manejo administrativo de los datos en tiempo real, asegurando que el sistema fuera funcional y fácil de usar.

Justificación de las decisiones de diseño

Las herramientas y tecnologías utilizadas en el desarrollo de este proyecto fueron seleccionadas en función de su facilidad de uso, compatibilidad y adaptabilidad a los requerimientos del sistema.

- **SQL Server** fue elegido por nuestra familiaridad con este motor de base de datos, ya que lo habíamos utilizado previamente en el curso. Es una base de datos estructurada con capacidades avanzadas que nos permitió gestionar relaciones entre varias tablas y facilitar la organización y el acceso eficiente a la información. Una de sus principales ventajas es la confiabilidad en las consultas y la rapidez con la que estas se ejecutan. Además, su alta compatibilidad con Python fue un factor clave en nuestra elección, ya que nos permitió integrar la base de datos de manera eficiente con el lenguaje de programación que utilizamos. SQL Server también cuenta con un robusto conjunto de herramientas y una amplia documentación, lo que facilitó significativamente su implementación en el proyecto. Otro aspecto que consideramos fue su capacidad para aplicar los principios de Consistencia, Disponibilidad y Confiabilidad, fundamentales en una base de datos relacional. Asimismo, la implementación de propiedades como la Atomicidad (ACID) juega un papel crucial en garantizar la integridad y seguridad de los datos. La suma de todos estos factores refuerza aún más nuestra decisión de optar por una base de datos relacional como SQL Server.
- **Python y Flask** fueron seleccionados para el backend debido a la versatilidad y simplicidad de Python, que permite un desarrollo rápido y eficiente. Flask, por su parte, es un framework minimalista que facilita la creación de APIs y la integración con bases de datos, lo cual fue esencial para garantizar la flexibilidad del proyecto.
- **React** fue elegido para el frontend por sus ventajas en la creación de interfaces dinámicas y de alto rendimiento. React permite una experiencia de usuario fluida y reactividad en tiempo real, lo que fue crucial para las funcionalidades administrativas que necesitábamos implementar.

Cada una de estas tecnologías se eligió no solo por su compatibilidad, sino también por su capacidad para integrar el sistema de manera eficiente y escalable en un futuro hipotético.

QA Manual

A lo largo del desarrollo del proyecto, realizamos un proceso de aseguramiento de calidad (QA) manual para garantizar que todas las funcionalidades del sistema fueran operativas y estuvieran libres de errores. Las tareas de QA manual se enfocaron principalmente en la comprobación de cada funcionalidad a través de la interfaz web, para asegurar que todo el sistema fuera intuitivo y fácil de usar para los administradores.

Entre las actividades de QA realizadas, se incluyeron:

- **Pruebas de validación de formularios:** Verificamos que todos los formularios (como los de alta de instructores, alumnos y actividades) estuvieran correctamente validados, tanto en el frontend como en el backend.
- **Comprobación de funcionalidades CRUD:** Realizamos pruebas de las funcionalidades de crear, leer, actualizar y eliminar (CRUD) en todas las secciones del sistema (actividades, instructores, alumnos, etc.), asegurándonos de que los datos se gestionan correctamente y se reflejan de manera precisa en la base de datos.
- **Revisión de la experiencia de usuario:** Evaluamos la usabilidad de la interfaz, verificando que la navegación fuera fluida y que la interfaz fuera intuitiva para el usuario final en la medida que nos fue posible, adaptado a nuestros conocimientos de desarrollo frontend.
- **Pruebas de rendimiento:** Comprobamos que el sistema funcionara de manera eficiente, incluso al manejar una gran cantidad de datos, para asegurar que no hubiera demoras ni caídas en el rendimiento.

Estas pruebas permitieron detectar y corregir varios problemas menores antes de la implementación final, asegurando así que el sistema fuera confiable y estuviera listo para su uso.

Resultados y Conclusión

Como resultado del proyecto realizado, se obtuvo un sistema de gestión de deportes de nieve que cumple con todos los aspectos necesarios para la administración eficiente de actividades, alumnos, instructores, clases y turnos. Este sistema se presenta a través de una página web amigable y funcional, diseñada para facilitar el uso tanto a nivel de administrador como en las distintas operaciones de gestión. El sistema garantiza una experiencia intuitiva y una integración efectiva entre la base de datos, el backend en Python y la interfaz desarrollada en React y Flask. Esto asegura que todas las acciones administrativas puedan ejecutarse de forma directa y confiable, cumpliendo con los requisitos del proyecto y superando los desafíos encontrados durante su desarrollo.

Bibliografía

- *Documentación oficial de SQL Server.* Microsoft. Recuperado de: https://docs.microsoft.com/es-es/sql/
- *Documentación oficial de Python.* Python Software Foundation. Recuperado de: https://docs.python.org/
- *Flask: Web Development, One Drop at a Time.* Armin Ronacher. Recuperado de: https://flask.palletsprojects.com/
- *React Documentation.* Meta. Recuperado de: https://reactjs.org/docs/getting-started.html
- *PyODBC Documentation.* Microsoft. Recuperado de: https://github.com/mkleehammer/pyodbc