Proyecto Final – Bases de Datos

Sistema de gestión para la Escuela de Deportes de Nieve UCU: "UCU Snow Zone"

Lucía Amor, Manuela Barbachan y Luis Gaudiano

Facultad de Ingeniería y Tecnologías, Universidad Católica del Uruguay

Bases de Datos

Prof. Christian Dinardi

25 de noviembre del 2024

Índice

Abstract:

El presente proyecto tiene como objetivo explicar el desarrollo de un sistema de gestión integral para la Escuela de Deportes de Nieve UCU, enfocado en mejorar la administración de funciones clave, como las inscripciones a cursos, el alquiler de equipamiento y la generación de reportes sobre el éxito de la escuela. Utilizando una base de datos relacional para una gestión estructurada de la información, el sistema integra tecnologías modernas para mejorar la eficiencia operativa y administrativa. Este sistema aborda problemas comunes de la gestión manual, como errores humanos y pérdida de tiempo, ofreciendo una solución digital confiable y optimizada. Con una arquitectura escalable, esta solución busca transformar las operaciones diarias, facilitando la toma de decisiones y garantizando un manejo adecuado de los recursos.

Imtroducción:

El desarrollo de este sistema de gestión para la Escuela de Deportes de Nieve UCU surge de la necesidad de optimizar los procesos administrativos clave. La plataforma facilita la inscripción de alumnos a cursos, el alquiler de equipamiento y la creación de reportes sobre el desempeño de las actividades. Con el fin de asegurar una gestión eficiente y organizada de los datos, se implementa una base de datos relacional en MySQL, conectada a un backend desarrollado en Python con Flask, lo que permite manejar operaciones CRUD de manera efectiva y segura.

El frontend, desarrollado con React, proporciona una interfaz amigable y accesible tanto para los administradores como para los usuarios, permitiendo interactuar de manera dinámica con la plataforma. Además, el sistema está diseñado para mejorar la eficiencia operativa de la escuela, brindando un espacio escalable que puede adaptarse a las necesidades futuras de la institución.

Marco teórico:

El desarrollo de sistemas de gestión modernos requiere la integración de varias tecnologías que se complementen entre sí, garantizando una arquitectura eficiente, escalable y flexible. Este proyecto emplea una combinación de herramientas probadas y de fácil integración, permitiendo la gestión adecuada de datos, la creación de interfaces dinámicas y la interacción segura entre el backend y el frontend.

Las bases de datos relacionales constituyen un pilar fundamental en el desarrollo de sistemas de gestión modernos debido a su capacidad para estructurar y vincular datos de manera eficiente. Este modelo, organiza los datos en tablas interrelacionadas mediante claves primarias y foráneas, permitiendo mantener la integridad y la consistencia de los datos. Cada tabla representa una entidad, mientras que las relaciones entre ellas se establecen mediante restricciones que aseguran la coherencia de los datos a lo largo del sistema.

Una de las características más relevantes de este modelo es el uso de SQL (Structured Query Language), que permite realizar consultas eficientes y manejar el volumen de datos esperado para aplicaciones de esta índole.

Las bases de datos relacionales destacan en entornos donde la información necesita ser precisa, vinculada y actualizada frecuentemente. Por esto, una de las piezas fundamentales de nuestro trabajo es MySQL, un sistema de gestión de bases de datos relacionales ampliamente utilizado por su robustez y eficiencia en el manejo de grandes volúmenes de información. Este modelo relacional es ideal para gestionar la información clave de la Escuela de Deportes de Nieve UCU, como actividades, alumnos, instructores y cursos.

Para interactuar y administrar de manera eficiente la base de datos MySQL, se emplea DataGrip, un IDE especializado en bases de datos que facilita la creación, consulta y mantenimiento de las bases de datos. DataGrip permite la ejecución de consultas SQL, la visualización de datos y la administración de esquemas de forma intuitiva, lo que acelera el proceso de desarrollo y permite una depuración efectiva. Esta herramienta no solo facilita la creación y modificación de las tablas y relaciones, sino que también automatiza tareas repetitivas, como la generación de scripts DDL, mejorando así la coherencia y optimización de la base de datos.

El desarrollo del backend se realiza utilizando Flask, un micro-framework de Python que proporciona una estructura para la construcción de aplicaciones web. Flask facilita la creación de APIs REST que permiten la comunicación eficiente entre el backend y el frontend.

Para la construcción de la interfaz de usuario, se utiliza React, una librería comúnmente utilizada para el desarrollo de aplicaciones web interactivas. React, permite crear una interfaz dinámica, de fácil navegación, que responde a las acciones del usuario. Esto mejora la experiencia de usuario al hacer que la interacción con la aplicación sea más fluida y eficiente.

Desarrollo:

Durante el diseño y diseño de la base de datos, se adoptaron decisiones fundamentales para garantizar la eficacia operativa y la seguridad de los datos. Estas decisiones no solo responden a los requerimientos inmediatos del proyecto, sino que también anticipan el crecimiento y la escalabilidad del sistema a largo plazo. A lo largo del proceso, se consideraron aspectos clave como la integridad de los datos, la optimización de recursos y la coherencia en las operaciones, lo que permitió estructurar una base de datos robusta y eficiente que podría gestionar tanto un volumen alto de usuarios como una amplia variedad de registros.

Se utilizó DataGrip y MySQL para crear la estructura de la base de datos, y uno de los aspectos más importantes fue eliminar redundancias y asegurar la consistencia de los datos almacenados. Con ello, se redujo el riesgo de inconsistencias y se mejoró la flexibilidad del sistema ante futuros cambios o expansiones. Esto lo logramos con la creación de una tabla intermedia alumno_clase, la cual maneja la relación muchos a muchos entre los alumnos y las clases. De este modo, evitamos la redundancia de datos y mejoramos la eficiencia en las consultas. La tabla intermedia facilita la gestión de inscripciones y actualizaciones, optimizando los recursos sin crear registros repetidos.

Otro punto clave fue la implementación de restricciones de integridad tanto a nivel de base de datos como en el backend. Estas restricciones garantizan que se cumplan los requisitos de

negocio y que las operaciones sobre los datos sean seguras y consistentes. Se buscó para validar los correos electrónicos de los usuarios, asegurando que solo aquellos pertenecientes a la comunidad UCU fueran aceptados, al verificar el dominio antes de la inserción de datos. Además, se estableció una restricción en los números de teléfono, asegurando su unicidad y validando su formato según el estándar nacional uruguayo (598), lo que previene duplicados y asegura la coherencia de los datos ingresados.

Para seguir con las restricciones, se estableció que se impida que un alumno se inscriba en más de una clase durante el mismo turno. Dado que las clases cubren todo el horario de un turno específico, esta restricción evita solapamientos y conflictos en la asignación de recursos, como el espacio en las aulas y los materiales necesarios para cada actividad. De igual manera, se implementó una validación similar para los instructores, quienes no pueden ser asignados a más de una clase en el mismo turno. Esta medida asegura que cada clase cuente con un único instructor disponible y evita sobrecargar a los docentes con más de una clase simultáneamente.

Estas restricciones fueron implementadas de manera que, si un usuario intenta realizar una operación CRUD (create, read, update, delete) que viole estas normas, el sistema no permitirá la operación y devolverá un mensaje de error.

Por otro lado, se diseñaron y desarrollaron diversos endpoints en el backend para manejar operaciones CRUD. Estos endpoints permiten gestionar inscripciones, equipos, actividades, usuarios y clases de manera eficiente. Por ejemplo, el endpoint POST /inscription permite inscribir a un estudiante en una clase, validando su edad y el equipamiento asignado. Otros endpoints de tipo POST, permiten registrar nuevos usuarios y actividades, mientras que los endpoints de actualización, como PATCH, permiten modificar inscripciones y otros datos relacionados. Las validaciones implementadas en los endpoints aseguran que todas las restricciones de la base de datos se respeten, proporcionando una gestión eficiente y coherente de los datos. Estas medidas contribuyen al funcionamiento robusto y escalable del sistema, garantizando que las operaciones realizadas a través de la interfaz de usuario y el backend se realicen de manera eficiente y sin errores.

Además, se prestó especial atención a la gestión de roles y permisos dentro del sistema. Se definieron tres roles principales: administrador, instructor y estudiante. Cada uno de estos roles tiene permisos diferenciados sobre las operaciones que puede realizar. Un administrador tiene la capacidad de gestionar inscripciones, clases y usuarios, mientras que los instructores y alumnos solo pueden ver las opciones disponibles. Los instructores también pueden marcar las clases como dictadas. Este modelo de control de acceso permite un manejo más preciso y seguro de las operaciones dentro del sistema.

Otro aspecto importante fue el manejo de las claves primarias y foráneas. Las claves primarias auto incrementables en las tablas simplificaron la gestión de identificadores únicos, mientras que las claves foráneas permitieron establecer relaciones entre entidades, como entre los alumnos, las clases y las inscripciones. Esto garantizó la integridad referencial entre las tablas y facilitó la ejecución de consultas complejas.

La gestión de equipos e inventarios fue otra área en la que se implementaron mejoras significativas. Nuestra primera versión, permitía alquilar cada pieza de un equipo de manera individual, pero a diferencia de esto, decidimos introducir el concepto de kits de equipos. Estos kits agrupan todos los componentes necesarios para una clase o actividad específica, dividiendo únicamente por talle y tipo de actividad a la que se está anotando. Este enfoque simplifica la gestión de recursos, evita errores en la asignación y asegura que cada alumno cuente con los materiales necesarios para participar en la clase, en buen estado. Además, al inscribirse en una clase que requiere un kit, el sistema asigna automáticamente los recursos correspondientes, lo que optimiza la administración del inventario y mejora la experiencia del usuario. Está pensado por el contexto en el que se crea esta escuela de deportes, como en Uruguay no se pueden realizar estas actividades, la gran mayoría de los alumnos no tienen el equipamiento necesario. Al pensar la gestión de la escuela se tuvo en cuenta este hecho, así que la escuela se encargaría de mantener los equipamientos en buen estado, y transportarlos a la locación de las clases, por otro lado, permite más ingresos para la escuela.

Incorporamos también que el sistema tenga un mecanismo para el cálculo automático de costos asociado con cada inscripción. Por ejemplo, si un alumno se inscribe en una clase y a su vez alquila un kit de equipamiento, el sistema calcula automáticamente el costo relacionado con dicha clase y su kit correspondiente. Además, si se modifica o elimina su inscripción, el sistema ajusta tanto el inventario de equipos como los costos de manera automática, garantizando que toda la información esté actualizada en tiempo real.

Además, se introdujo una tabla exclusiva para el registro de clases dictadas, que no es accesible para los usuarios finales, pero que juega un papel crucial en la administración del sistema, ya que permite registrar detalles sobre la fecha y el id de la clase a la que pertenece, con todos sus respectivos datos. Esta tabla, a pesar de no tener utilidad particular para los requisitos pedidos, presenta una oportunidad de escalamiento del sistema a futuro, permitiendo más reportes, sistema de control de listas y otros.

Otro aspecto relevante fue la implementación de validaciones automáticas para verificar ciertos requisitos antes de permitir la inscripción de un alumno. Por ejemplo, se introdujo el uso de triggers en la base de datos para gestionar ciertos requisitos específicos de las operaciones. Un trigger es un mecanismo que se ejecuta automáticamente en respuesta a ciertos eventos en la base de datos, como INSERT, UPDATE o DELETE. En este caso, el trigger verifica antes de permitir una inscripción en la clase si el alumno cumple con la edad mínima requerida. Si el alumno no cumple con el criterio, el trigger evita que se inserte el registro de la inscripción, garantizando que no se violen las reglas preestablecidas institucionalmente y reduciendo el riesgo de errores durante el proceso de inscripción.

La implementación de estas restricciones y validaciones no solo se centró en la base de datos, sino que también se aplicó a nivel de backend. A través de validaciones en el servidor, se garantiza que las reglas de negocio, como la restricción de turno, sean verificadas antes de permitir cualquier modificación en los registros. Así, si un alumno intenta inscribirse en dos clases que se solapan, o un instructor intenta ser asignado a más de una clase en el mismo turno, el sistema rechazará automáticamente la operación, permitiendo una gestión consistente.

El diseño del sistema se pensó con una perspectiva de escalabilidad y mantenimiento. Si bien en un principio los requerimientos eran relativamente sencillos, se anticipó un crecimiento significativo en el número de usuarios, clases y recursos involucrados. Por esta razón, se adoptaron soluciones que no solo permiten manejar el volumen actual de datos, sino que también aseguran que el sistema pueda adaptarse a un aumento en la carga operativa sin comprometer su rendimiento.

Si bien en un principio se pensó hacer un sistema con login y autenticación de usuarios no fue posible, por lo que solo al administrador se le permite la inscripción de usuarios. En una segunda versión se implementaría un sistema de login y autenticación, con rutas protegidas que permitan al alumno inscribirse en las clases que desee y que el instructor solo pueda acceder a las clases que dicta.

Bitácora:

Durante las primeras semanas se hizo una lista de los elementos que faltaban a las tablas para poder cumplir con los requisitos de la entrega, desde la tabla de clase_dictada hasta datos extra de instructores como mail y teléfono, estos campos se pusieron por una eventual necesidad de una tabla personas que necesitara los mismos campos tanto de los alumnos como de instructores, además de que son datos relevantes que se necesitan de un empleado a nivel de gestión. Se creó una columna *grupal* en la tabla de clases que indica si una clase es grupal o individual.

Con la estructura de tablas hecha, se procedió a su creación e inserción de datos base, sobre la cual trabajar, así como una primera aproximación al frontend, creando los componentes, el ruteo de páginas y los modales, tanto de inserción como de creación de elementos.

Hubo dificultad para crear la API con Flask, esto generó retrasos en los avances del proyecto y no permitió una entrega tan completa como se había pensado en un principio. Al tener la API, se procedió a crear los endpoints y a su vez hacer la conexión con el frontend. A pesar de varios intentos de implementación de jwt o un sistema de login, no hubo éxito, por lo que se rediseñó un poco la lógica del frontend.

Sobre la última semana algunos de los endpoints no funcionaban como se esperaba, especialmente las restricciones, pero logramos solucionar estos parcialmente, enfocándonos en los casos más prioritarios para garantizar la funcionalidad básica del sistema. Esto implicó realizar ajustes rápidos tanto en la lógica del backend como en el manejo de las respuestas en el frontend. Aunque no se alcanzó la implementación completa de todas las funcionalidades planeadas, los aspectos fundamentales, como la creación, consulta y actualización de registros, fueron funcionales y demostraron el potencial del sistema.

Bibliografía

- *Welcome to Flask — Flask Documentation (1.1.x)*. (n.d.). Flask.palletsprojects.com. https://flask.palletsprojects.com/
- MySQL. (2019). *MySQL :: MySQL Documentation*. Mysql.com. https://dev.mysql.com/doc/
- *Getting started | DataGrip*. (n.d.). DataGrip Help. https://www.jetbrains.com/help/datagrip/getting-started.html
- *DataGrip Help*. (2024). DataGrip Help. https://www.jetbrains.com/help/datagrip/db-tutorials-introduction.html
- *DataGrip Help*. (2024). DataGrip Help. https://www.jetbrains.com/help/datagrip/connecting-to-a-database.html