UCU
SNOW
SCHOOL

Proyecto final

BASE DE DATOS I

UCU SNOW SCHOOL

INTEGRANTES DEL PROYECTO

- Franco Filardi
- Manuela Guédez
- Mateo Hernandez
- Agustín Pose

SEMANA 1

12/10/2024 - 18/10/2024

TÓPICO	DESCRIPCIÓN	FECHA
Lectura del proyecto	Realizar de forma conjunta una lectura general de la letra del proyecto con el fin de poder entender a qué queremos llegar y desde dónde partimos.	14/10/2024
División de tareas	En base a las fortalezas y preferencias de los integrantes del equipo realizar una primera división de tareas, asignándonos referentes y encargados de frontend, backend y lógica de negocio.	15/10/2024

SEMANA 2

19/10/2024 - 25/10/2024

TÓPICO	DESCRIPCIÓN	FECHA
Creación de repositorio	Crear repositorio para el proyecto.	19/10/2024
Investigación para implementación	Investigar diferentes formas de conectar backend en Python con frontend desarrollado con librería de React, preferentemente ReactJS.	20/10/2024
API Flask	Leer cómo levantar una API desde Python con la librería Flask y qué dependencias son necesarias para levantar el proyecto	22/10/2024
Creación de base de datos	Escribir primer archivo para crear la base de datos con las tablas y atributos dados en la letra del proyecto.	23/10/2024

SEMANA 2 cont.

19/10/2024 - 25/10/2024

TÓPICO	DESCRIPCIÓN	FECHA
Conexión BE con BD	Realizar las primeras conexiones del backend en Python con la base de datos creada, realizar consultas para corroborar el correcto funcionamiento	24/10/2024
Proyecto frontend	Crear proyecto para frontend con ReactJS, implementando los componentes para las páginas de la aplicación y lógica básica para la interacción con el usuario	25/10/2024
Implementación de Flask	Instalar dependencias y realizar endpoint para loguear un usuario en la aplicación y corroborar el correcto funcionamiento con Postman	25/10/2024

SEMANA 3

26/10/2024 - 1/11/2024

TÓPICO	DESCRIPCIÓN	FECHA
Listado de funciones necesarias	Realizar un listado con las funciones que tendría que tener la aplicación según la letra del poryecto	27/10/2024
Definición de endpoints	Realizar un listando con los endpoints necesarios para poder implementar los requerimientos del proyecto.	28/10/2024
smtp	Tras conversar con el docente, se decidió implementar un algoritmo con el que podamos avisar a alumnos e instructores sobre ABM de clases de las que son partícipes	1/11/2024

SEMANA 4

2/11/2024 - 8/11/2024

TÓPICO	DESCRIPCIÓN	FECHA
Reunión	Como consecuencia de haber realizado las primeras implementaciones de ABM de clases, nos dimos cuenta de que con las tablas con las que contábamos no era suficiente para lograr que el sistema emule clases como esperábamos que lo haga, por lo que se necesita organizar una reunión para agregar / modificar tablas para lograrlo.	2/11/2024
Implementación modificación de clases	Realizar implementación de endpoint para la modificación de clases	6/11/2024
Reunión 2.0	Realizar una reunión para proponer diferentes formas de modelar el problema (información de clases con días, fechas e implementación de pasaje de lista) para consultar durante la clase.	7/11/2024

SEMANA 5

9/11/2024 - 15/11/2024

TÓPICO	DESCRIPCIÓN	FECHA
Frontend	Interfaz frontend de instructor y alumno terminadas, con conexión al backend pendiente	10/11/2024
Creación de clases	Tras reunión realizada, se decidió agregar los atributos start_date y end_date para poder delimitar las fechas de las clases. Esta información nos es útil para poder, con un algoritmo, determinar las fechas en las que las clases deben dictarse, estos datos se almacenarán en una nueva tabla class_session.	13/11/2024
Interfaz Administrador	Interfaz de admin finalizada en el frontend, teniendo el admin acceso a clases, horarios, alumnos, instructores y pudiendo modificarlos.	15/11/2024

SEMANA 5 cont.

9/11/2024 - 15/11/2024

TÓPICO	DESCRIPCIÓN	FECHA
Pasaje de listas	Se implementó la funcionalidad de pasaje de lista; como consecuencia de que se tienen las fechas en las que se deben dictar las clases; se obtienen los alumnos anotados y luego se almacenan en una nueva tabla class_attendance los alumnos que asistieron. Para esto se almacenan los siguientes datos: id_class_session, student_id y attended (booleano que determina si el alumno asistió o no). Una vez pasa la lista, se marca como dictada la clase.	15/11/2024
Inscripción y dar de baja estudiantes	Se implementaron las funcionalidades de inscribir y dar de baja a estudiantes de cursos. Para poder realizar esto se tuvo que realizar una extensa consulta que verifique que el estudiante efectivamente pueda inscribirse a la clase (que no se superponga con otra).	16/11/2024

SEMANA 5 cont.

9/11/2024 - 15/11/2024

TÓPICO	DESCRIPCIÓN	FECHA
Activities	Se implementaron endpoints para crear nuevas actividades y modificar los precios de las ya existentes	17/11/2024
Dockerización	Se realizó una primera prueba para dockerizar el proyecto	17/11/2024

SEMANA 6

18/11/2024 - 22/11/2024

TÓPICO	DESCRIPCIÓN	FECHA
Turnos	Se implementaron endpoints para crear nuevos turnos y modificar los horarios de los ya existentes	18/11/2024
Soft-delete	Se implementó un soft-delete, agregando una nueva columna de tipo booleano a todas las tablas que se ven afectadas por los eliminar implementados. Como por ejemplo, en caso de que se elimine un instructor, si este tenía clases asignadas se emula el cascade, eliminando setteando como eliminadas las clases que dictaba, y las inscripciones de los alumnos a esas clases, así como el equipamiento que habían rentado.	20/11/2024

UCU SNOW SCHOOL

ESQUEMA RELACIONAL RESULTANTE

UCU SNOW SCHOOL

login (email, password, person_ci, role_id, is_deleted)

activities (activity_id, description, cost)

equipment (equipment_id, activity_id, description, cost)

equipment_rental(class_id, person_id, equipment_id)

instructors (person_id, instructor_ci, first_name, last_name, is_deleted)

turns (turn_id, start_time, end_time, is_deleted)

students (person_id, student_ci, first_name, last_name, birth_date, is_deleted)

classes (class_id, instructor_ci, activity_id, dictated, start_date, end_date, is_group, is_deleted)

student_class (class_id, student_ci, equipment_id, is_deleted)

roles (role_id, role_name)

person (person_id, person_ci, name, last_name, is_deleted)

days (day_id, day_name)

class_session (id_class_session, class_id, class_date, day_id, dictated, is_deleted)

class_day (class_id, day_id, is_deleted)

class_attendance (id_class_session, student_id, attended)

UCU SNOW SCHOOL

FUNDAMENTACIÓN DE DECISIONES DE IMPLEMENTACIÓN

UCU SNOW SCHOOL

ATRIBUTOS NUEVOS EN LOGIN . . .

login (email, password, person_ci, role_id, is_deleted)

El sistema desarrollado maneja **diferentes tipos de inicio de sesión**, esto con el fin de poder realizar una primera capa de control, otorgando o quitando vistas de funciones según el rol correspondiente al usuario logueado

Para poder administrar y persistir esta información, se agregó la columna person_ci, con el fin de poder generar una relación entre la cuenta y la persona correspondiente; así como también el rol (role_id) al cual corresponde el usuario que ha ingresado.

Por último, se agregó el atributo is_deleted, con el fin de poder realizar una eliminación lógica de los usuarios. Gracias a esto, los datos persisten en la base de datos más allá de que se haya dado de baja a la persona del sistema.

Tanto person_ci como role_id corresponden a claves foráneas de dos nuevas tablas...

. . . TABLAS NUEVAS

UCU SNOW SCHOOL

person (person_id, person_ci, name, last_name, is_deleted)
roles (role_id, role_name)

Como consecuencia de que se nos proporcionaron dos tablas diferentes (una para alumnos y otra para instructores) y precisábamos homologar la información para poder relacionarla en una misma tabla de login, decidimos crear la tabla **person** con el fin de poder partir de ella para generar los nuevos registros tanto de instructores como de estudiantes.

En esta, se persiste un person_id que será usada como primary key, con la cual podamos identificar a las personas dentro del sistema, y luego los atributos que tenían en común instructor y student (nombre, apellido y ci).

Se creó una nueva tabla **roles** con el fin de poder restringir los roles con los que un usuario puede ser agregado al sistema. Siendo estos student, instructor, y por defecto hay uno ya agregado con el rol de administrador.

UCU SNOW SCHOOL

ATRIBUTOS NUEVOS EN INSTRUCTORS Y STUDENTS

instructors (person_id, instructor_ci, first_name, last_name, is_deleted)

students (person_id, student_ci, first_name, last_name, birth_date, is_deleted)

Agregada la tabla de personas, se utilizó como primary key el id de las personas, y junto con instructor_ci / student_ci son foreign keys.

De igual forma, se agregó el atributo is_deleted para poder propagar el delete, cuando se eliminan las personas del sistema.

UCU SNOW SCHOOL

ACTUALIZACIONES EN TABLA CLASSES

classes (class_id, instructor_ci, activity_id, dictated, start_date, end_date, is_group, is_deleted)

Para poder implementar el registro de las clases que hay, se agregaron los atributos de fecha de inicio y de fin del curso, así como también dos booleanos, **is_group** para determinar que la clase sea grupal o individual e **is_deleted**, con el fin de que si un instructor es eliminado se pueda dar de baja las clases de las que era responsable.

Se dejó en desuso el atributo **dictated**, entendimos que esta tabla almacena de forma abstraca la información inicial de la clase, por lo que no correspondía marcarla como dictada o no con la información que aquí se almacena.

¿CÓMO MARCAMOS COMO DICTADAS LAS CLASES?

UCU SNOW SCHOOL

days (day_id, day_name)
class_session (id_class_session, class_id, class_date, day_id, dictated, is_deleted)
class_day (class_id, day_id, is_deleted)

Sin embargo, que no podamos usar el dictated en la tabla anterior, no quiere decir que no lo hayamos implementado . . .

Para poder marcar las cases como dictadas, entendimos que debíamos tener clases que dictar, lo cual necesitaba de fechas específicas. En base a esto fue que decidimos crear las tablas **days** y **class_days**, con estas podemos almacenar los días en los que han de dictarse las clases.

Desde la aplicación, a la hora de crear las cases, se pide que se ingrese, además de los datos que se almacenan en la tabla classes, los días; como almacenamos fecha de inicio y de fin, se usa un algoritmo que calcula las fechas en las que la clase debe ser dictada y esto se almacena en la tabla **class_session**.

¿CÓMO MARCAMOS COMO DICTADAS LAS CLASES?

UCU SNOW SCHOOL

class_attendance (id_class_session, student_id, attended)

Una vez almacenadas las fechas en las que se deben dictar las cases, es posible desde la aplicación, mostrar al instructor las clases que tiene que dictar en el día de la fecha. Con estas, puede acceder a la lista de alumnos inscriptos y marcar la asistencia; para poder marcar la asistencia de los alumnos se creó la tabla **class_attendance**, en la que se almacena el id de la clase (class_session), para poder matchear ocn la fecha y la clase específica, así como también un booleano que determina la eventual asistencia del alumno con el id especificado.

Luego de ingresados los datos en la tabla de asistencias, se marca la clase como dictada, esto es, se realiza un update en la tabla class_session con la nueva información.

UCU SNOW SCHOOL

PARA LA LISTA DE ASISTENCIA LOS ALUMNOS DEBEN ESTAR INSCRIPTOS

student_class (class_id, student_ci, equipment_id, is_deleted)

Para poder mantener la información de las inscripciones de los alumnos a las clases, se agregó un único atributo, el is_deleted, con el fin de poder persistir la información a futuro, en caso de que se dé de baja a la persona del sistema

Se dejó en desuso la columna de equipment como consecuencia de que entendimos que para una misma clase el alumno podría llegar a requerir más de un alquiler, y como la PK es el id de la clase, no podíamos tener filas con ids repetidos.

UCU SNOW SCHOOL

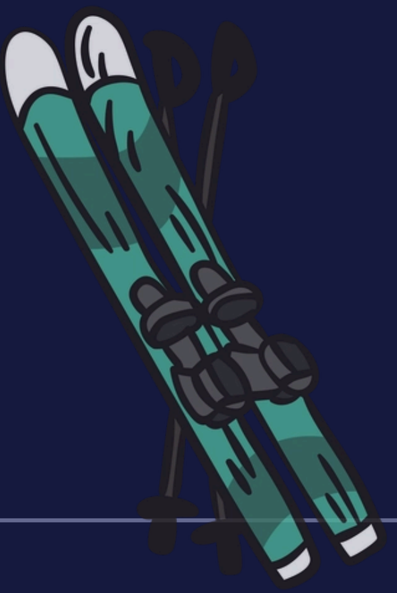
ALQUILER DE EQUIPAMIENTO . . .

equipment (equipment_id, activity_id, description, cost)

equipment_rental(class_id, person_id, equipment_id)

Para poder implementar la funcionalidad de alquiler de equipamiento se creó una nueva tabla en la que se registran los alumnos que hayan alquilado equipos para sus clases.

Esto soluciona el problema de IDs repetidos dado que toma todos los atributos como PK, permitiendo al alumno rentar más de un equipo para la misma clase.



UCU SNOW SCHOOL

¿CÓMO INTERACTUAR CON EL SISTEMA CREADO?

UCU SNOW SCHOOL

BACKEND DEVELOPMENT

Se desarrolló un programa en python que realiza una conexión con la base de datos y realiza las consultas, inserciones y actualizaciones necesarias.

Se dividió el código en dos archivos principales:

app.py

Utiliza las librerías de **Flask** para poder levantar una API para ser accedida desde el frontend y la libraría de JWT para poder generar tokens de autenticación.

services.py

Utiliza la librería **MySQL connector**, con la que se realiza la conexión directa con la base de datos, es quien tiene los scripts para las queries.

UCU SNOW SCHOOL

FRONTEND DEVELOPMENT

La aplicación frontend se desarrolló con ReactJS

Posee tres vistas principales:

- De administrador
- De instructor
- De alumno

Para ver más detalles del proyecto acceder al repositorio con este link