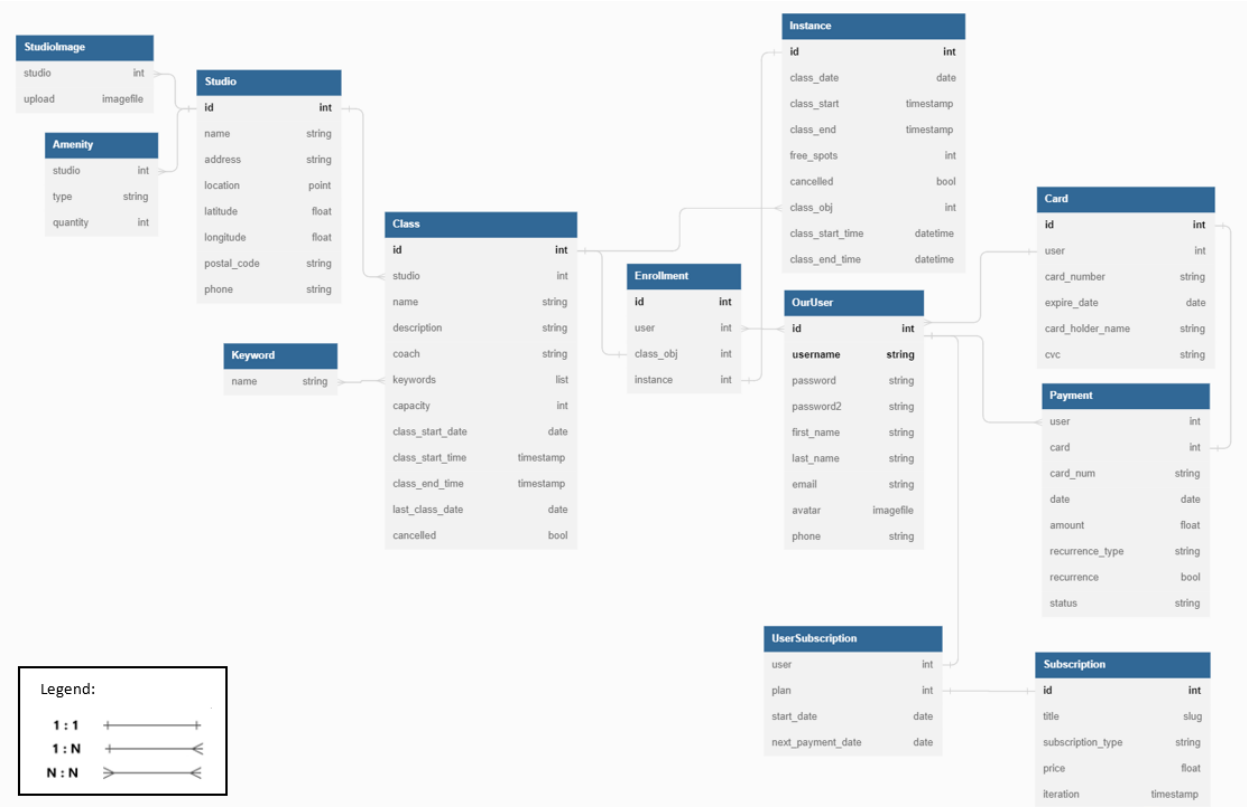
Project Documentation

Authors: Nicolae Binica, Bingcan Guo, Valentina Manferrari

Design of Models

The project models can be divided into 4 macro-areas: Users, Studios, Classes, and Subscriptions. Every *studio* can have multiple *classes*, each of which can have multiple *instances* recurring weekly. A *user* owning one or more *cards* can make a *payment* to purchase a *subscription*. Users that have active subscriptions can enroll in *classes*. The following diagram shows in greater detail all the relationships between each model:



ER Diagram Source Link: https://dbdiagram.io/d/6375bdc2c9abfc6111734a2e

API Endpoints

The following section contains a full comprehensive list of all the APIs used in this project along with their description and specifications (Method, Authorization, Parameters).

`{{base_url}}` value is http://127.0.0.1:8000/. Note: parameters marked by a star (*) are required fields.

Users

Signup

Allows the user to create an account and generates an authorization token for future logins. Note that email and phone are not required but if you decide to input them you will have to input valid values (i.e. a valid email in the format email@something.ext and a phone number of length ###-###-####)

URL: `{{base_url}}/accounts/signup/`

Method: POST

Authorization: None

Parameters: username*, password*, password2*, email, first_name, last_name, avatar, phone

---

Login

Allow the user to log in using their credentials, a token is generated for future actions authentication.

URL: `{{base_url}}/accounts/login/`

Method: POST

Authorization: None

Parameters: username*, password*

Edit Profile

Call used to edit the profile of a user. All the password fields (old_password, password1, password2) must be included if the user wants to change the password. The email, if included, must have the correct format.

There are 2 types of available methods. PUT method requires all fields to be included in the parameters. PATCH method allows for partial update of a user profile

URL: `{{base_url}}/accounts/edit_profile/`

Method: PUT/PATCH

Authorization: `Bearer {{JWT Token}}`

Parameters: username*, old_password*, password*, password2*, email, first_name, last_name, avatar, phone

---

Verify Token

This API can be used to verify if a given token is valid or expired.

URL: `{{base_url}}/accounts/verify_token/`

Method: POST

Authorization: None

Parameters: token*

Studios

List Studios by Location

This API can be used by a user to see which studios are closer to them by providing their location coordinates. The studios will be ordered by proximity and within the specified distance radius (in km). If no radius parameter is specified, the API will return all existing studios.

URL: `{{base_url}}/studios/list_by_location/`

Method: GET

Authorization: None

Parameters: Latitude*, Longitude*, radius

View Studio Details

This API can be used to view the general information of a specified studio.

URL: `{{base_url}}/studios/get_studio_page/<int:studio_id>/`

Method: GET

Authorization: None

Parameters: None

Search Studio with Filters

This API can be used to query studios based on studio name, class name, coach, or amenity filters. Note that you can use multiple filters at the same time.

URL: `{{base_url}}/studios/search_studios/`

Method: GET

Authorization: None

Parameters: name, coach, class_name, amenity

Classes

View All Studio Classes

This API can be used to view all classes happening (i.e. not cancelled) in a given studio.

URL: `{{base_url}}/classes/get_all_studio_classes/<int:studio_id>/`

Method: GET

Authorization: None

Parameters: None

View Studio Schedule

This API can be used to view all the instances of a class of a given studio that are yet to happen.

URL: `{{base_url}}/classes/get_studio_schedule/<int:studio_id>/`

Method: GET

Authorization: None

Parameters: None

Enroll in a Class

This API can be used by a registered user with an active subscription to enroll in a certain class, provided the selected class has enough available spots.

URL: `{{base_url}}/classes/enroll_class/`

Method: POST

Authorization: `Bearer {{JWT Token}}`

Parameters: user, class_obj

Drop a Class

This API can be used by a user currently enrolled in a class to drop any instance of that class provided the specific class and instance ids. If the user does not provide the instance id they will unenroll from the entire class.

URL: `{{base_url}}/classes/drop_class/`

Method: POST

Authorization: `Bearer {{JWT Token}}`

Parameters: user, class_obj, instance

View User Class History

This API can be used by a registered user to view their entire class history (i.e. all registered past class instances ordered chronologically).

URL: `{{base_url}}/classes/user_class_history/`

Method: GET

Authorization: `Bearer {{JWT Token}}`

Parameters: user

Search Class Schedule

This API can be used by a registered user to view their class schedule (i.e. all registered future class instances ordered chronologically).

URL: `{{base_url}}/classes/user_class_schedule/`

Method: GET

Authorization: `Bearer {{JWT Token}}`

Parameters: user

Search Studio Schedule

This API can be used to search among all instances of classes that are yet to happen in a given studio ordered chronologically.

URL: `{{base_url}}/classes/search_studio_schedule/<int:studio_id>/`

Method: GET

Authorization: None

Parameters: studio_id

Subscriptions

View All Subscription Options

This API can be used to view all available subscription options a user can subscribe to.

URL: `{{base_url}}/subscriptions/api/view/all/`

Method: GET

Authorization: None

Parameters: None

Manage User Subscription

This API can be used to view a user's subscription using the GET method (no parameters needed), and modify user's subscription using the POST method(upon 'action' and selected subscription passed from request)
Only users with registered cards and have no current subscription can subscribe.
Only users with registered cards and current subscriptions can update their subscriptions.
Subscription plan informations, start day, and related payments will be updated.
Only user with current subscriptions can delete their subscriptions. User subscription information will be updated, and related payments will be updated to CANCELLED.

This API also checks for auto payment:

- If today is the scheduled payment day, make current payment, schedule future payment, and update user subscription's next_payment_date. All will be done automatically.

URL: `{{base_url}}/subscriptions/api/my_subscription/`

Method: GET / POST

Authorization: `Bearer {{JWT Token}}`

Parameters: action, plan

View All Payments

This API can be used to view a user's payment history, including payments whose status are SUCCESS, UNPAID, and CANCELLED.

SUCCESS payments are paid already, card num cannot be changed.
UNPAID payments will be automatically paid on the scheduled day with updated card. If user has no valid credit/debit card, UNPAID payments card numbers will be updated to 0000000000000000.
CANCELLED payments have no way to be paid again, card num cannot be changed.

URL: `{{base_url}}/subscriptions/api/payment/view/`

Method: GET

Authorization: `Bearer {{JWT Token}}`

Parameters: None

View and Edit Credit Card

This API can be used to view a user's card using the GET method (no parameters needed), and modify user's card using the POST method('action' and card information(if any) passed from request)

Assume a user can only have one card, as consulted in tutorial mentoring session 1.

Only users no card can add card via 'create' action. Input card informations will be validated: all fields are required; customized validation for card number and cvc; can't add an expired card. Related payments will be updated.
Only users with current card can update their card info via 'update' action. Input card informations will be validated: all fields are required; customized validation for card number and cvc; can't add an expired card. Note all fields need to be entered again. Related payments will be

updated.
Only user with current card can delete their card info via ‘delete’ action. Related payments will be updated, as UNPAID payments card numbers will be updated to 0000000000000000.

URL: `{{base_url}}/subscriptions/api/card/`

Method: GET / POST

Authorization: `Bearer {{JWT Token}}`

Parameters: action, card_id, card_num, card_holder_name, expire_date, cvc