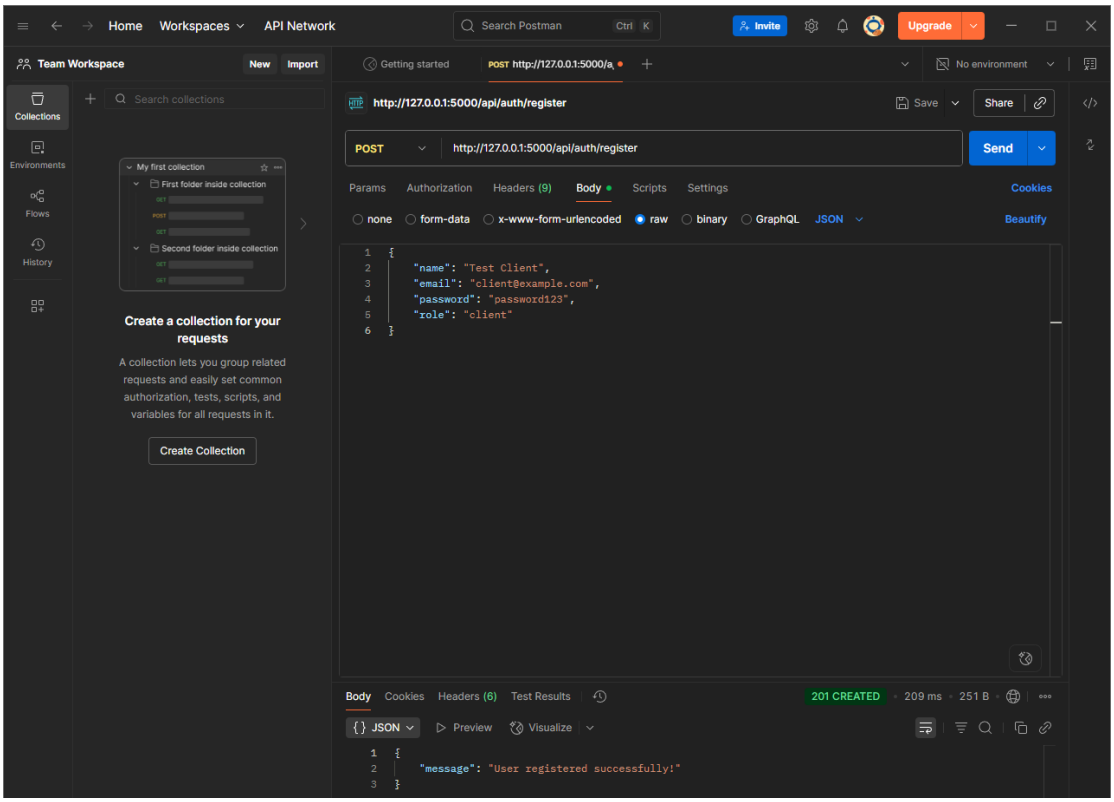
FitConnect Project: Backend API Development Plan

To support the "FitConnect" application, the backend will provide a clear and secure set of API endpoints. These endpoints will be responsible for handling all data operations, such as user management, trainer information, services, and bookings.

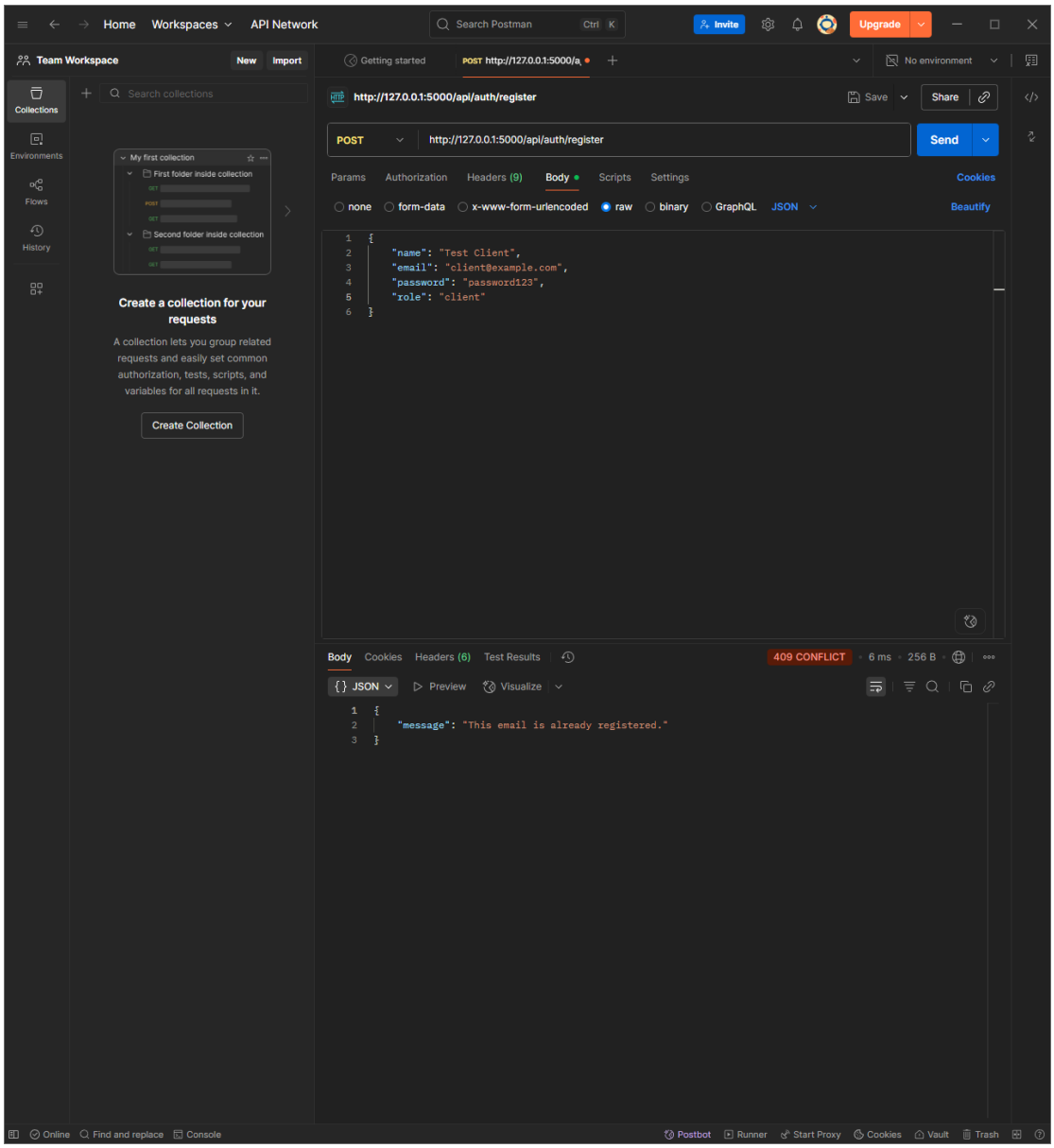
1. Authentication APIs

This is the foundation of the application, responsible for user registration and login.

- **POST /api/auth/register - User Registration**
    - **Functionality**: Creates an account for a new user (either a trainer or a client).
    - **Frontend Sends**: The user's name, email, password, and a role ("trainer" or "client").
    - **Backend Operation**: Validates if the email is already registered, encrypts the password, and then stores the new user information in the database.
    - **Returns**: A success message or an error notification.
- **Testing**



Try registering an account

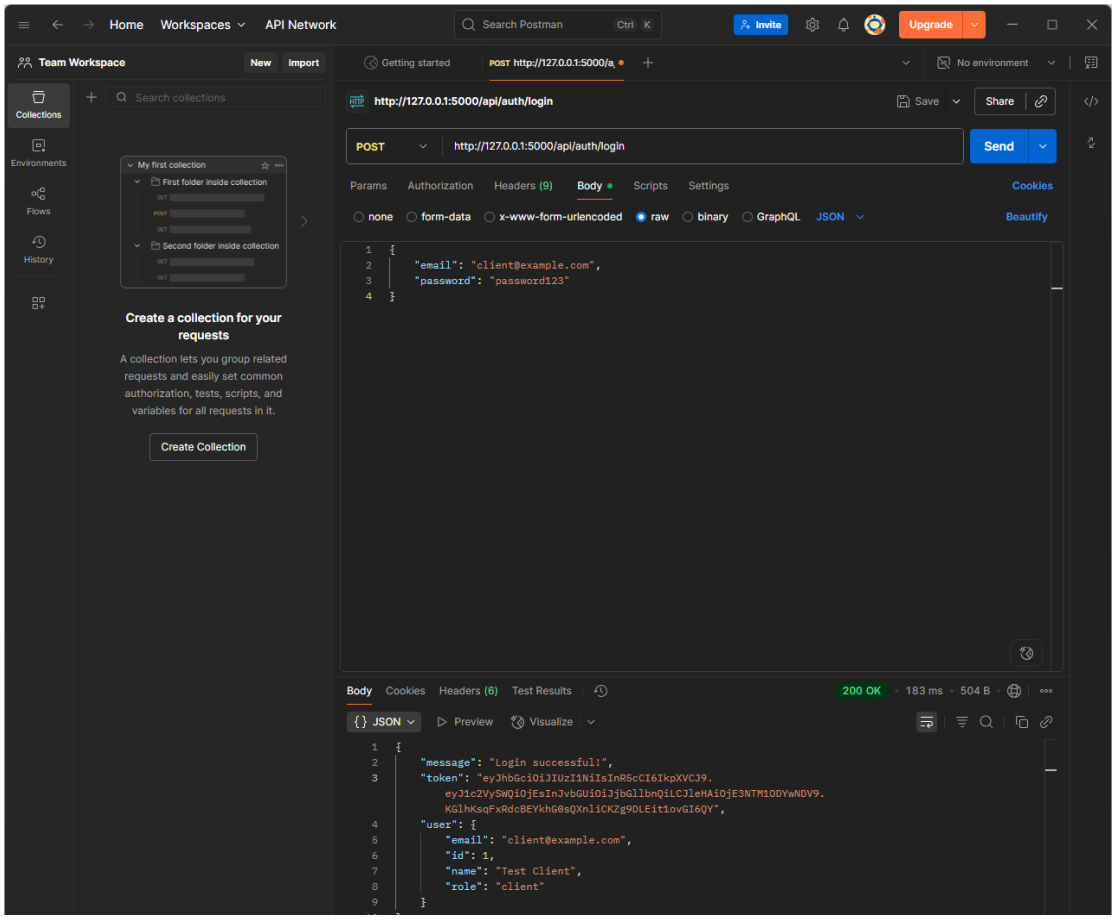


If we register with the same account, it will return”is already registered”

	id	name	email	password	role	created_at
▶	1	Test Client	client@example.com	$2b$12$OSmE52waFsLS30qLIrgxbuwtdZbyJEc...	client	2025-07-27 01:58:44
	NULL	NULL	NULL	NULL	NULL	NULL

In the database we have the record

- **POST /api/auth/login - User Login**
    - **Functionality**: Verifies a user's identity and allows them to access the application.
    - **Frontend Sends**: The user's email and password.
    - **Backend Operation**: Finds the user in the database by email and verifies if the password is correct.
    - **Returns**: On success, returns a token for authenticating subsequent requests; on failure, returns an error message.
- **Testing**

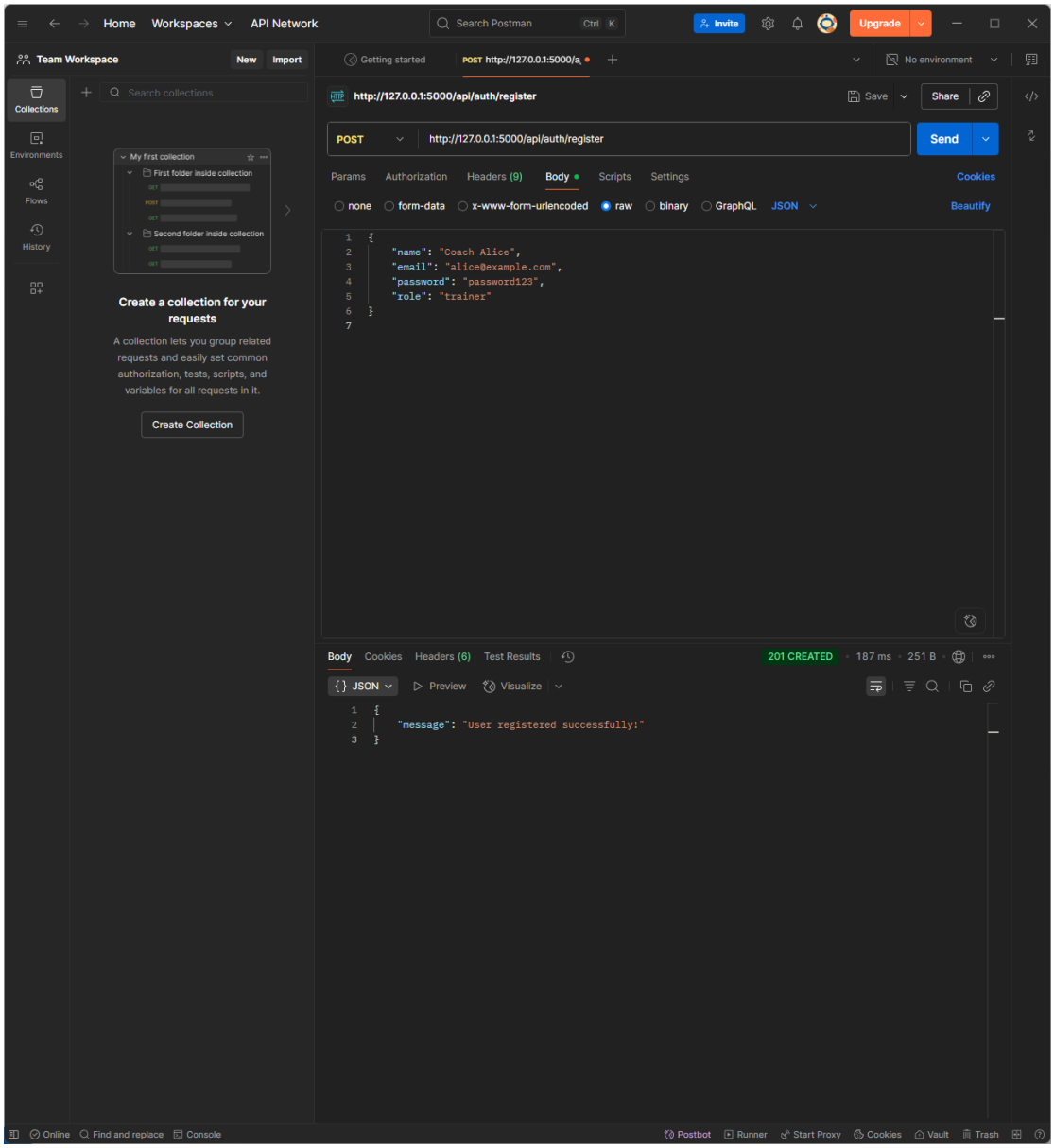


When we log in with the account we create, it will return “login successful”

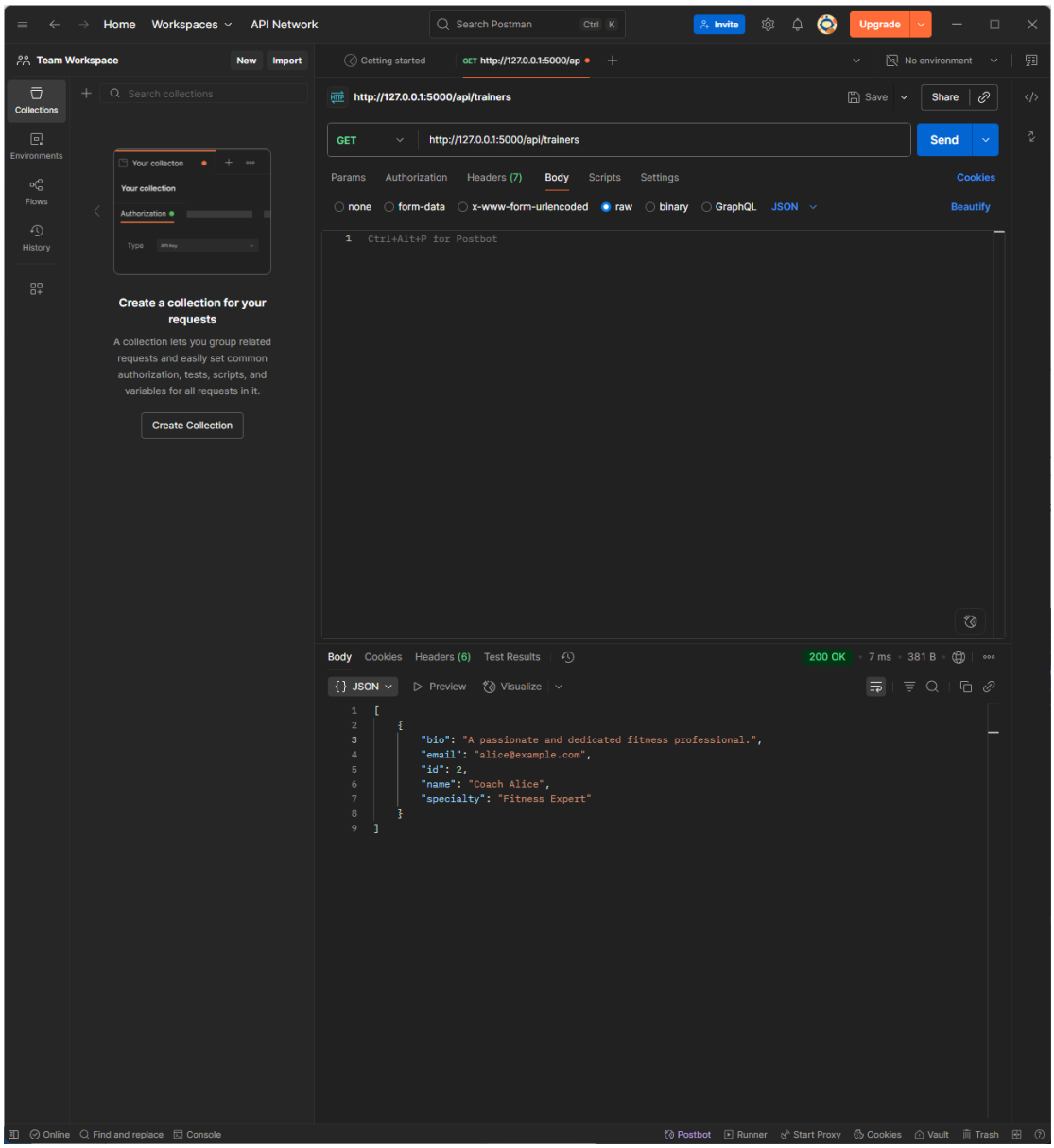
2. Trainers APIs

These endpoints are responsible for displaying and managing trainer information.

- **GET /api/trainers - Get All Trainers**
    - **Functionality**: This is the core endpoint needed for the "Featured Trainers" section on your website's homepage.
    - **Frontend Sends**: Nothing.
    - **Backend Operation**: Queries the database for all public information of users with the role of "trainer" (e.g., name, specialty, avatar).
    - **Returns**: A list containing the information of all trainers.

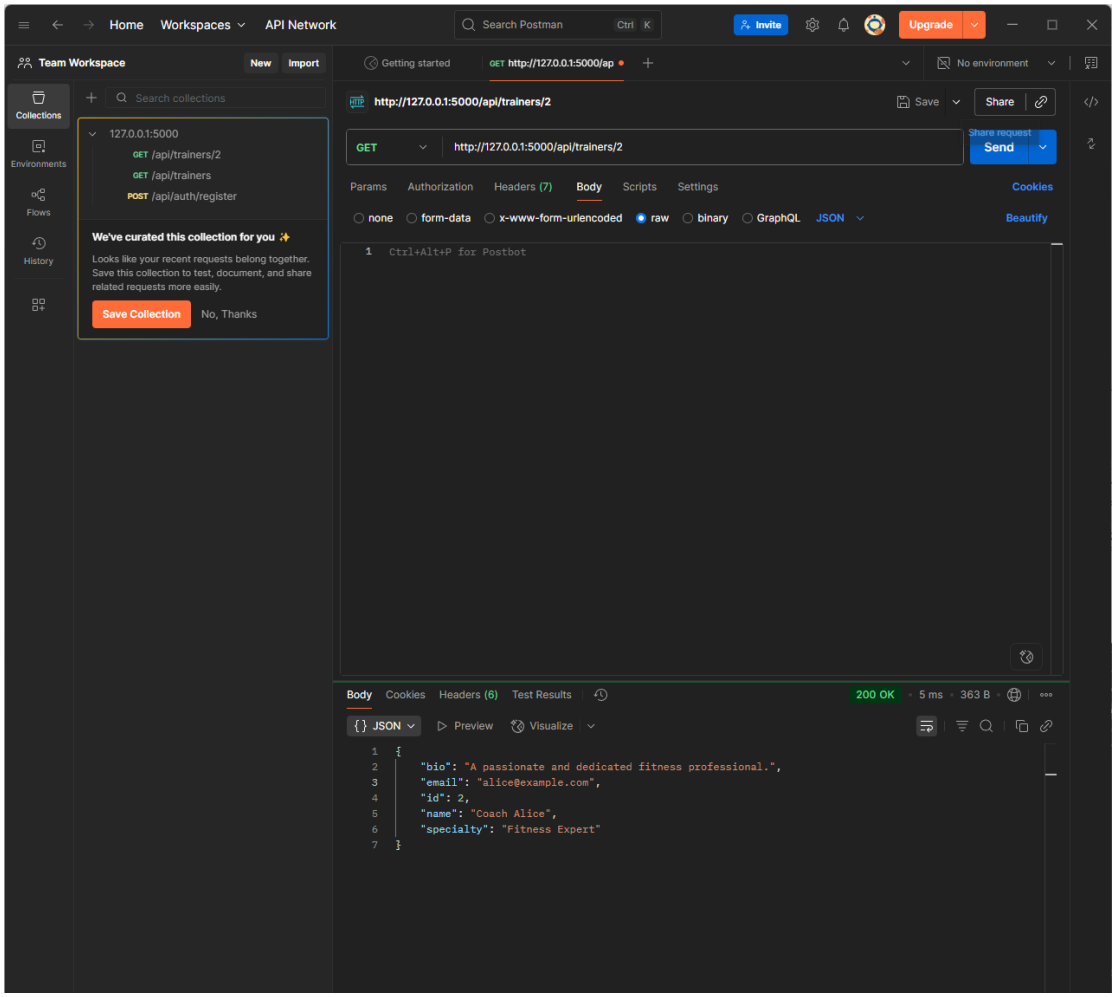


We registered a trainer account first



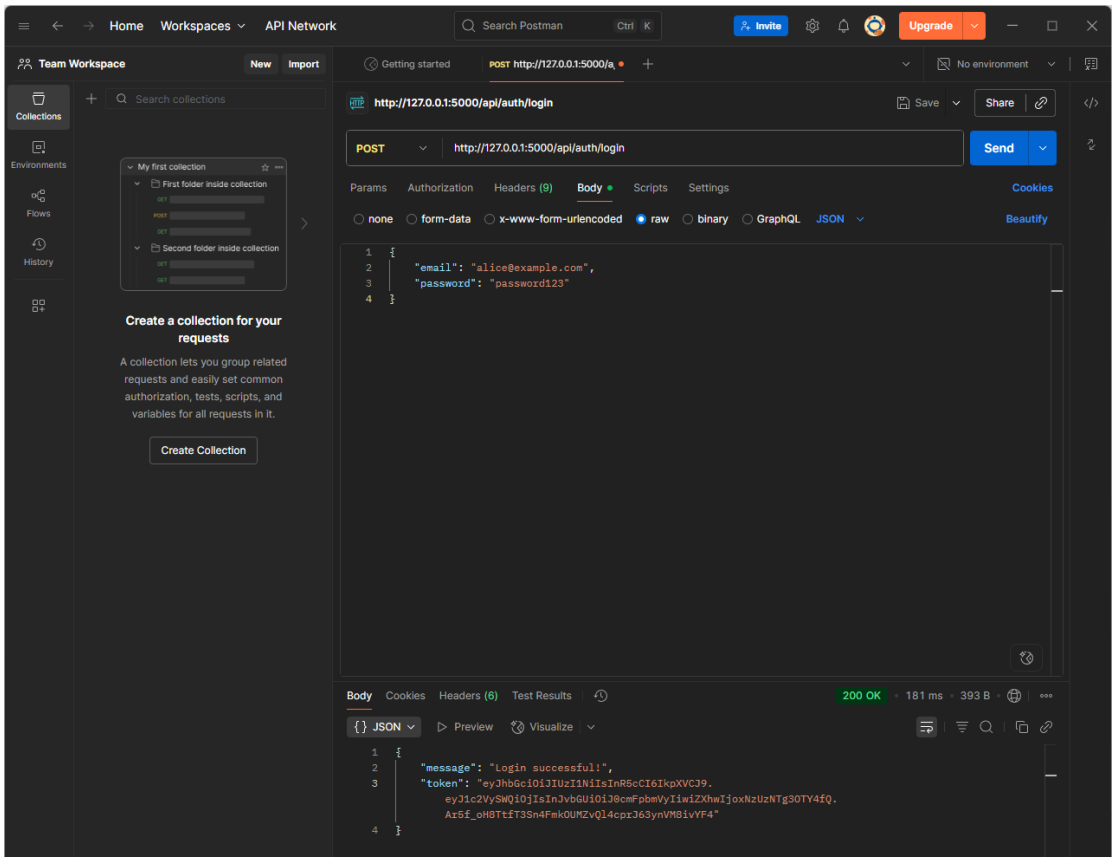
Then use ‘GET’ to get all the trainers we created.

- **GET /api/trainers/:id - Get Single Trainer Details**
    - **Functionality**: Called when a user clicks on a trainer's card to view their detailed profile.
    - **Frontend Sends**: The unique ID of the trainer.
    - **Backend Operation**: Queries the database for the detailed profile of the specific trainer based on the ID, including all services they offer and their availability.
    - **Returns**: The detailed information for that trainer.
- **Testing**

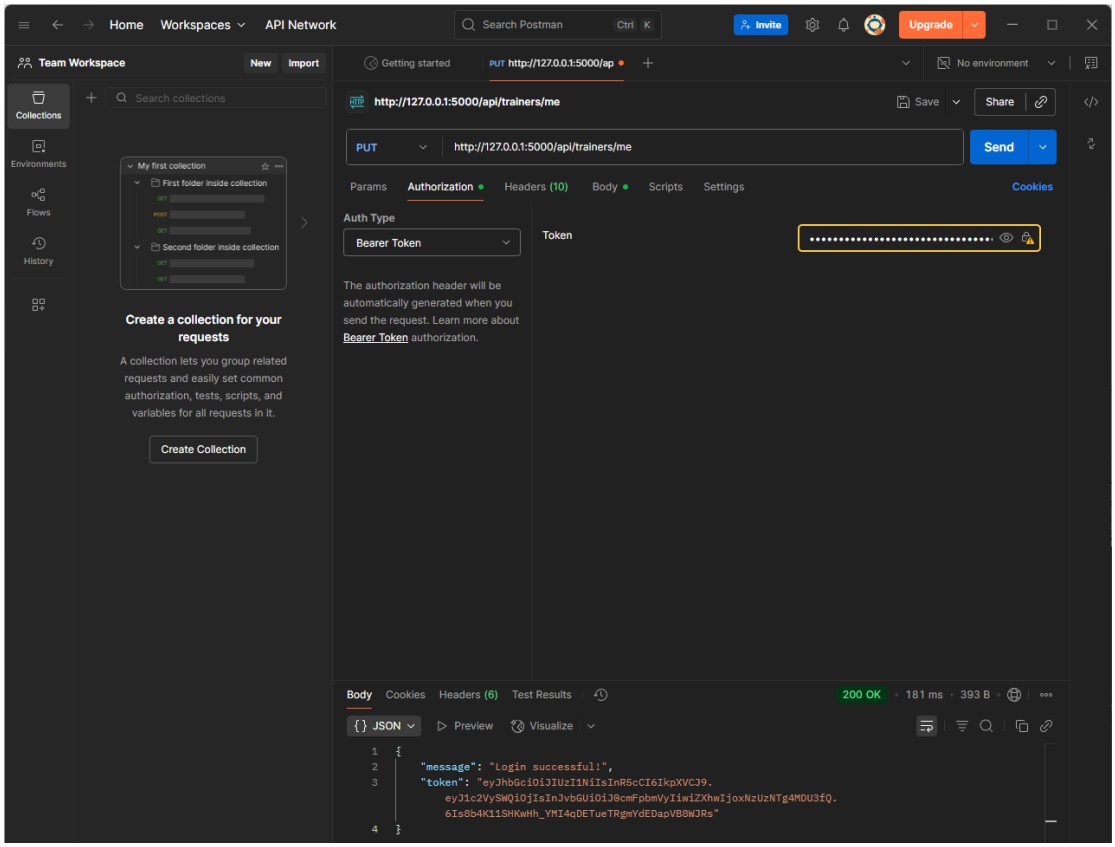


We use the specific trainer id “2” to get the trainer’s detail

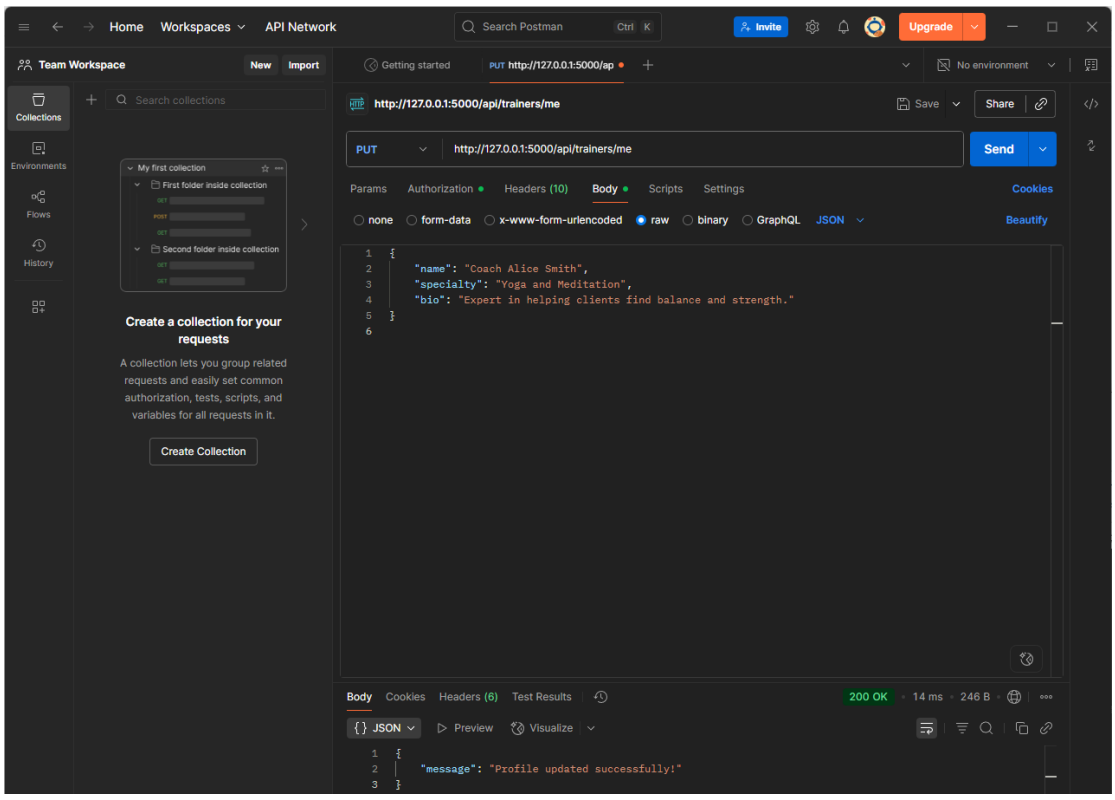
- **PUT /api/trainers/me - Update Trainer Profile (Login Required)**
    - **Functionality**: Allows a logged-in trainer to update their own personal introduction, specialty, avatar, etc.
    - **Frontend Sends**: The updated trainer information.
    - **Backend Operation**: Verifies the user's identity and then updates the corresponding trainer record in the database.
    - **Returns**: The updated trainer information after a successful update.
- **Testing**



We login in the trainer account first and got the token



Enter token

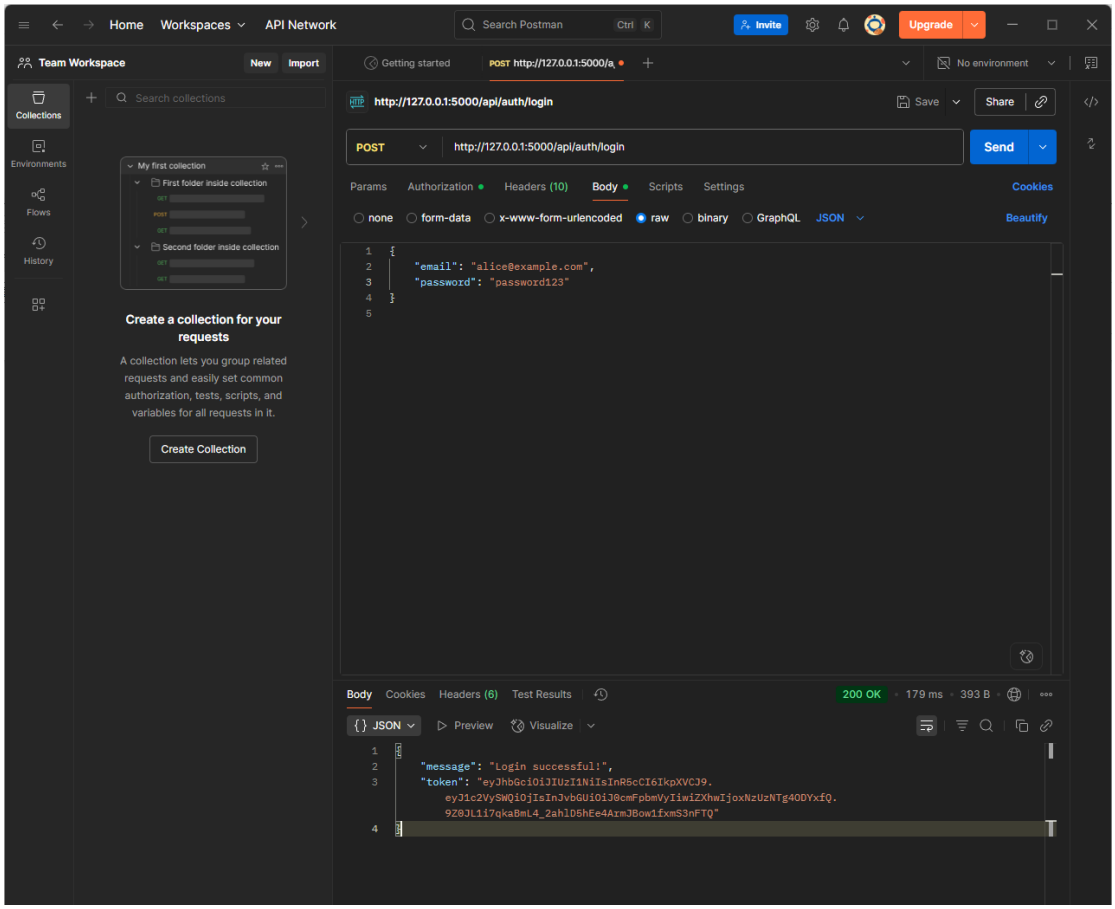


Use “PUT” to update the trainer’s information

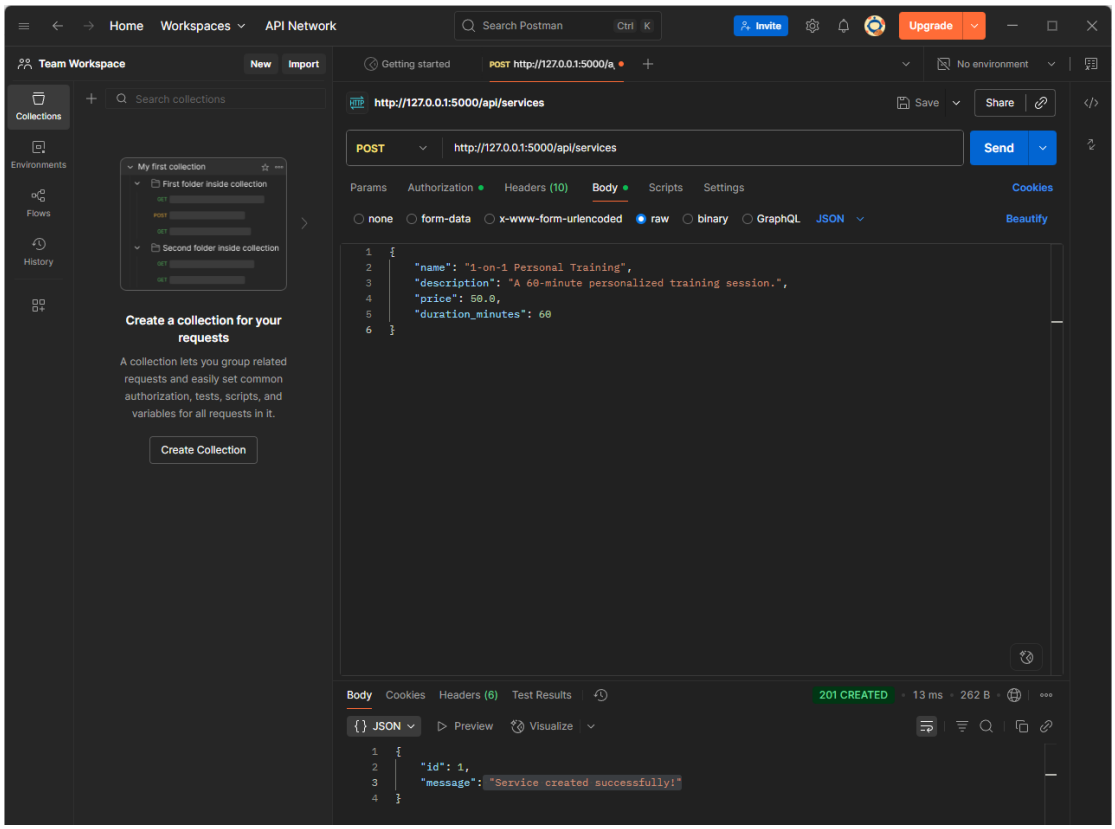
3. Services APIs

These endpoints allow trainers to manage the courses or services they offer.

- **POST /api/services - Create New Service (Trainer Login Required)**
    - **Functionality**: Allows a trainer to add a new service item, such as a "One-Hour Personal Training Session" or a "Fat Loss Camp."
    - **Frontend Sends**: The service's name, description, duration, price, etc.
    - **Backend Operation**: Associates the new service with the currently logged-in trainer and saves it to the database.
    - **Returns**: The successfully created service information.
- **Testing**

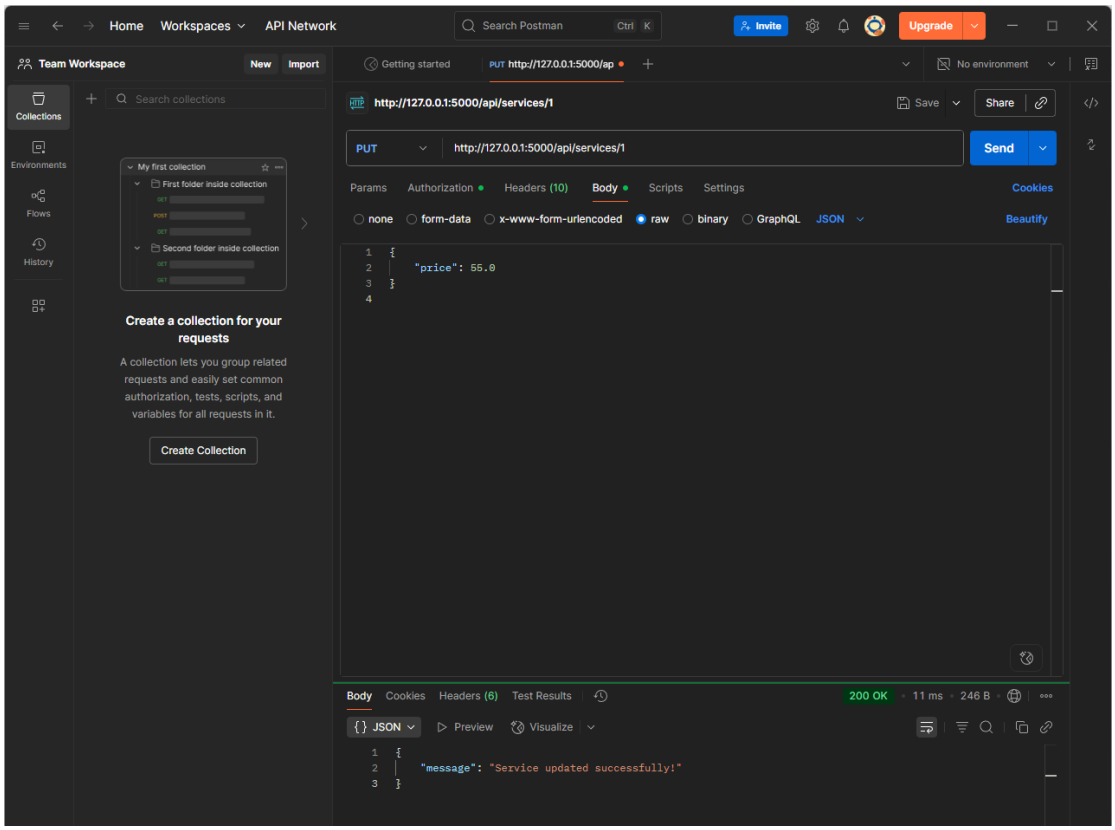


Log in the trainer account



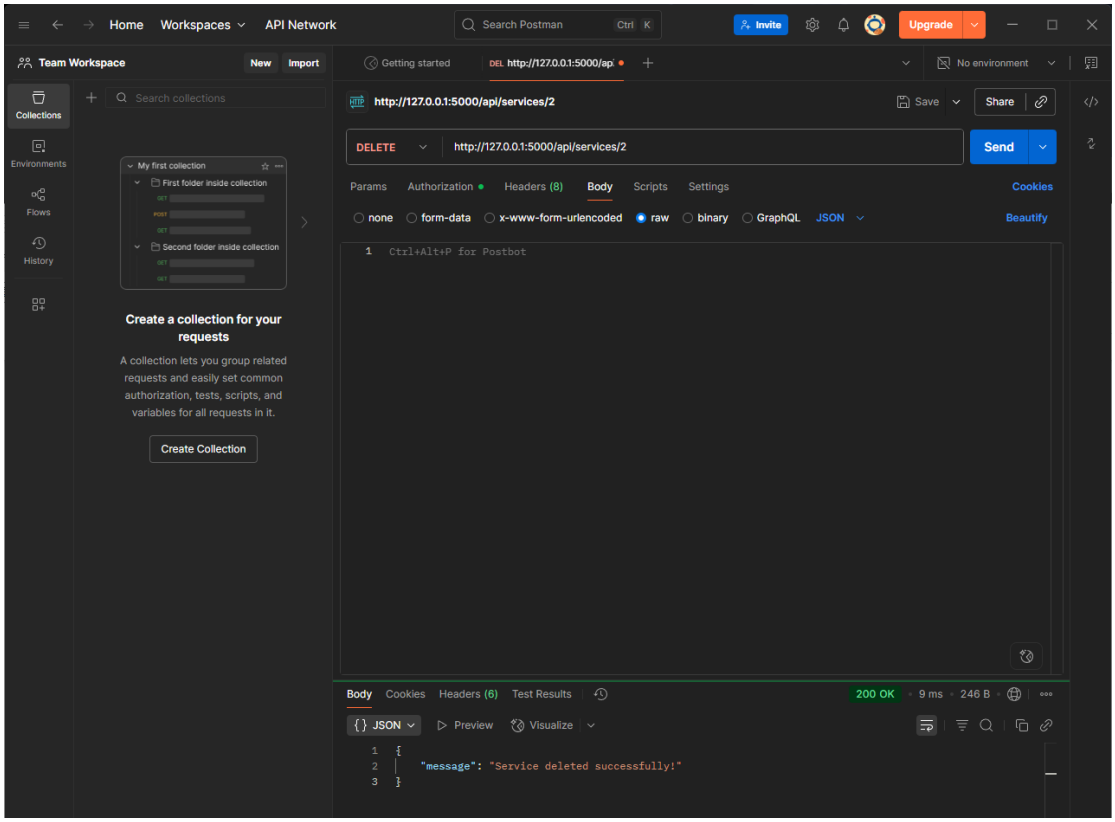
Publish the services provided by the coach

- **PUT /api/services/:id - Update Service (Trainer Login Required)**
    - **Functionality**: Allows a trainer to modify the information of a published service.
    - **Frontend Sends**: The unique ID of the service and the information to be updated.
    - **Backend Operation**: Updates the corresponding service record in the database.
    - **Returns**: The successfully updated service information.
- **Testing**



Use “PUT” and service id “1” to update the service provided by trainer

- **DELETE /api/services/:id - Delete Service (Trainer Login Required)**
    - **Functionality**: Allows a trainer to remove a service item.
    - **Frontend Sends**: The unique ID of the service.
    - **Backend Operation**: Deletes the corresponding service record from the database.
    - **Returns**: A confirmation message of successful deletion.
- **Testing**

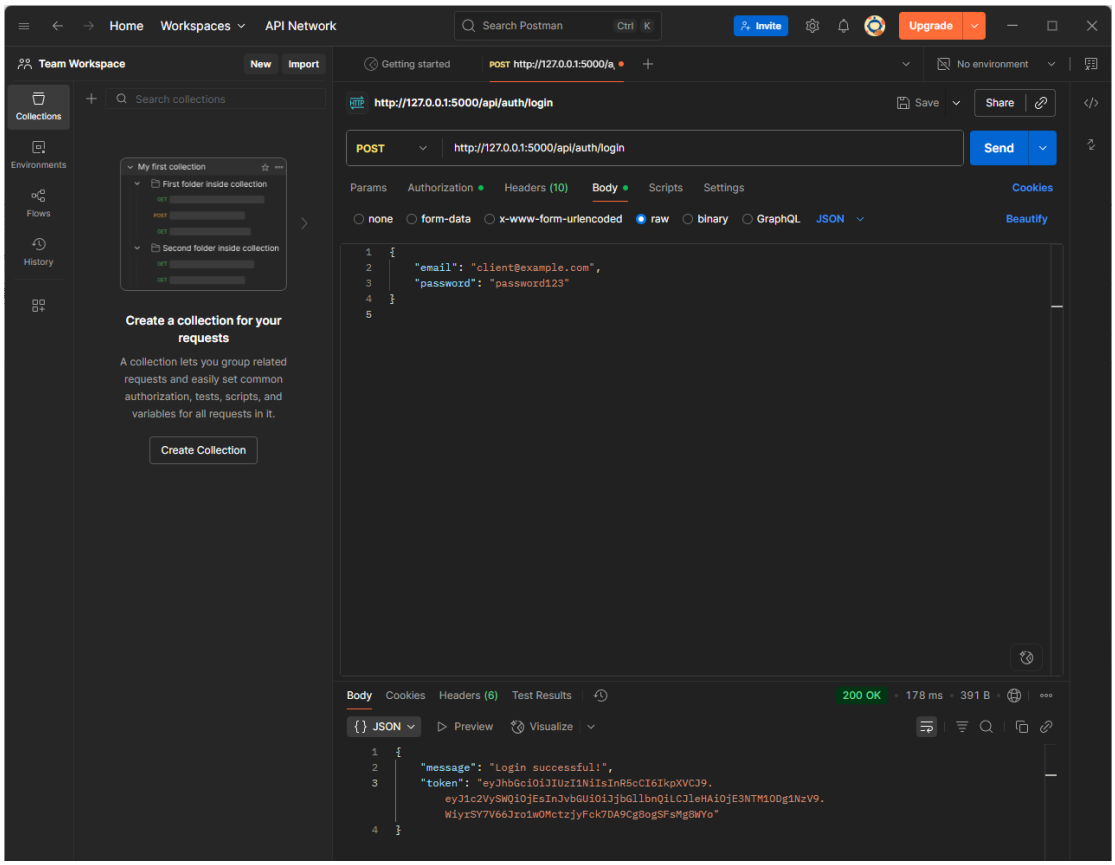


Use “DELETE” to delete the service we create with the service id “2”

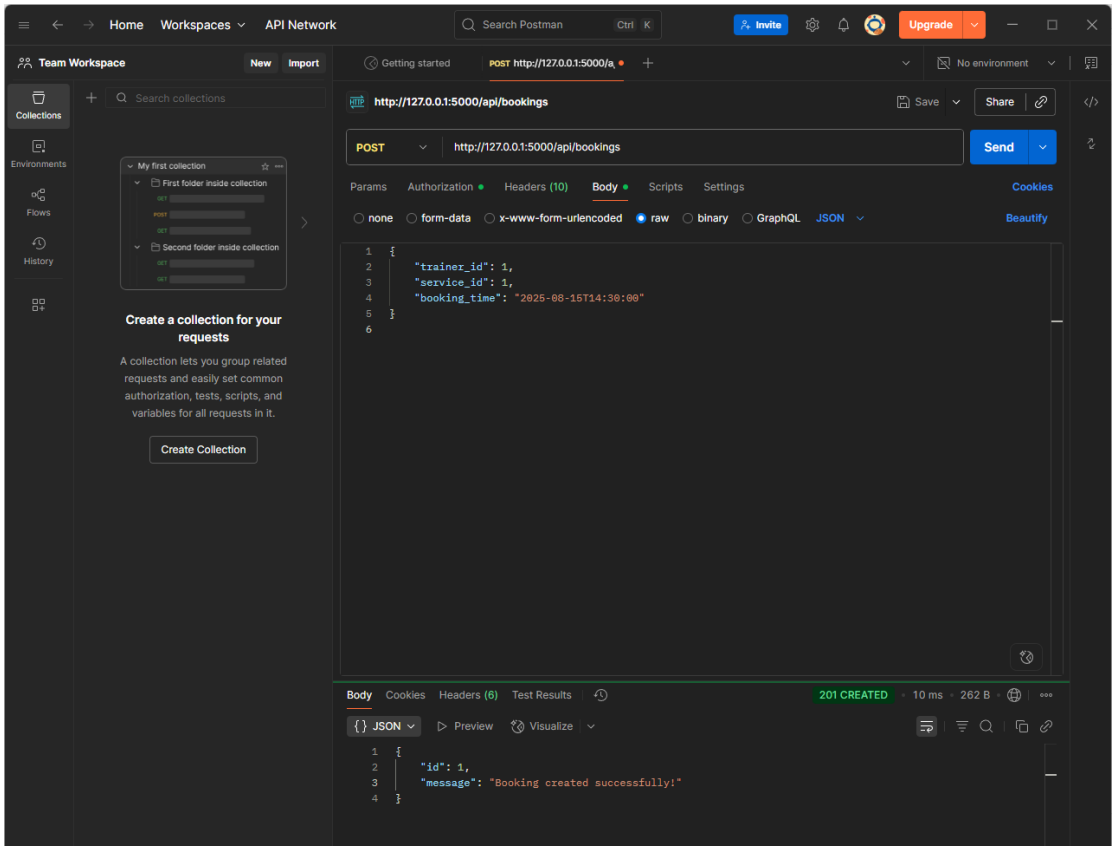
4. Bookings APIs

This is the core functionality connecting clients and trainers, responsible for handling session bookings.

- **POST /api/bookings - Create New Booking (Client Login Required)**
    - **Functionality**: Allows a logged-in client to book a specific service from a trainer at a particular time.
    - **Frontend Sends**: The trainer's ID, the service ID, and the selected date and time.
    - **Backend Operation**: Checks if the time slot is available, then creates a new booking record, associating it with the client and trainer.
    - **Returns**: The details of the successful booking.
- **Testing**

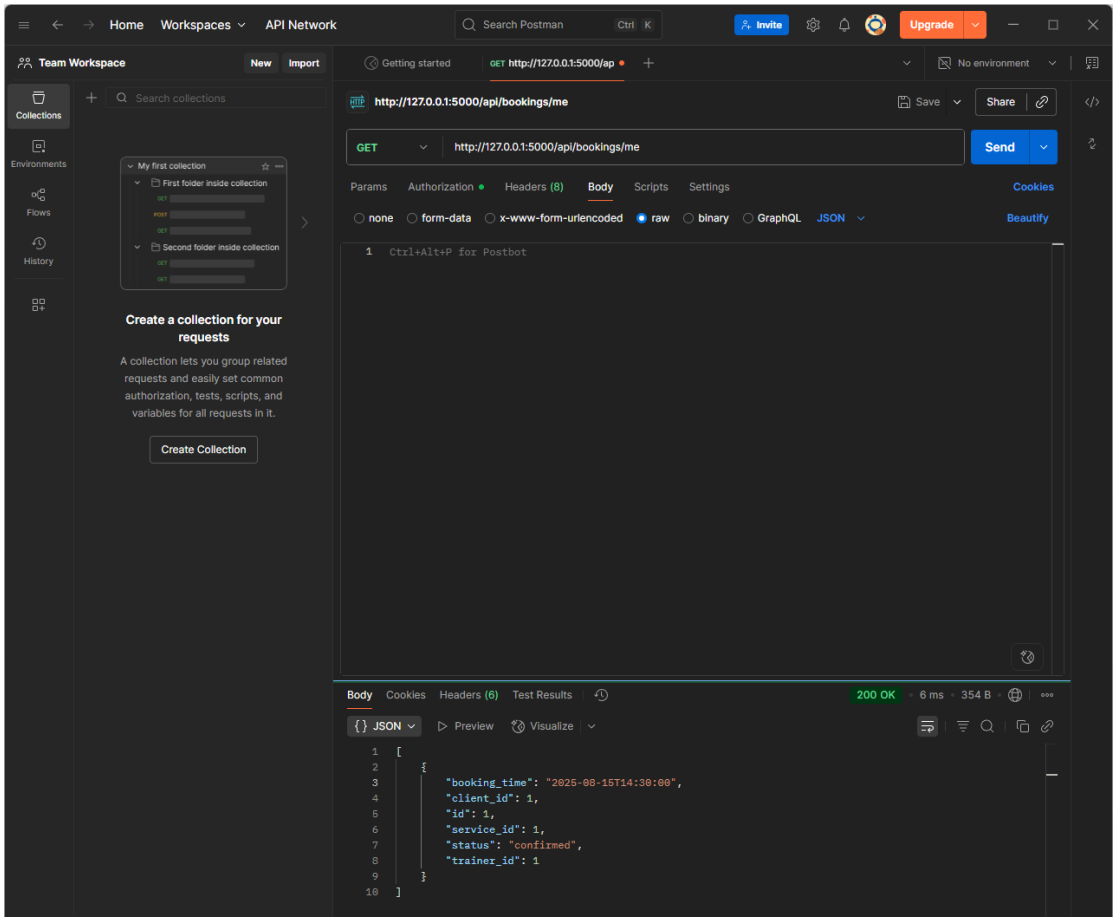


Log in the user account we create first



Use “POST” to create a new booking

- **GET /api/bookings/me - Get My Bookings (Login Required)**
    - **Functionality**: Allows a logged-in user (either a client or a trainer) to view all their related booking records.
    - **Frontend Sends**: Nothing.
    - **Backend Operation**: Queries for all upcoming and completed bookings associated with the user's identity.
    - **Returns**: A list containing all booking information for that user.
- **Testing**



When the users log in, they can find their booking with “GET”