House trading REST API

IT325 Web Services Final Project

by

Abdelmalek Bettaieb

January 2023

BA/IT Junior Student

Professor Montassar Ben Messaoud

Tunis Business School

Ben Arous,TUNISIA.

2022-2023

Contents

1 Introduction **2**

2 Technical overview **3**

2.1 Language and tools 3

2.1.1 Javascripts and nodejs 3

2.1.2 NestJS and MVC 3

2.2 Security and JWT 4

2.2.1 JWT 4

2.2.2 PassportJS 4

2.3 Database: PostgreSQL 5

2.4 ORM and databasee integration 5

2.4.1 Entity definition example: House 6

2.5 Tables schema 7

2.6 Rest API routes 8

2.6.1 User authentication 8

2.6.2 User management 10

2.6.3 House management 11

3 Conclusion **15**

Chapter 1

Introduction

This Project is made for the IT325 Web service course At the core of the project, I have employed the Node.js alongside NestJS framework for server-side scripting, PostgreSQL for data storage, TypeORM for database management, and RESTful API for seamless communication between the client and server. Furthermore, we have implemented JSON Web Tokens (JWT) for secure authentication and authorization,

In the following chapters you will find multiple usecase of authentification , CRUD operations , MVC pattern and data validation.

The main idea is to create a showcase project that represents a backend for a company that posts houses for sale

You will find in the following chapters examples of every CRUD operation, JWT authentication and authorization, MVC design pattern examples, Database relationships, Entities Validation logic and code snippets to explain some tricky parts.

Chapter 2

Technical overview

2.1 Language and tools

2.1.1 Javascripts and nodejs

Node.js and JavaScript are good for backend development because they have several key features that make them well-suited for server-side programming. Firstly, Node.js is built on top of the V8 JavaScript engine which was developed by Google. This engine is known for its high performance, providing Node.js with the ability to execute code quickly and efficiently. This is important for backend applications that need to handle a large number of requests or process large amounts of data. Additionally, Node.js provides a rich set of built-in modules and APIs, which makes it easy to perform common tasks such as reading and writing files, connecting to a database, and handling HTTP requests.

Another advantage of using JavaScript for backend development is that it is a widely-used and well-known language. This means that it is easy for developers to find resources and tutorials to help them learn, and it also makes it easier to find developers with experience in JavaScript.

2.1.2 NestJS and MVC

NestJS is a framework for building efficient, scalable Node.js server-side applications. It is built on top of the popular Express.js framework and uses a modular architecture that makes it easy to organize and structure code. One of the key features of NestJS is its use of the Model-View-Controller (MVC) pattern, which is a common architectural pattern used in web development.

In MVC, the Model represents the data and the business logic of the application. The View represents the user interface and the way the data is presented to the user. The Controller handles the communication between the Model and the View and handles user input. NestJS uses the MVC pattern to separate the different concerns of an application, making it more organized and easier to maintain.

NestJS also has a powerful module system that allows you to group related functionality together and easily reuse code across your application. This makes it easier to manage dependencies, and also allows you to easily test and debug your code.

2.2 Security and JWT

2.2.1 JWT

JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. JWT is often used for authentication and authorization purposes, as the claims encoded within a JWT can be trusted to be both authentic and valid. They are often used to authenticate users and pass information about the user between systems or services

2.2.2 PassportJS

Passport.js is a popular authentication middleware for Node.js that allows you to use a variety of authentication strategies, including JSON Web Tokens (JWT). To use JWT with Passport, you will need to install the passport-jwt module and configure it with your JWT secret and the options for extracting the JWT from the request.

By using the passport-jwt module, you can authenticate a user by extracting the JWT from the request, and then use the information from the JWT payload to find the user in the database. Once the user is found, you can then pass the user information to the route handler or use it to grant access to protected routes.

Passport.js provides a simple and flexible way to handle authentication in your application, and the passport-jwt module makes it easy to integrate JWT-based authentication into your application. By using Passport, you can authenticate requests in a consistent way across your application, making it more secure and easier to manage.

2.3 Database: PostgreSQL

PostgreSQL is a powerful, open-source, object-relational database management system (ORDBMS). It is known for its stability, data integrity, and concurrency control. It supports a wide variety of data types and includes powerful features such as full-text search, advanced indexing, and support for stored procedures and triggers. It is often used in high-load and enterprise environments, and is also a popular choice for web and mobile application development, data warehousing, and business intelligence. It's also supports SQL standards and has an active and vibrant community.

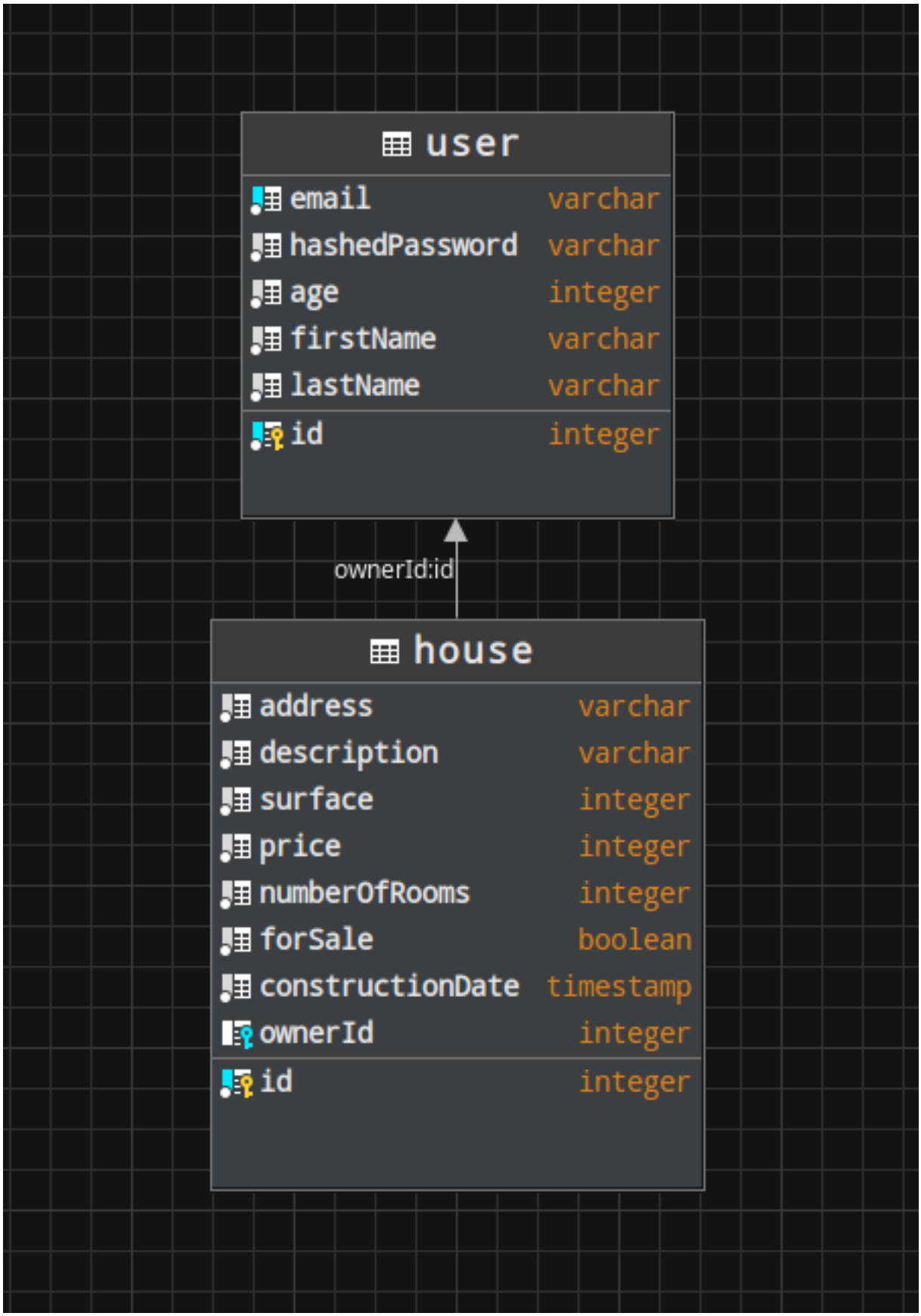
2.4 ORM and databasee integration

ORM (Object-Relational Mapping) is a technique that maps a relational database to an object-oriented programming language and allows developers to perform database operations using an object-oriented API instead of writing raw SQL. It makes the code more readable, maintainable and reduces the risk of SQL injection attacks. It's widely used in many programming languages and frameworks to facilitate the interaction with databases. This project uses TypeORM which is the one of the most popular Javascript ORMs

2.4.1 Entity definition example: House

```
You, 21 hours ago | 1 author (You)
@Entity()
export class House {
  @PrimaryGeneratedColumn()
  public id: string;

  @Column()
  @IsString()
  @MaxLength(100)      You, 21 hours ago • id adressing …
  @MinLength(3)
  public address: string;

  @Column()
  @IsString()
  @MinLength(3)
  @MaxLength(100)
  description: string;

  @Column()
  @IsNumber()
  surface: number;

  @Column()
  @IsNumber()
  price: number;

  @Column()
  @IsNumber()
  numberOfRooms: number;

  @Column()
  @IsBoolean()
  public forSale?: boolean;

  @Column()
  @IsDateString()
  public constructionDate?: Date;

  @ManyToOne(() => User, (user) => user.ownerHouses)
  public owner?: User;
}
```

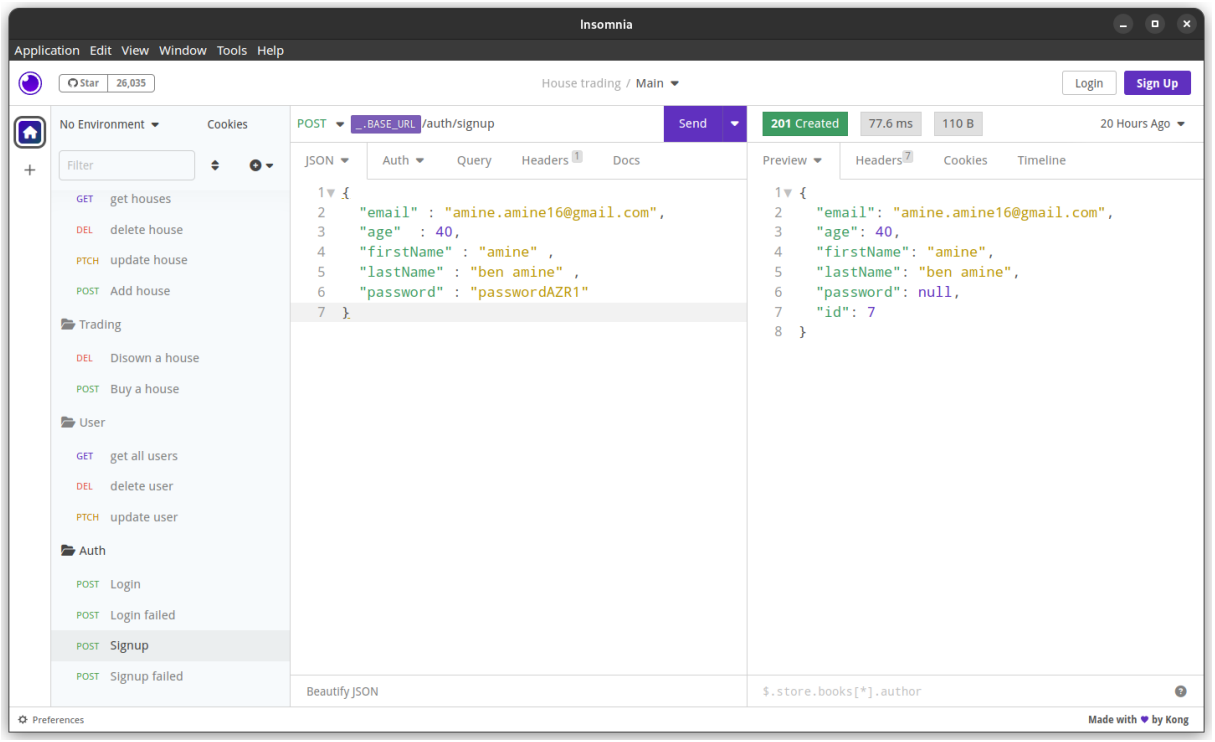
2.5 Tables schema



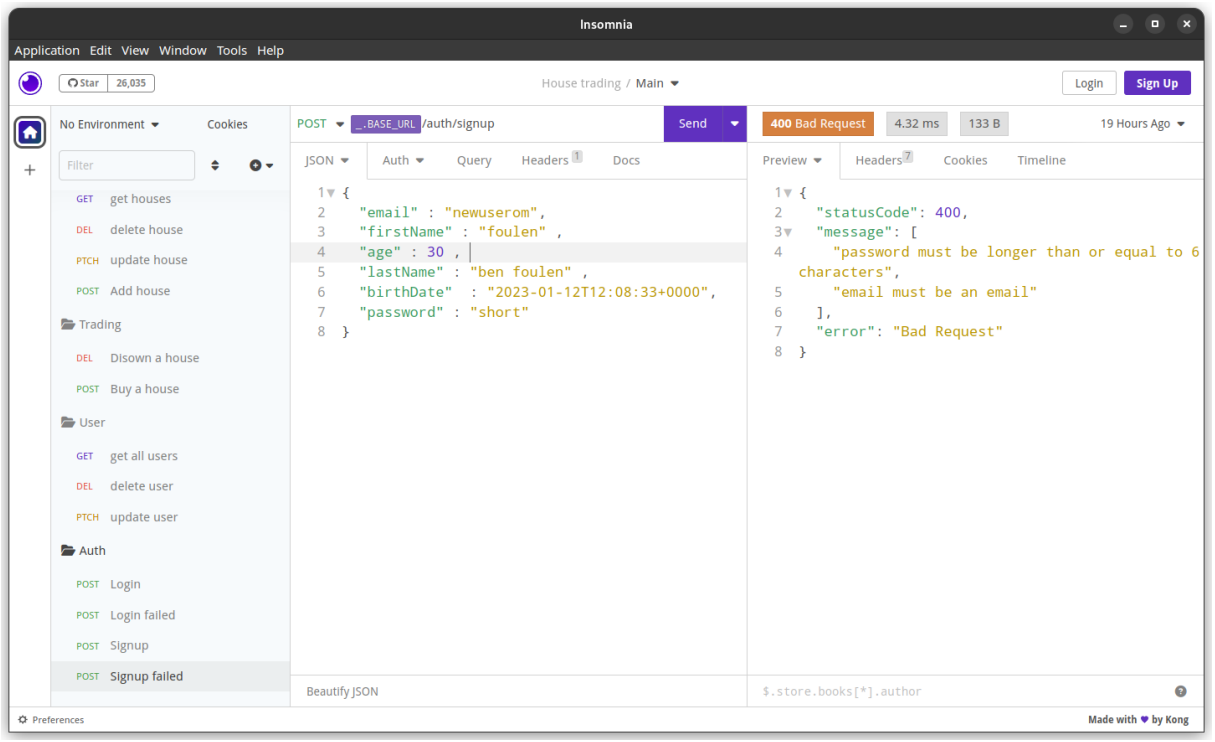
2.6 Rest API routes

2.6.1 User authentication

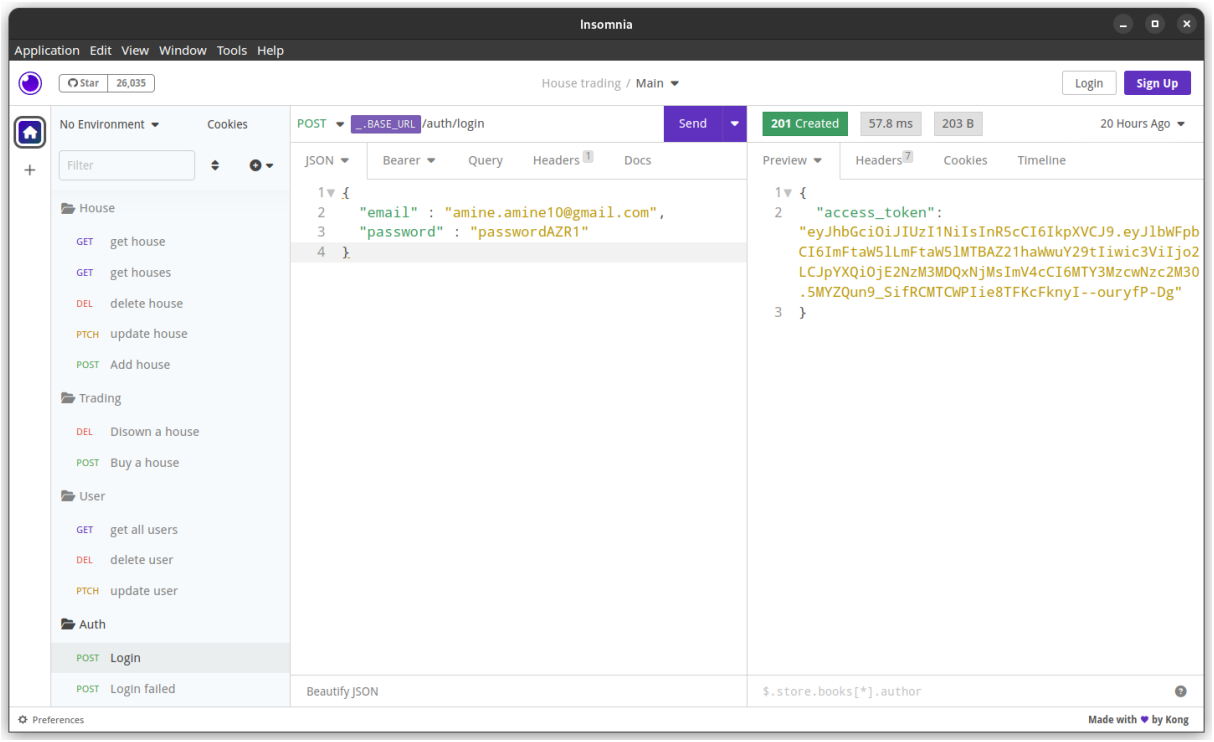
Sign-up



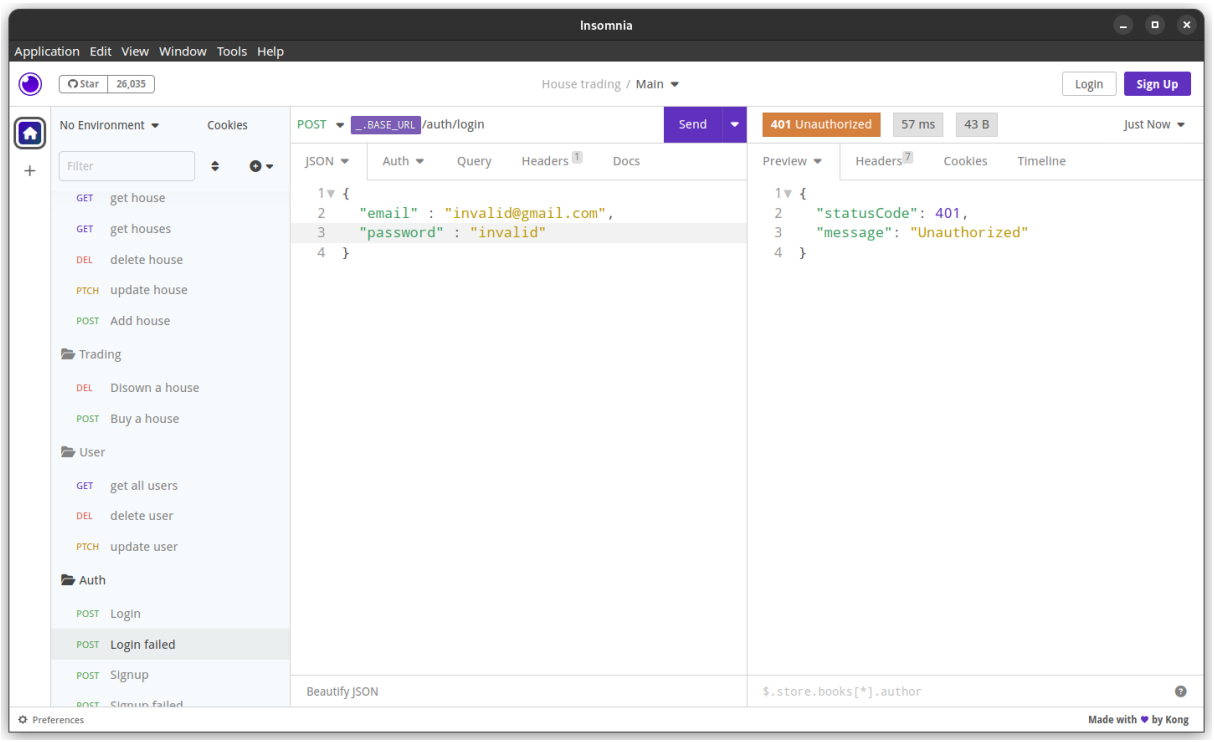
Sign-up with error



Login

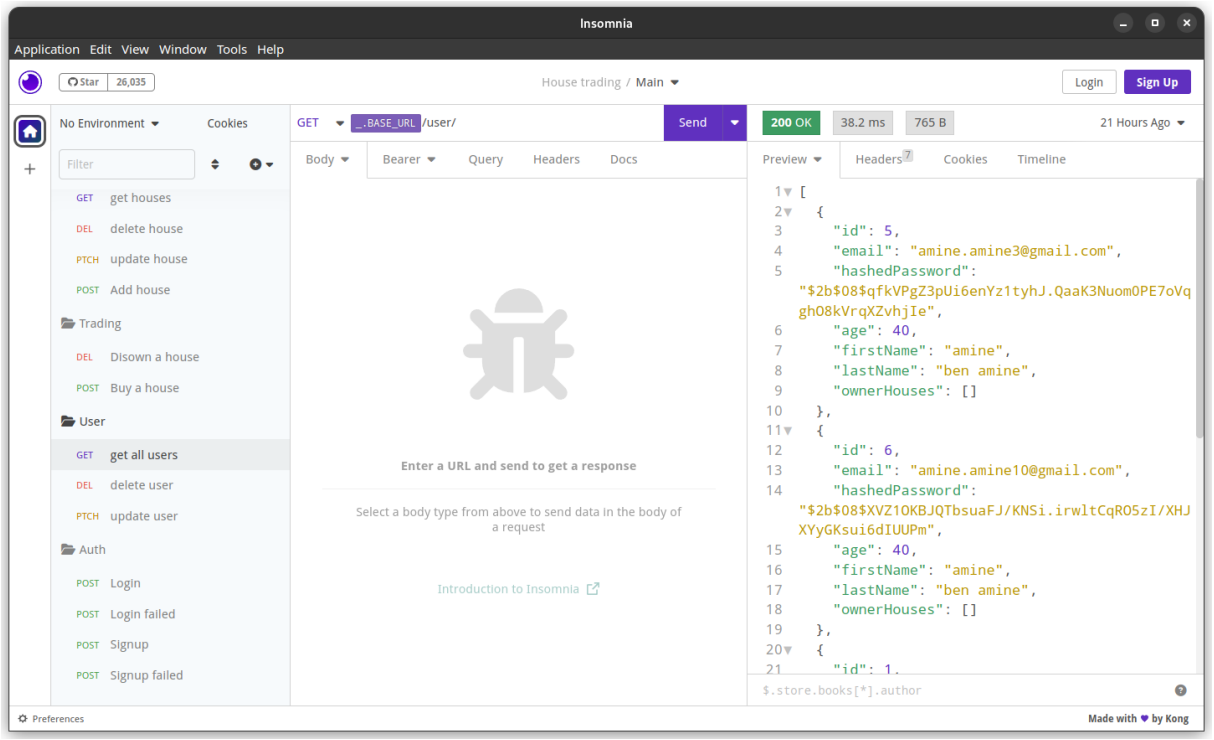


Login with error

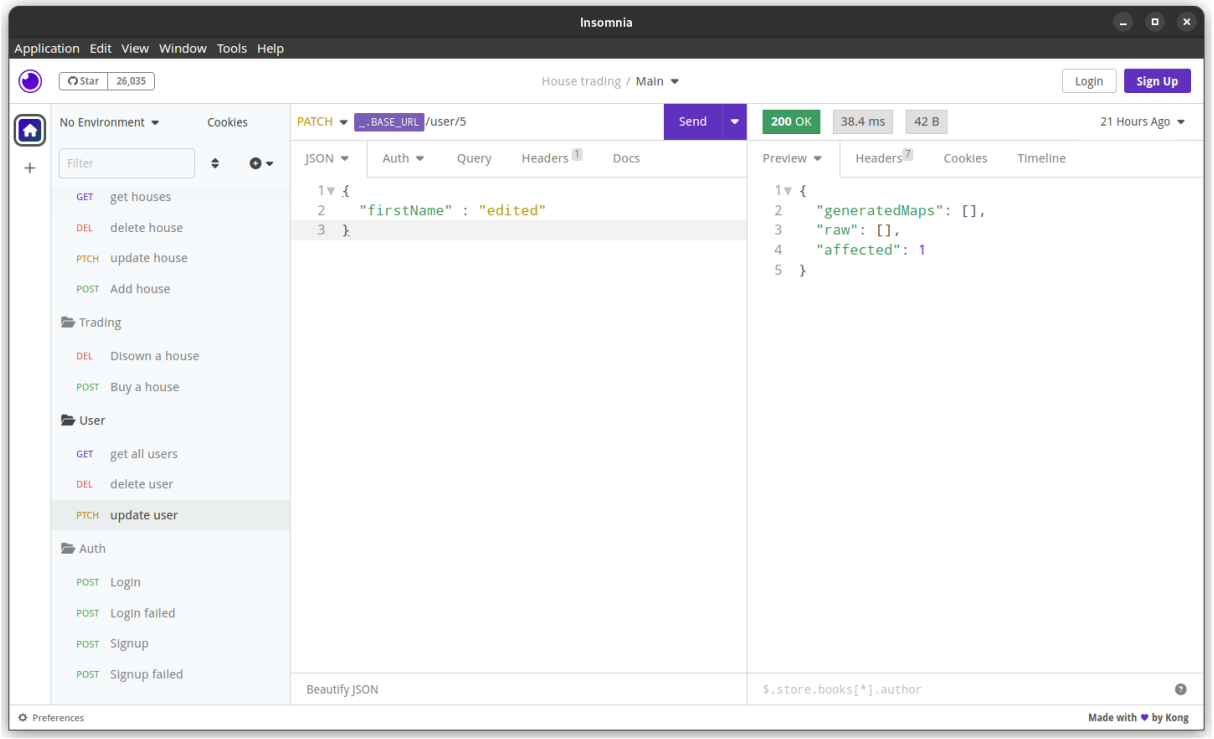


2.6.2 User management

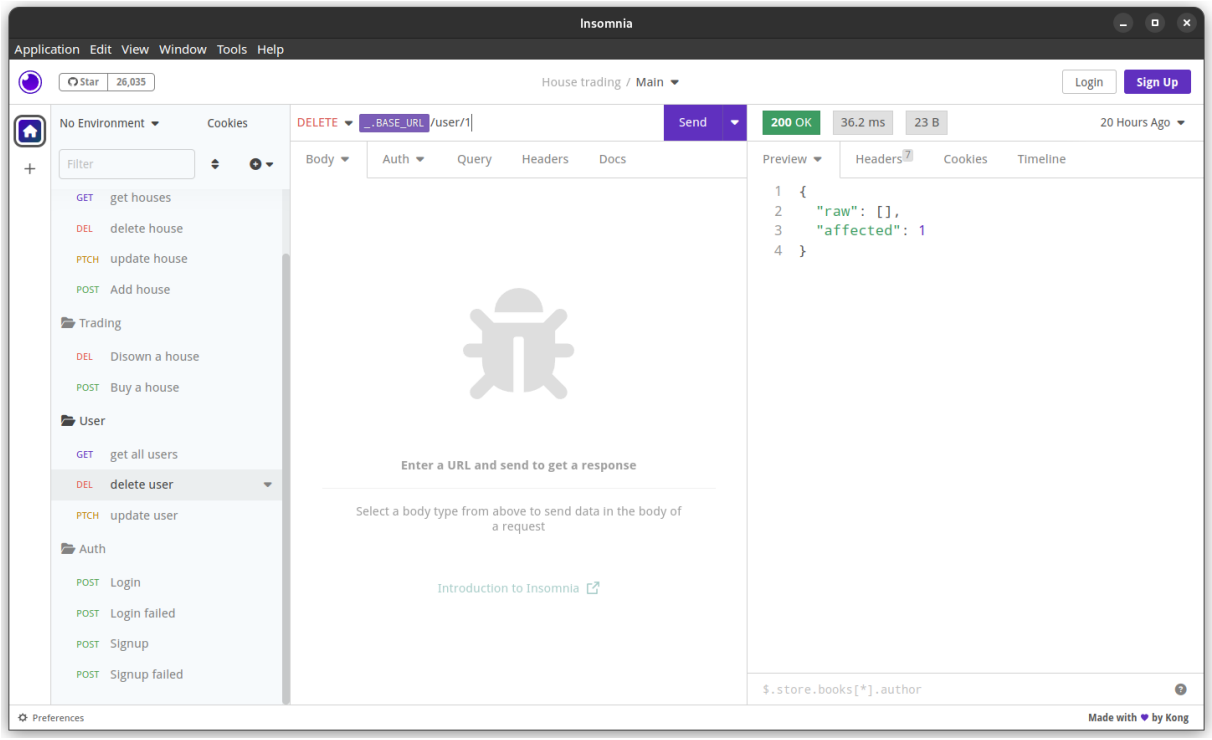
List all users



Edit User

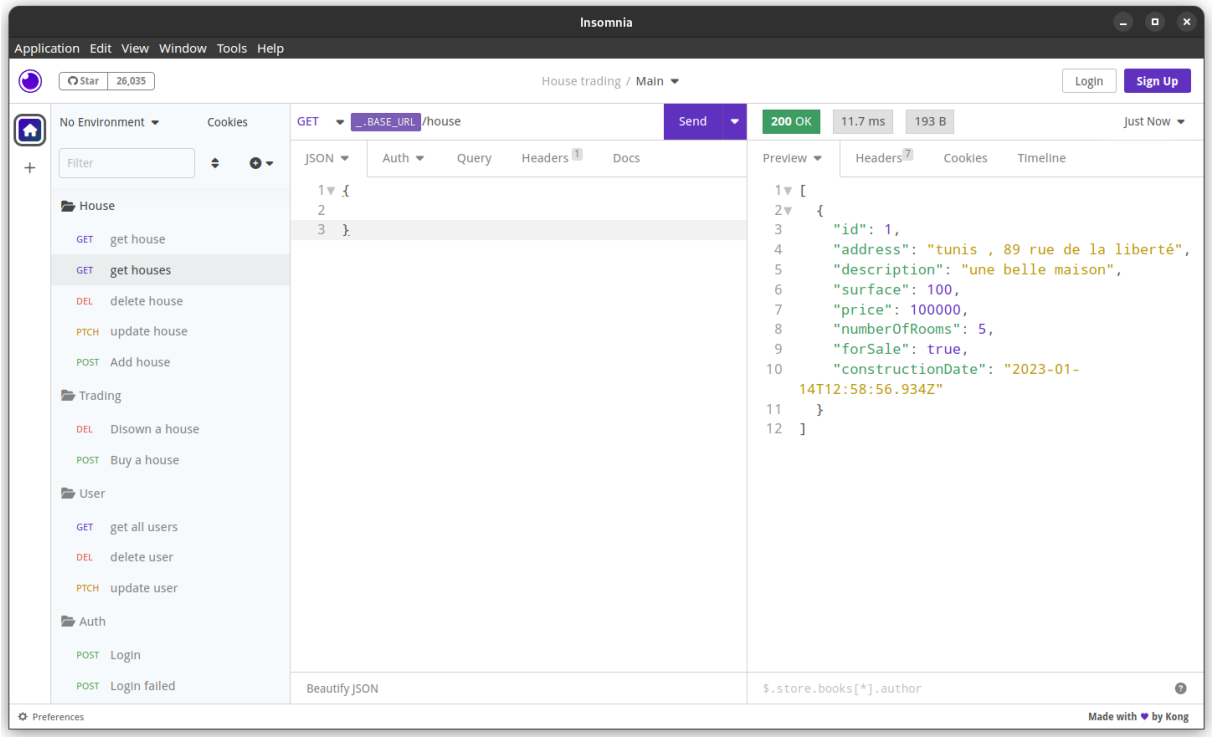


Delete User

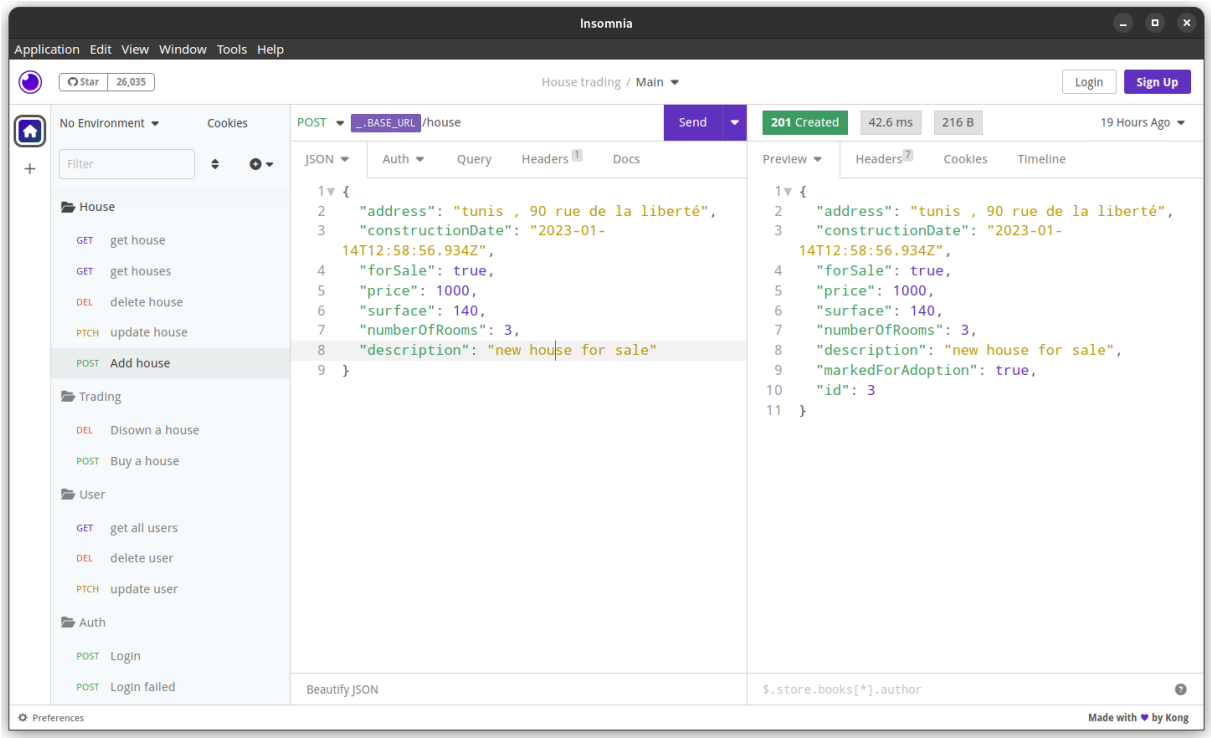


2.6.3 House management

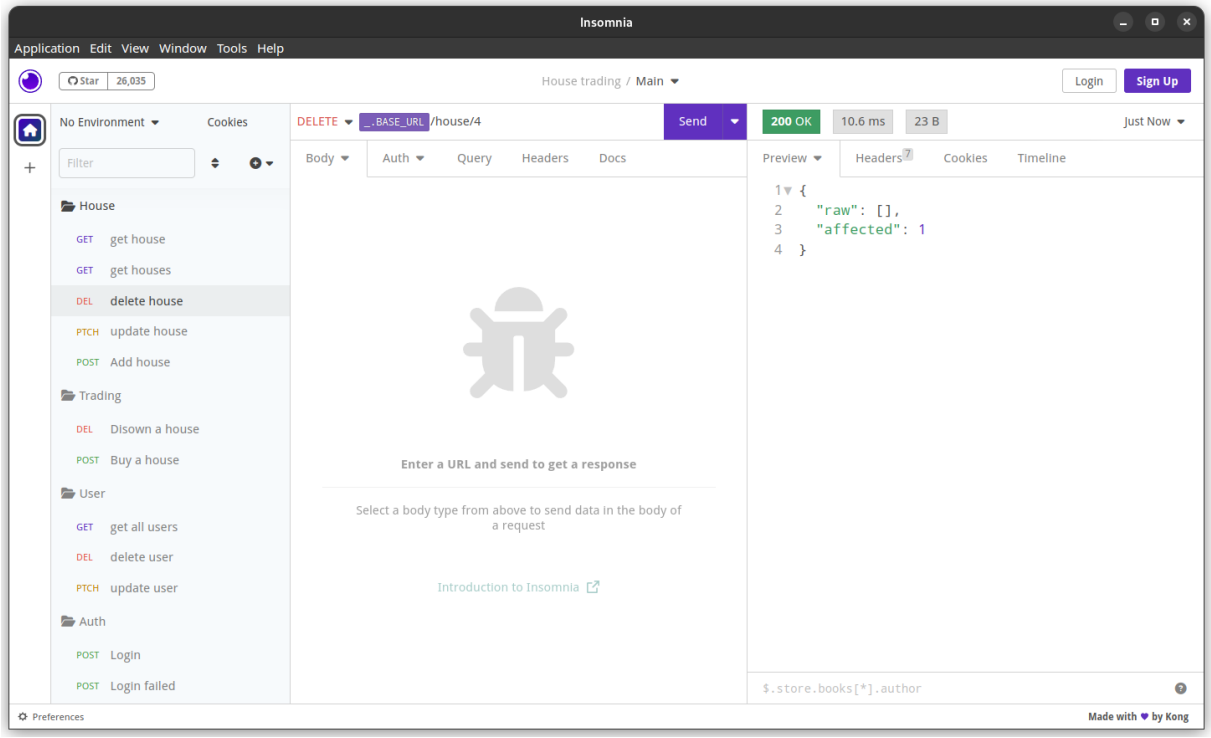
List all houses



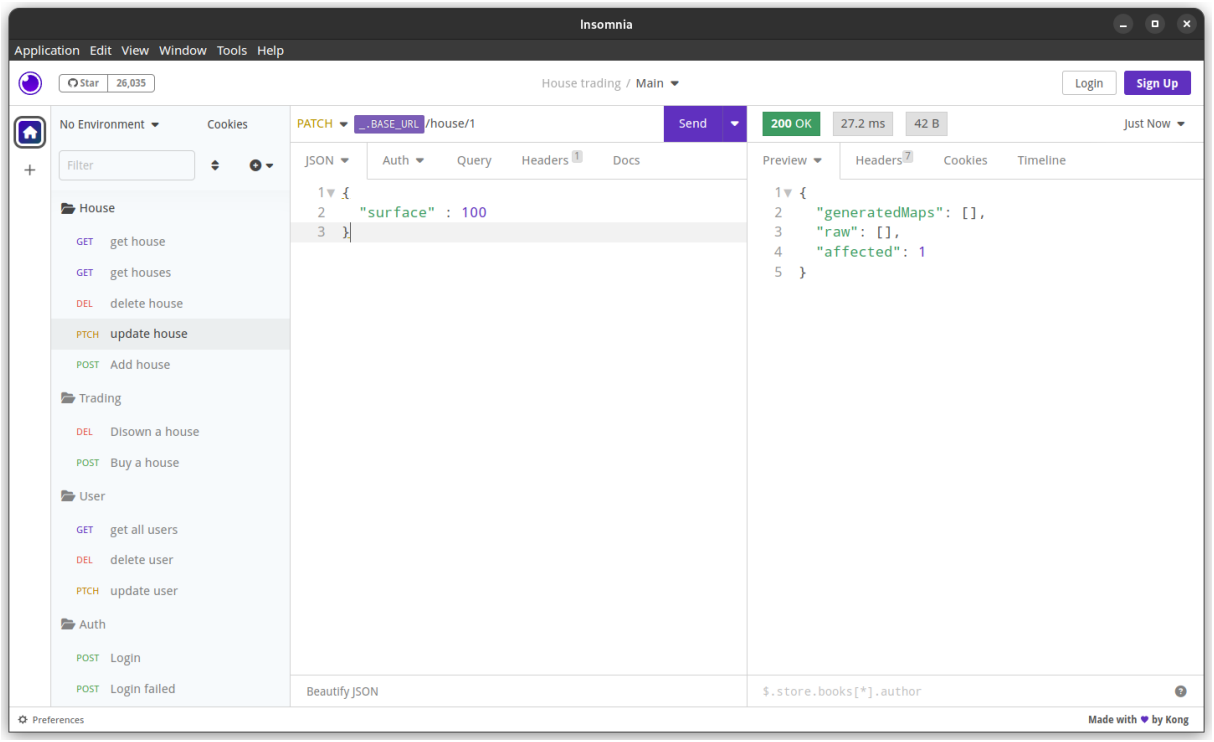
Add a house



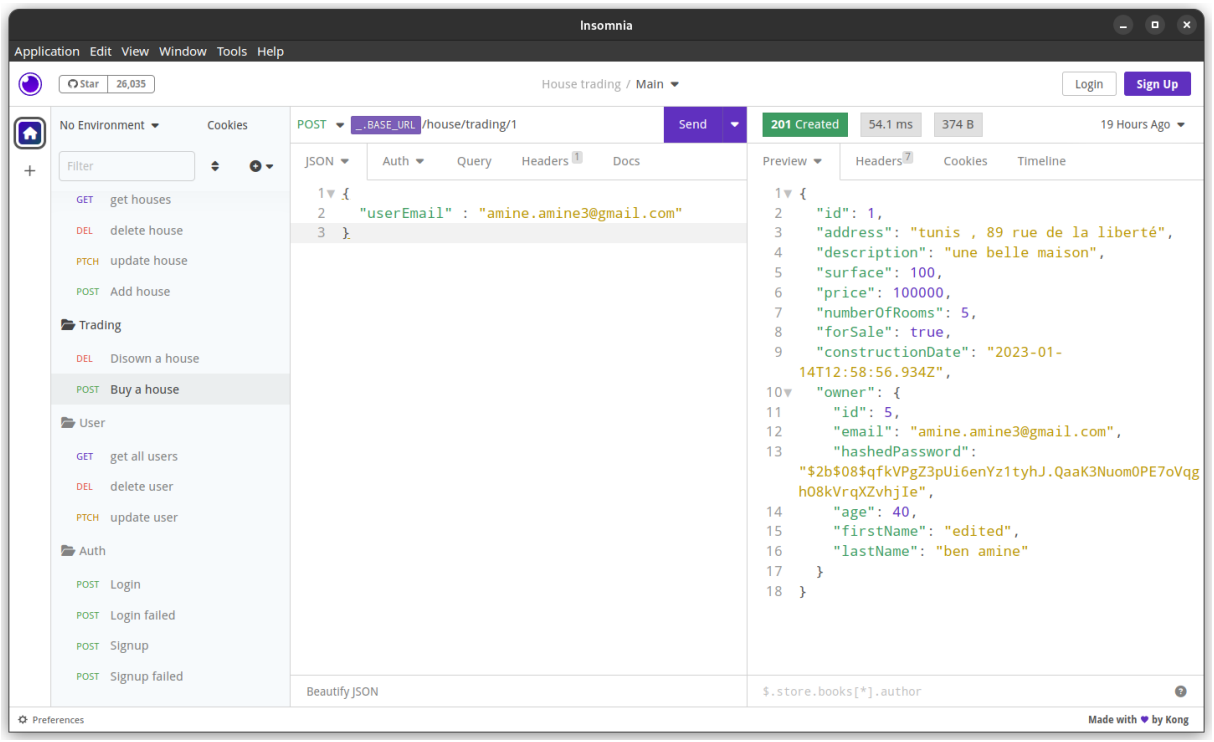
Delete house



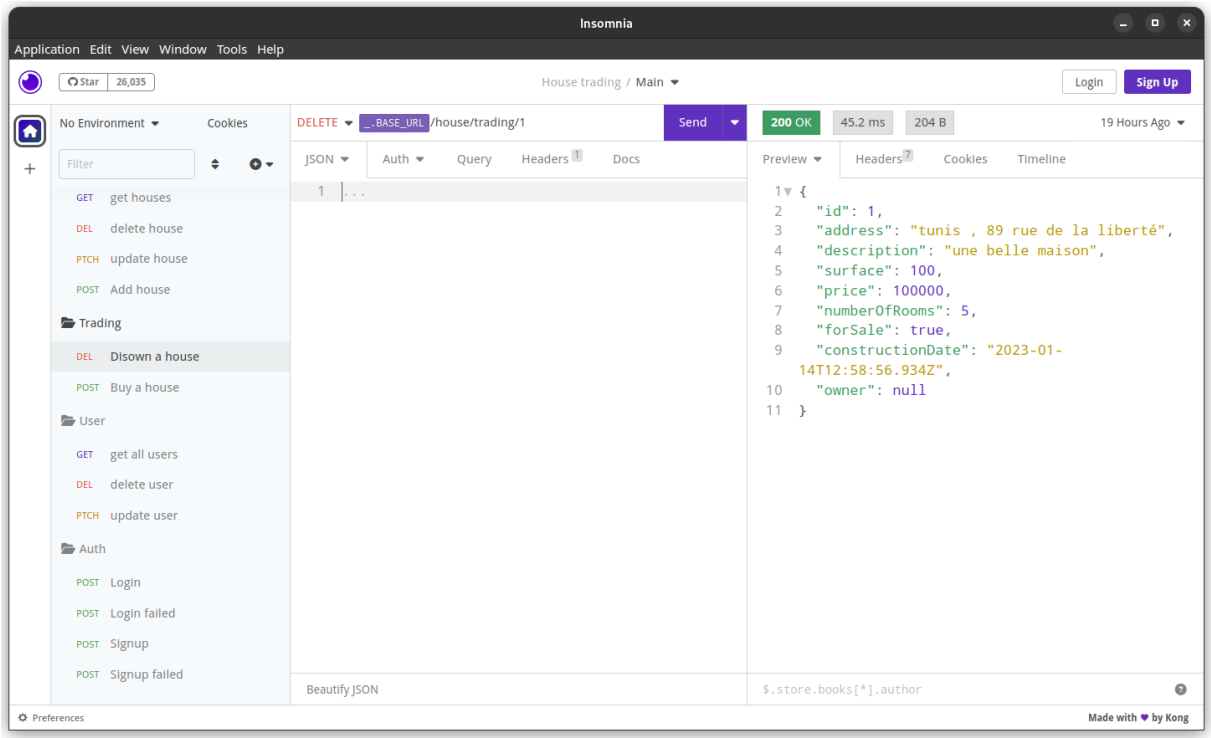
Update house



Buy a house



Disown a house



Chapter 3

Conclusion

In addition to learning Node.js, Nest.js, MVC, JWT, MongoDB and Mongoose, this demo project also provided an opportunity for me to learn about developing RESTful APIs. REST (Representational State Transfer) is a widely used architectural style for building web services. I learned about the various HTTP methods and how to use them properly in the context of RESTful APIs. I also learned about the importance of endpoints, request and response formats, and how to handle errors in a RESTful API.

Furthermore, I learned how to implement authentication and authorization using JWT tokens in RESTful APIs, it was a great experience to understand how to secure our endpoints and how to handle different roles and permissions.

Overall, this showcase project was a great learning opportunity for me to understand how to design, develop, and deploy a RESTful API with Node.js, Nest.js, MVC, JWT and postgreSQL.

It allowed me to gain a deeper understanding of how to build scalable, maintainable, and secure APIs that are easy to consume by other applications. This knowledge will be extremely valuable as I continue to develop web services and other backend applications in the future.