TUNIS BUSINESS SCHOOL

Web Service Project Report
TunInvest

Student:
Nada JOUIBLI

Professor:
Montassar BEN MESSAOUD
Course:
IT325: WEB SERVICES

January, 2023.

Abstract

TunInvest is an API that was developed to help Tunisians report on the problems they face in their communities. The API enables decision-makers, such as investors, the government, and entrepreneurs, to see and track these issues in order to address them more effectively. The API has the potential to significantly improve the quality of life in Tunisian communities by providing a platform for individuals to voice their concerns and for decision-makers to take action. Through a detailed analysis of the API's design and implementation, this report demonstrates the API's potential to facilitate communication and problem-solving within communities.

Keywords: API, Community development, Quality of life, Collaboration, Problem-solving.

Contents

Chapter 1

Introduction

The TunInvest API is designed to allow Tunisian citizens to report issues, needs, and potential enhancements in their local communities, as well as provide a means for decision-makers to identify and address these issues. This API was developed using FastAPI and Python and utilizes a PostgreSQL database managed with SQLAlchemy and Alembic. Security measures include OAuth2 JWT and password hashing for user authentication. Testing was conducted using Postman, and version control was managed with Git and GitHub. The API was containerized using Docker and an attempt was made to deploy the API using Render. The goal of this API is to improve the quality of life for Tunisian citizens and contribute to the development of the country through better communication and understanding of community issues.

1.1 Motivations

The motivation behind creating TunInvest is to address several issues faced by Tunisian citizens and decision-makers in the country.

- Difficulty in identifying and addressing issues faced by citizens at a local level.
- Inefficiency in communication between citizens and decision-makers.
- Lack of transparency and accountability in the decision-making process.
- Difficulty in gathering and analyzing data on community needs and problems.
- Lack of coordination and collaboration between different sectors and stakeholders.
- Ineffective use of resources and funds due to a lack of understanding of local needs and priorities.
- High unemployment and inflation rates.

1.2 Objectives

The main objective of TunInvest is to facilitate the development of targeted and effective solutions to the challenges facing Tunisian communities.

- Citizens will be able to post about the issues they face in their local communities, including the type of challenge, sector, and location.
- Decision-makers will be able to view all issues posted by citizens and choose to focus on a specific issue or project to address.

Chapter 2

Design And Implementation

In this section, we will delve into the details of the design and implementation of the TunInvest API. This includes the choice of technologies, the database design, the security measures, and the testing and deployment processes.

2.1 Architecture And Technology

2.1.1 Users

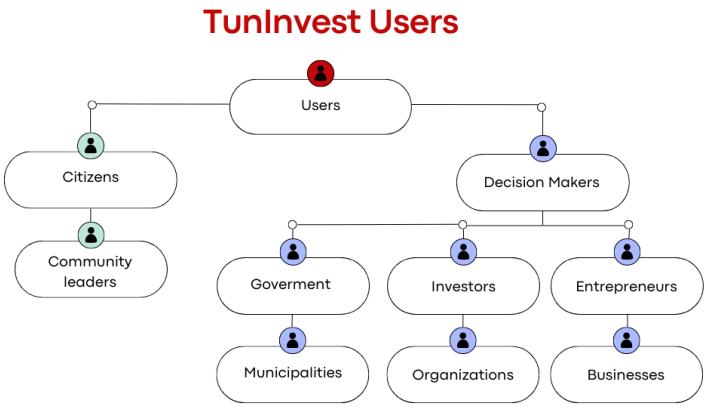

Figure 2.1: TunInvest Users

Figure 2.1 shows the different users of this TunInvest API. On the one hand, TunInvest is designed to be used by all Tunisian citizens. It will allow them to post any issues, problems, and complaints within their local community. On the other hand, it aims to provide all stakeholders whether they were Tunisian entrepreneurs or international investors with detailed and accurate information about the troubles Tunisian communities face with the aim to inspire them to create businesses that focus on these issues and help to solve the problems.

2.1.2 Architecture

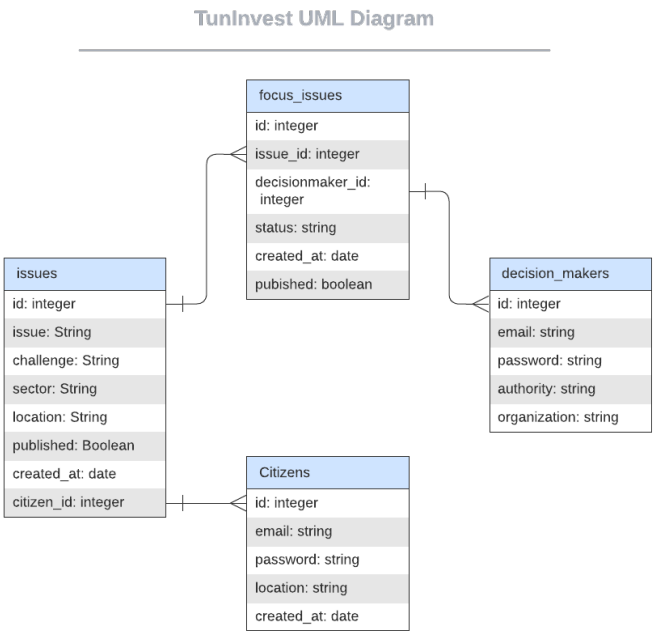

Figure 2.2: UML Diagram of the System Architecture

As shown in Figure 2.2, the UML diagram shows that the database is formed of 2 tables for the users: citizens, and decision-makers, and 2 others for the issues posted by the citizens and the focus issues selected to address and work on by the decision-makers.

2.1.3 Technology

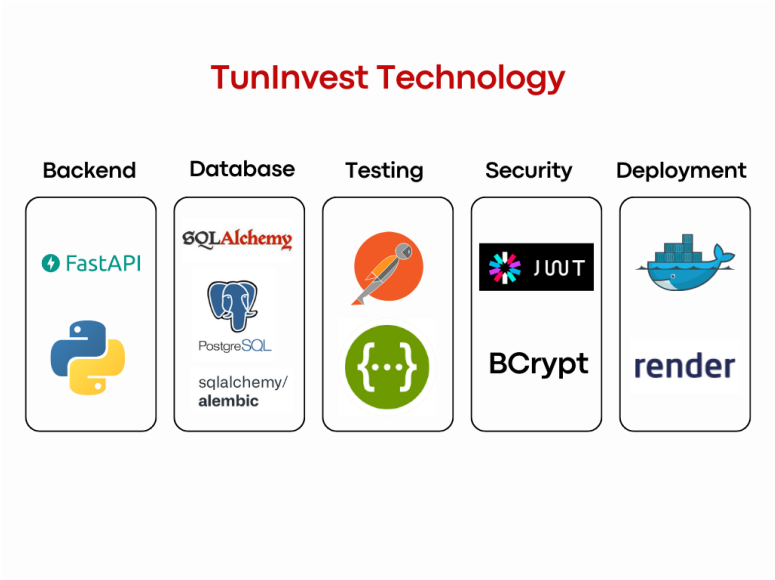

Figure 2.3: Implemented Technology

2.1.3.1 Fast API v0.88.0

FastAPI is a modern, fast (high-performance) web framework for building APIs with Python 3.6+. Some of its characteristics are fast, robust, and standard-based. It also supports asynchronous programming, which allows better performance and scalability. [14]

2.1.3.2 Postgres

Postgres is a free and open-source relational database management system (RDBMS). It is designed to handle various workloads and has a strong reputation for its reliability, data integrity, and correctness. [10]

2.1.3.3 SQLAlchemy v1.4.45

SQLAlchemy is a Python library that provides a set of high-level APIs for connecting to relational databases, in this case, Postgres. It provides a powerful and flexible Object-Relational Mapper (ORM) that allows direct interaction with databases without having to write raw SQL statements.[3]

2.1.3.4 Pydantic v1.10.2

Data validation and settings management using Python type annotations. It enforces type hints at runtime and provides user-friendly errors when data is invalid. It also defines how data should be in pure, canonical Python and validate it.[1]

2.1.3.5 Alembic v1.6.5

Alembic is a database migration tool for SQLAlchemy. It manages and applies changes to a database schema over time. It provides a simple and consistent way to version the database schema and to apply updates to it in a controlled and reversible manner.[2]

2.1.3.6 PyJWT v1.7.1

PyJWT is a Python library that allows you to encode and decode JSON Web Tokens (JWT). JWT is an open, industry standard for representing claims securely between two parties.[12]

2.1.3.7 Passlib v1.7.4

Passlib is a password hashing library for Python, which provides cross-platform implementations of over many password hashing algorithms like Bcrypt, which is a specific algorithm used for securely hashing passwords.[9]

2.1.3.8 Bcrypt v4.0.1

Bcrypt is a specific algorithm used by passlib for securely hashing passwords.[4]

2.1.3.9 Postman

Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.[11]

2.1.3.10 Git v2.38.1

Git is a distributed version control system that tracks changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.[7]

2.1.3.11 Github

GitHub is a web-based platform for version control and collaboration that uses Git, a distributed version control system.[8]

2.1.3.12 Docker v20.10.20

Docker is an open-source platform that enables developers to build, deploy, and manage containers—standardized that combine application source code with the operating system (OS) libraries required to run that code in any environment.[5]

2.1.3.13 Dockerhub

Docker Hub is a cloud-based registry service that allows you to store and distribute Docker images. It is the default registry for Docker and a hosted version of Docker Registry.[6]

2.1.3.14 Render

Render is a unified cloud to build and run all apps and websites with free TLS certificates, global CDN, private networks, and auto-deploys from Git.[13]

2.2 Features and functionalities

2.2.1 HTTP Methods

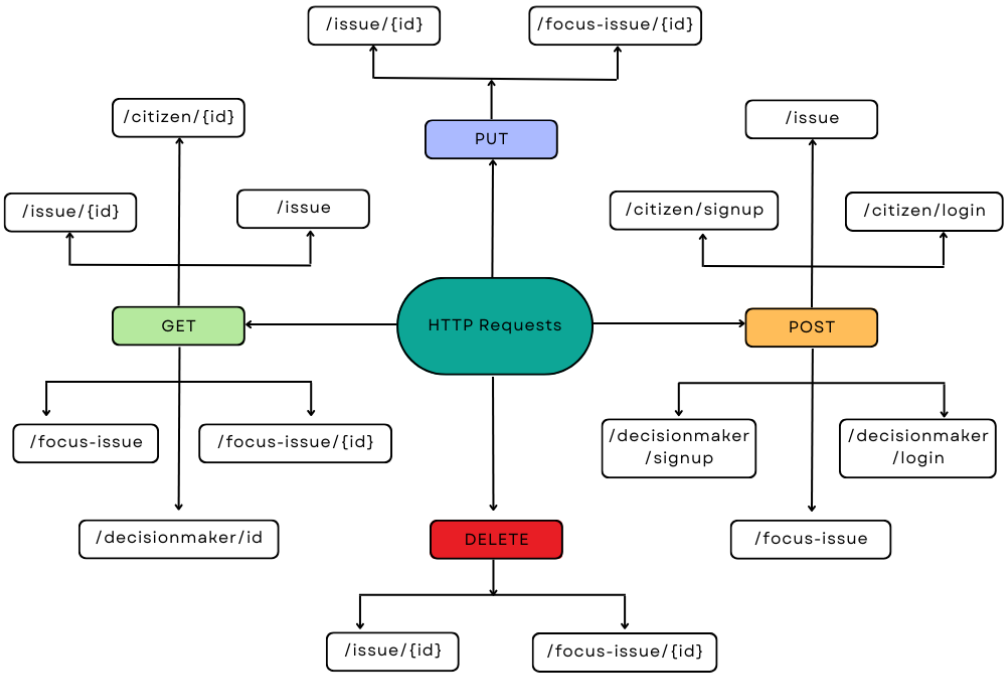

Figure 2.4: HTTP Methods

2.2.1.1 GET Requests

- GET /citizen/id
  Displays the profile of a citizen with information like their email and location.
- GET /decisionmaker/id
  Displays the profile of a decision maker with information regarding their organization, authority, and email.
- GET /issue
  Displays the list of issues posted by Citizens.
- GET /issue/id
  Displays a specific issue posted with information about the citizen.
- GET /focus-issue
  Displays the list of issues picked by decision-makers to work on.
- GET /focus-issue/id
  Displays a specific issue and shows its characteristics along with the identity of the citizen who posted it and the decision-maker who is working to solve it.

2.2.1.2 POST Requests

- POST /citizen/signup
  Used to sign up citizens.
- POST /citizen/login
  Used to login in citizens.
- POST /decisionmaker/signup
  Used to sign up decision-makers.
- POST /decisionmaker/login
  Used to login in decision-makers.
- POST /issue
  Used by citizens to post about issues. They have to enter the description, challenge, sector, and location.
- POST /focus-issue
  Used by decision-makers to post about the issues they chose to focus on and solve. They have to enter the issue id and the status of the progress of the project.

2.2.1.3 PUT Requests

- PUT /issue/id
  Allows citizens to update their issue posts.
- PUT /focus-issue/id
  Allows decision-makers to update the status of their projects.

2.2.1.4 DELETE Requests

- DELETE /issue/id
  Allows citizens to delete their issue posts.
- DELETE /focus-issue/id
  Allows decision-makers to delete their projects.

2.2.2 Security Measures

2.2.2.1 Input Validation

Using Pydantic and the Base Model, many schemas were defined to control and check that all input entered by users was in the correct format and does not contain any malicious content.
Some of the schemas used are:

- CreateCitizen, CreateDM: to control the signing up of users.
- IssueInfo, FocusInfo: to control the entered details about the issue and focus.

2.2.2.2 Encryption

Passwords entered by users are hashed using Passlib and Bcrypt. This ensures that even if a password is compromised, the attacker will not be able to use the actual password. Two functions were used:

- Hashing passwords when signing in.
- Verifying passwords when logging in.

2.2.2.3 Authentication

The ability to verify the identity of citizens and decision-makers was implemented to confirm that they are who they claim to be. Authentication was implemented through:

- A process to check if the user is logged in before allowing them to get or post information.
- A process to verify the identity of the user making a request.

2.2.2.4 Authorization

The ability to determine whether a user has access to specific resources or is able to perform certain actions. Authorization was implemented through:

- Only allowing signed-in users to access certain information or perform certain actions.
- Allowing users to only update and delete their own posts and not others.
- Using JSON Web Token (JWT) as a method of authorization including the use of both access tokens and refresh tokens.

Chapter 3

Evaluation

In this part, we will compare TunInvest to other solutions in the market and identify its strengths and weaknesses.

3.1 Market Analysis

After a thorough analysis of the market, these are the key takeaways that were identified:

- There is no direct competition with TunInvest API. No API in Tunisia can help you collect all information about issues faced within specific communities.
- However, there are possible ways for Tunisians to express their own concerns and for stakeholders to find out about these problems and address them accordingly as shown in Figure 3.1 and in Table 3.1.

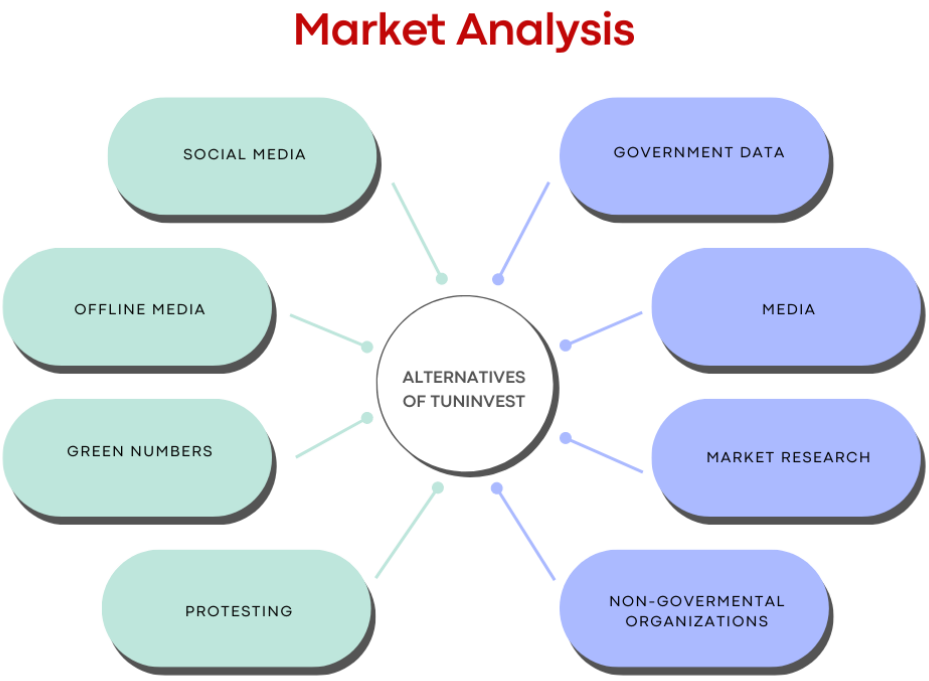

Figure 3.1: Alternatives to TunInvest API

Alternatives Used By Citizens	Alternatives used by Decision Makers
Social Media platforms: Tunisians use the power of social media to post about their concerns and problems. Some communities and specific cities do have Facebook groups where they exchange thoughts and ideas.	**Government Data:** Potential investors can reach out to government agencies and collect information about specific industries that require attention and investments.
Offline Media: TV shows, radio stations, and newspapers often interview and talk to people about certain trends and difficulties faced by certain communities.	**Media:** Online and offline media can be a source to learn about the struggles faced by Tunisians and can inspire future entrepreneurs to come up with innovative ideas to respond to these needs.
Green Numbers: Many institutions set free numbers to call and reach out to complain about poor service, policies, safety concerns, etc.	**Market Research:** Stakeholders can create surveys and polls to understand the need of Tunisians and identify potential profitable investments.
Protesting: Some communities in Tunisia opt for protests, strikes, and boycotts in order to express their dissatisfaction or opposition to a particular issue.	**Non-governmental organizations:** Most organizations work to respond to local needs and problems. Decision makers may gather information from them.

Table 3.1: Alternatives of TunInvest API

3.2 Strengths and Limitations

Based on the market research, TunInvest has many strengths and weaknesses illustrated in Table 3.2

Strengths	Limitations
Provides real-time data.	Requires citizens' engagement.
Allows centralized data collection.	Data quality depends on citizens' input.
Captures direct community concerns with no middleman and censorship.	Requires resources for maintenance and development.
Facilitates communication and collaboration.	Requires a dedicated team for the analysis of the data.
Ensures alignment of investments to Tunisians' needs.	Vulnerability to spam or fraudulent reports.
Allows long-term monitoring and tracking.	Limited coverage can lead to underrepresenting certain groups.
Improves transparency and accountability.	Individuals may not have access to the internet or the skills necessary to use the API.
Facilitates direct feedback.	Dependency on stakeholders to take action.

Table 3.2: Strengths and Limitations of TunInvest API

Chapter 4

Conclusion and Projections

4.1 Challenges

Some of the challenges I faced while preparing for this project:

- Deployment to render continuously showed an error concerning the version of the pip package although it was up to date.
- The refresh end-point showed a 422 Unprocessable Entity.
- Two-factor authentication implementation: I tried to send an email to users when they sign up but it didn't work out well.

4.2 Scalability

The TunInvest API has the potential to significantly improve the quality of life in Tunisian communities by providing a platform for individuals to voice their concerns and for decision-makers to take action. To maximize the potential of the API, several scalability methods were proposed to enhance the utility and services of the API. These methods include:

- Dashboards and visualizations: By providing investors and decision-makers with interactive dashboards and visualizations of the data collected through the API, they can quickly identify patterns and trends in the issues reported. This can help them make more informed decisions about how to address the most pressing concerns.
- Machine learning: By integrating machine learning algorithms, the API can be used to automatically classify and prioritize reported issues. This can provide decision-makers with a more efficient way to identify and address problems.
- Predictive analysis: By analyzing historical data and identifying patterns, the API can be used to predict future issues and help decision-makers take preventative action.
- Mobile App: Creating a mobile app that allows users to easily report issues from their phones and access the data via the API can increase the number of users and the amount of data that can be collected.
- Integration with other platforms: Integrating the TunInvest API with other platforms and tools, such as social media, can provide additional context and insights into the issues reported.
- Automated notifications: By providing automated notifications to relevant stakeholders when issues are reported, the API can help ensure that problems are addressed in a timely manner.
- Global Communities: This API can be used in the context of any other country and location.

4.3 Conclusion

The purpose of this project was to build an API that has a social and economic impact on Tunisia. Leveraging technology toward making Tunisia a better country is one of my career goals and I am glad I had the chance to use the knowledge I acquired in my web service class toward addressing daily problems encountered by Tunisians in their day-to-day life.

Appendix A

1 Project Structure

1.1 Visual Studio Code

TunInvest API was developed using the source code editor Visual Studio Code (VS Code). In Figure 4 we can see the structure of the code.

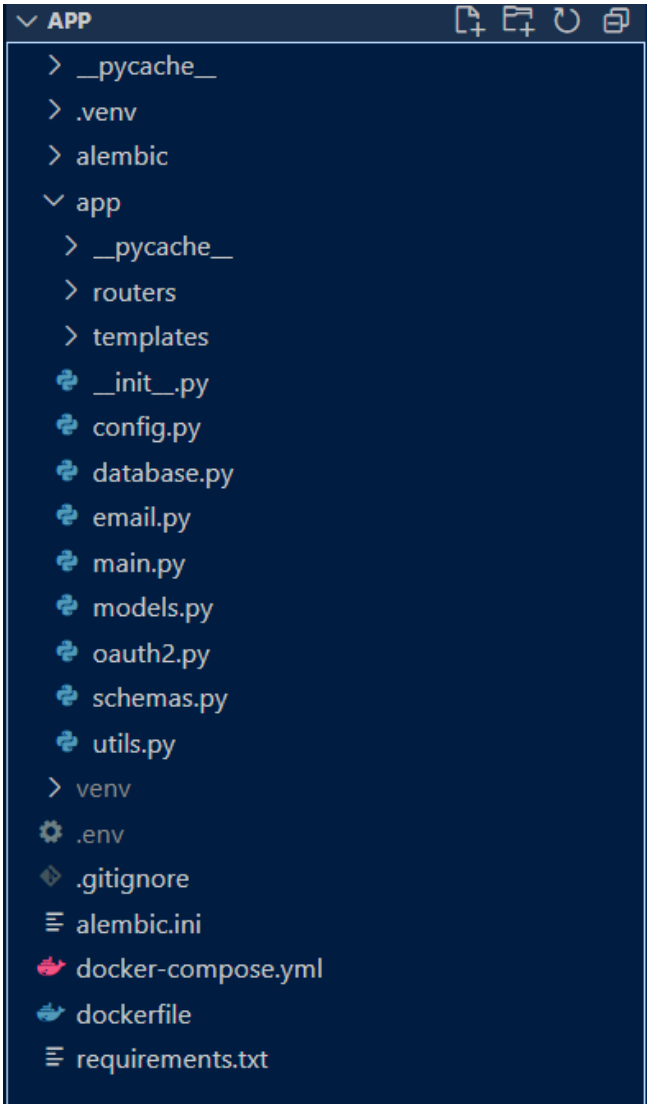

Figure 1: Project Structure

1.2 Database Tables

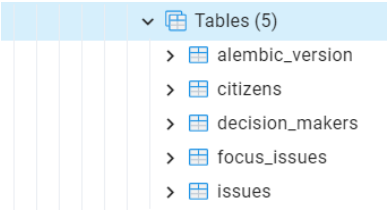

Figure 2: Tables

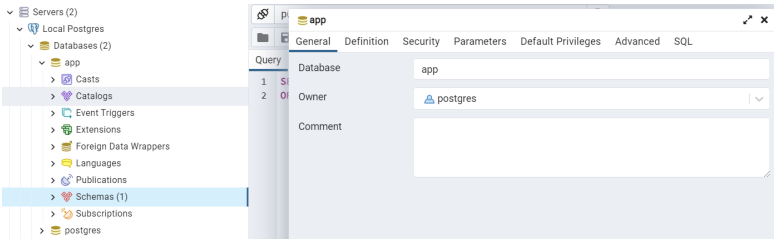

Figure 3: Postgres Configuration

1.3 Postman

The following figures show some of the operations implemented within Postman.

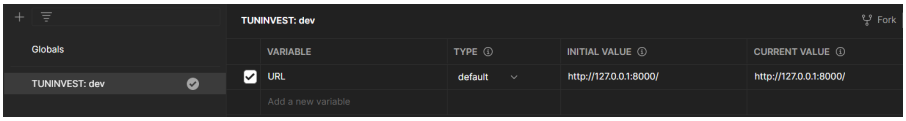

Figure 4: Development Environment

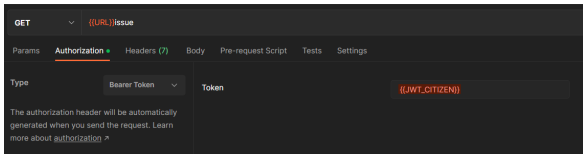

Figure 5: Authorization

Figure 6: Test code

1.4 Github Repository

Please refer to the repository using the following link: Github Repository.

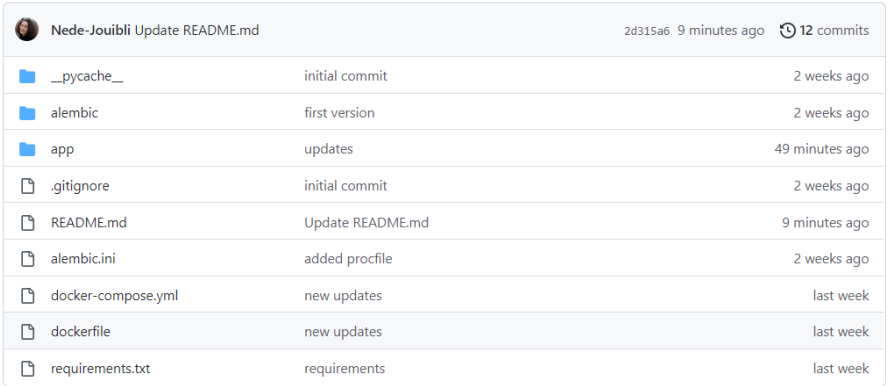

Figure 7: Github Repository

1.5 Dockerhub Repository

Please refer to the repository using the following link: Dockerhub Repository.

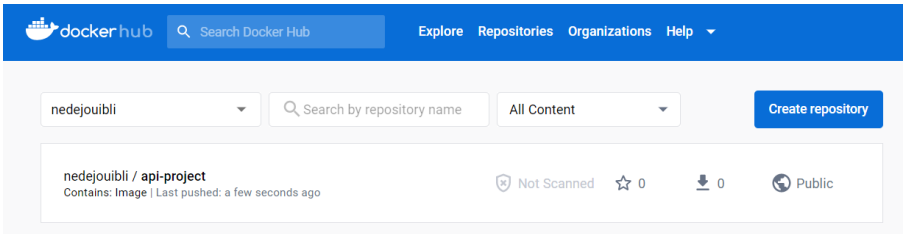

Figure 8: Dockerhub Repository

2 Response Examples

2.1 Authentication Required

In this example, the response shows all characteristics regarding the issue and the identity of the citizen who posted it.

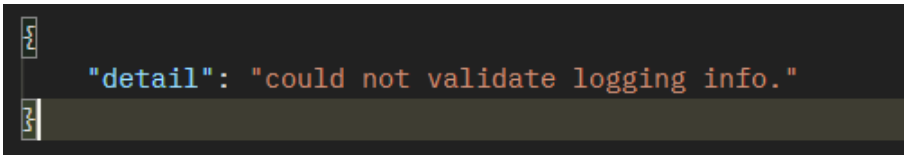

Figure 9: Post Issue Response

2.2 post /citizen/login

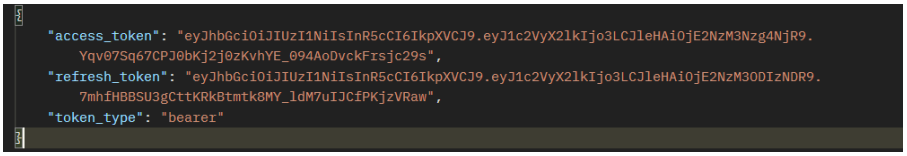

Figure 10: Login Response

2.3 post /issue

In this example, the response shows all characteristics regarding the issue and the identity of the citizen who posted it.

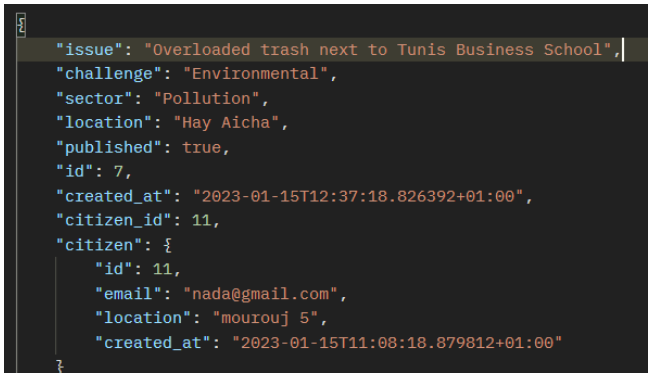

Figure 11: Post Issue Response

./references.bib

Bibliography

[1] https://pydantic-docs.helpmanual.io.

[2] https://alembic.sqlalchemy.org.

[3] Mike Bayer. *SQLAlchemy: The Database Toolkit for Python*. https://www.sqlalchemy.org/.

[4] *Bcrypt*. https://pypi.org/project/bcrypt/.

[5] *Docker*. https://www.docker.com/.

[6] *Docker Hub*. https://hub.docker.com/.

[7] *Git*. https://git-scm.com/.

[8] *GitHub*. https://github.com/.

[9] *Passlib*. https://passlib.readthedocs.io/en/stable/.

[10] *Postgres*. https://www.postgresql.org/.

[11] *Postman*. https://www.postman.com/.

[12] *PyJWT*. https://pyjwt.readthedocs.io/en/latest/.

[13] *Render*. https://render.com/.

[14] Juan Sanchez. *FastAPI*. https://fastapi.tiangolo.com.