Wanderly REST API

IT325 Web Services Final Project

by

Mariem Ferchichi

January 2025

BA/IT Senior Student

Business Analytics Major

Information Technology Minor

Tunis Business School

Ben Arous,TUNISIA.

2024-2025

Declaration of Academic Ethics

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I declare that I have properly and accurately acknowledged all sources used in the production of this report.

I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: January 20th, 2025 Mariem Ferchichi

Contents

Abstract **4**

1 Introduction **6**

2 Explanation of the work carried out **7**

2.1 API Features 7

2.2 Development Thought process 8

2.3 Database Structure 8

2.3.1 Use Case Diagram 9

2.3.2 Database Design 9

2.4 JWT Contribution 10

2.4.1 User Authentication: 10

2.4.2 Token-Based Authorization: 10

2.4.3 Stateless Authentication: 10

2.4.4 Role-Based Access Control: 11

2.4.5 Secure Communication: 11

2.4.6 Session Management: 11

2.4.7 Password-Protected Features: 11

2.4.8 JWT Workflow: 11

2.5 EJS/HTML/CSS Contribution 12

2.6 API Testing: All the HTTP Methods used 12

2.6.1 Insomnia Contribution 12

2.6.2 User Routes 13

2.6.3 Attractions Routes 14

2.6.4 Cultural Experience Routes 14

2.6.5 Accommodation Routes 15
2.6.6 Events Routes 16
2.6.7 Cuisines Routes 16
2.6.8 Transportation Routes 17
2.6.9 Local Services Routes 17
2.7 Git Contribution 18
2.8 Configuration 18
2.9 Technology Stack 19
2.9.1 Flask 19
2.9.2 Node.js 19
2.9.3 Typescript 19
2.9.4 Fastify 19
2.9.5 Swagger 20
2.9.6 JSON Web Tokens 20
2.9.7 Zod 20
2.9.8 bcrypt 21

3 Challenges **22**

4 Evaluation and Results **23**

5 Conclusion **24**

A Screenshots of Swagger Documentation **25**

Abstract

Adoption of digital technologies has accelerated largely in the last decade and has reached a critical stage today. [1]Tunisia alone is a driving force of the digital economy in North Africa and across the African continent. The digital economy accounts for 11 per cent of Tunisia's gross domestic product, making it one of the strongest and fastest growing sectors in the country. [2]

Specifically, the digitalization of tourism in Tunisia represents a significant shift in how the country manages and promotes its cultural, historical, and natural assets. As part of the broader trend of digital transformation across various sectors, tourism in Tunisia is increasingly embracing digital tools and platforms to improve visitor experiences, streamline services, and enhance economic growth.

This report delves into the design, development, and functionality of an online API, aimed to revolutionize tourism in the country. In response to the increasing demands of a digitized dining system, the API includes multiple features aimed at enhancing operational efficiency, user engagement, and overall traveling experiences for both locals and foreigners. The report comprehensively explores the technical architecture of the Wanderly API, outlining its modular and scalable design. The API's core functionalities include a list of all tourist attractions and accommodations in the database as well as the events available in real time. The integration of robust security measures and adherence to industry standards are highlighted to underscore the API's commitment to data privacy and regulatory compliance. As well as a focus on user experience that is maintained through automatic documentation generation using Swagger, offering clarity and accessibility to potential integrators.

Keywords : Digitisation, RESTAPI, python, flask-api, Insomnia, Swagger, redis, sqlalchemy

.

Motivation

Tourism plays a pivotal role in Tunisia's economy, contributing significantly to job creation and foreign exchange earnings. Despite its vast array of attractions, ranging from the ancient ruins of Carthage to the serene landscapes of the Sahara Desert, the country's tourism industry faces challenges in effectively marketing its assets to a global audience. Tunisia is often overlooked or underestimated, with many people quick to dismiss its beauty and the diversity of its offerings. This perception undermines the country's potential as a premier travel destination and leaves many of its hidden gems undiscovered by travelers. The motivation behind Wanderly the Discover Tunisia API stems from the desire to bridge this information gap by leveraging technology to promote Tunisia as a must-visit destination. By showcasing the rich and varied experiences the country has to offer, we aim to challenge misconceptions and highlight Tunisia's unique blend of history, culture, and natural wonders. Providing an easily accessible API empowers developers, businesses, and tourism agencies to create innovative applications that reveal Tunisia's true potential. This initiative not only aligns with the broader goals of digital transformation but also seeks to enhance the traveler's experience, boost local businesses, and ultimately contribute to the sustainable growth of Tunisia's tourism sector.

Chapter 1

Introduction

In an era where technology facilitates global connectivity and accessibility, the integration of software solutions into the tourism industry has become a crucial factor in enhancing travel experiences. Tunisia, a country rich in culture, history, and natural beauty, offers a myriad of attractions waiting to be explored by both locals and international tourists. However, the fragmented availability of information about these attractions often poses a challenge to potential visitors. To address this gap, we propose the development of an Application Programming Interface (API) designed to centralize and streamline the discovery of Tunisia's treasures. The Wanderly API aims to provide a comprehensive and user-friendly platform for accessing information about Tunisia's cultural heritage, tourist destinations, and essential travel services. My final Project for the IT325 Web Services course this semester consists of an online RESTful API developed using Flask- A python Micro-web framework and other additional packages described below in Technology Stack Section. I have also used technologies such as Insomnia, VSCode, and Git-Version Control to maximize the project's quality. This projects aim is to facilitate the information sharing of the Tunisian restaurant data in a secure manner to help the economy as well as tourism.

Chapter 2

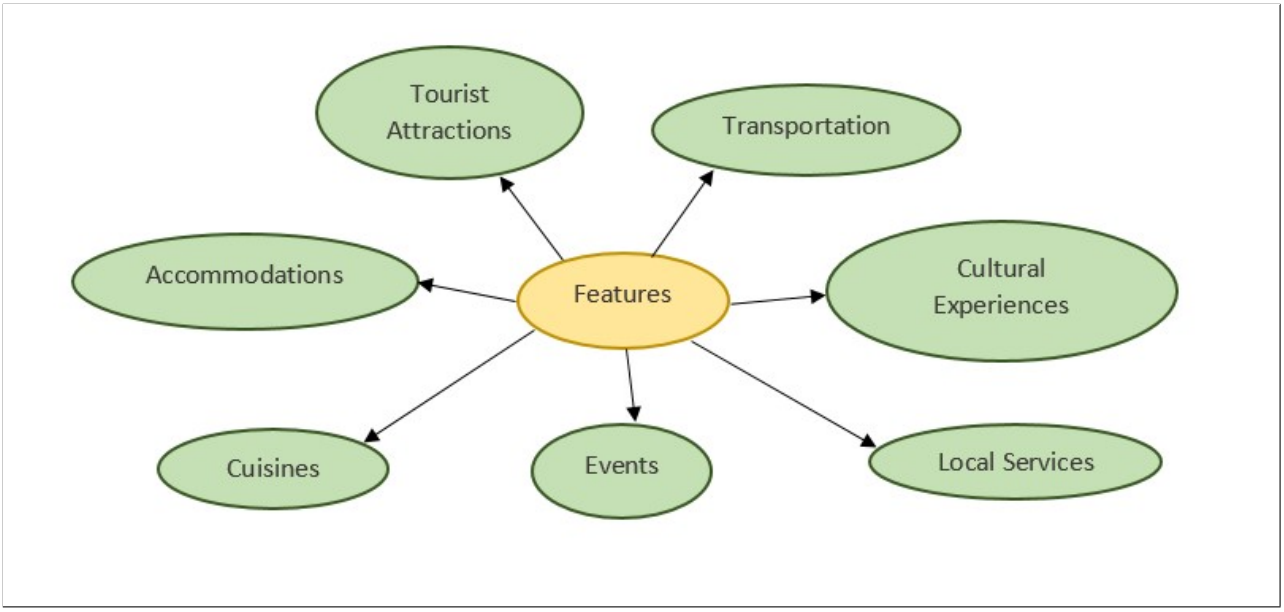
Explanation of the work carried out

2.1 API Features

The Wanderly API aims to provide a comprehensive and user-friendly platform for accessing information about Tunisia's cultural heritage, tourist destinations, and essential travel services.

1. **Tourist Attractions:** Details about landmarks, historical sites, museums, beaches, and parks.
2. **Cultural Experiences:** Information about festivals, local traditions, music, art, and cuisine.
3. **Accommodation:** Hotels, guest houses, and unique stays.
4. **Transportation:** Public transport routes, car rentals, and nearby airports.
5. **Events:** Upcoming events, concerts, and exhibitions.
6. **Cuisine:** Restaurants, cafes, and local dishes with their descriptions and locations.
7. **Local Services:** Emergency contacts, hospitals, and embassies.

The figure below encompasses a large variety of features aimed to give the user all the information necessary on his journey to discover all that Tunisia has to offer:



2.2 Development Thought process

This project was developed using **Flask** - a python micro web framework. I used a micro service architecture style as it involves developing a software system as a collection of small, independent, and loosely coupled services. Each microservice is responsible for a specific business capability and communicates with others through well-defined APIs. This promotes scalability, maintainability, and ease of development as well as the independent deployment of services.

For documentation, I used **Swagger** as it helps in maintaining up-to-date and consistent API documentation.

I used a **Test Driven Development** where tests are written before the code to catch errors easily and early in the coding process.

I also used **RedisDB** as a catching layer to improve read efficiency. Redis is an in-memory data store that can be used as a cache, message broker, and more. By storing frequently accessed data in memory, it reduces the need to retrieve data from slower storage systems like databases.

2.3 Database Structure

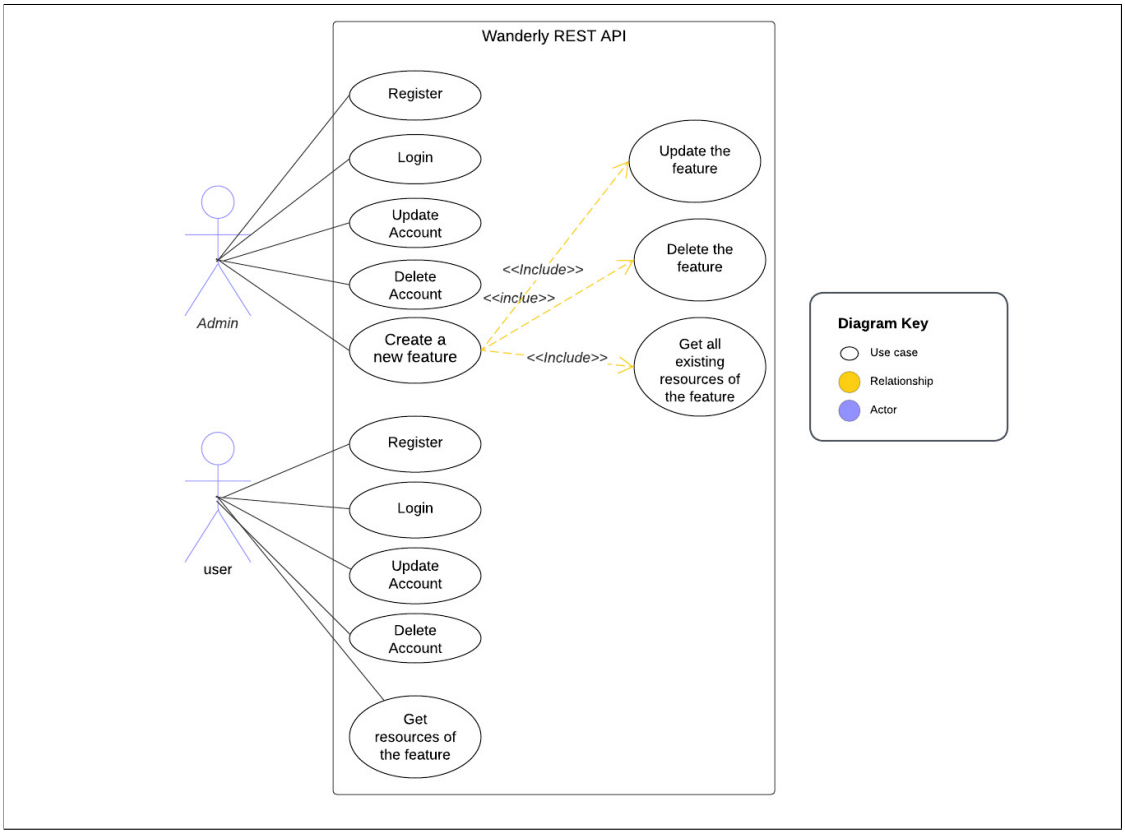
The database consists of 8 tables: User, Tourist Attractions, Cultural Experiences, Accommodations, Transportation, Events, Cuisine,and Local services. The relations between them is as follows:

Tourist Attractions, Cultural Experiences, Accommodations, Transportation, Events, Cuisine, and Local Services are independent entities.

Admins have the ability to manage (add, update, delete)Tourist Attractions, accommodations, events, etc.

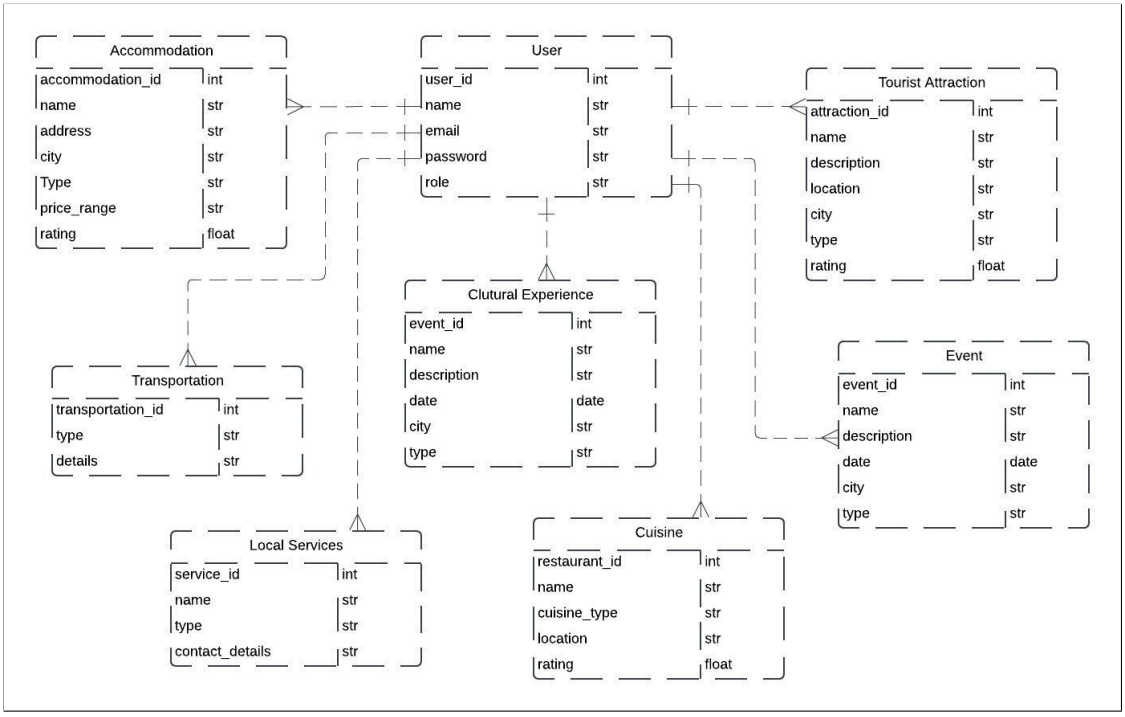
2.3.1 Use Case Diagram

The figure below shows the Use Case Diagram of the project.



2.3.2 Database Design

The figure below shows the Entity Relationship Diagram ERD of the project.



2.4 JWT Contribution

JSON Web Token is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key. [3]

2.4.1 User Authentication:

JWT is used to generate an access token during user login (`/login` route).
The `create_access_token` function generates a signed token containing the user's unique identity (`user.id`) and an expiration time (`expires_delta`).

2.4.2 Token-Based Authorization:

The `@jwt_required()` decorator ensures that specific routes (e.g., `/logout`, `/current_user`, `/admin`, etc.) are accessible only to authenticated users who provide a valid JWT in the request headers.

2.4.3 Stateless Authentication:

JWT eliminates the need for server-side session storage. Tokens are self-contained, carrying user identity and any claims (e.g., roles) required for authorization.

This makes the application scalable since no session data needs to be maintained on the server.

2.4.4 Role-Based Access Control:

The `/admin` route demonstrates role-based access control. The JWT token's payload (`get_jwt_identity`) is used to fetch the user's role (`user.role`). Only users with the "admin" role can access this route.

2.4.5 Secure Communication:

The tokens are signed with a secret key (`SECRET_KEY`) and algorithm (`HS256`), ensuring that they cannot be tampered with. Only the server with the correct secret can validate the token.

2.4.6 Session Management:

Routes like `/logout` and `/current_user` rely on the JWT to identify and interact with the currently logged-in user without additional database queries or sessions.
Tokens are stateless, meaning the user's session persists until the token expires or is manually revoked.

2.4.7 Password-Protected Features:

The `/change_password` route uses the JWT to identify the logged-in user securely and allows them to update their password only after verifying the old password.

2.4.8 JWT Workflow:

1. The user sends a request with the sign in that has the email and password.
2. The authorization server will process the request and sends a response that contains the access and refresh tokens if the user's credentials are valid and a new session is created.
3. The user uses the access token to send requests for data from the resource server.
4. Resource server processes requests and sends protected resources.
5. The token has expired and the user will request data using it.

6. Resource server sends a 401 unauthorized response because the token is invalid.
7. User sends a request to the authorization server to get new access and refresh tokens.
8. User receives new access and refresh tokens from the authorization server with a new session.

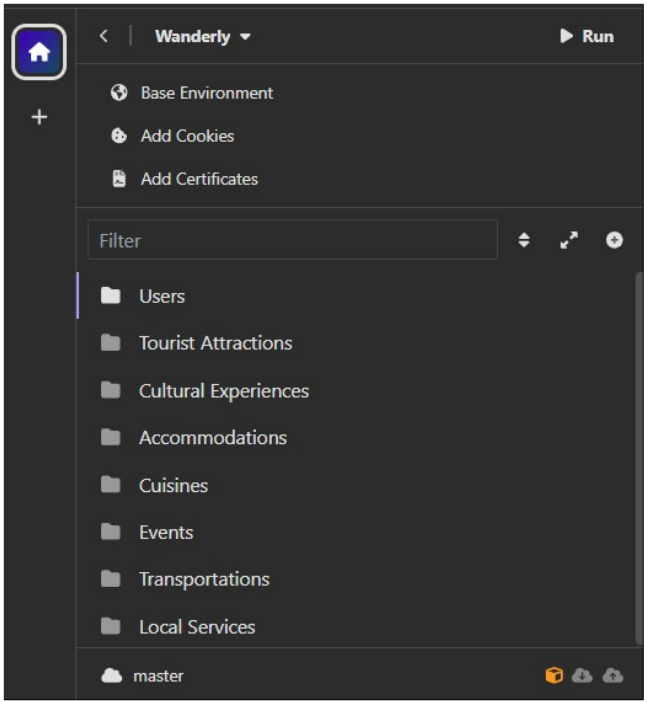
2.5 EJS/HTML/CSS Contribution

EJS is a simple templating language which is used to generate HTML markup with plain JavaScript. It also helps to embed JavaScript to HTML pages. [4] I used HTML,CSS, and EJS to design the interactive web pages used in my app.

2.6 API Testing: All the HTTP Methods used

2.6.1 Insomnia Contribution

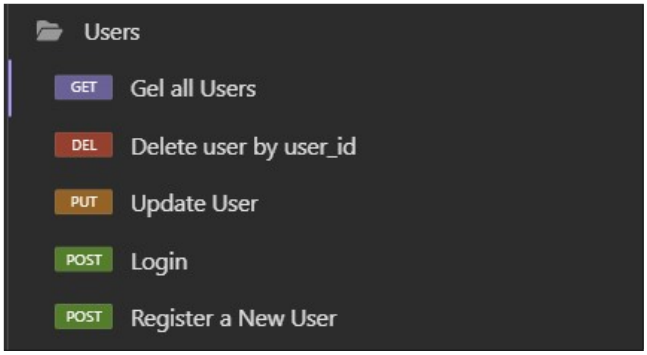
Insomnia is an application used for API testing. It is an HTTP client that tests HTTP requests, utilizing a graphical user interface, through which we obtain different types of responses that need to be subsequently validated. [5] I have used Insomnia for the automatic testing of my API requests, as well as some snippets such as "Response Time less than 200ms","Status Code is 200"..etc.



2.6.2 User Routes

The user routes are defined in the "UserRoutes" function, which is passed an instance of the Fastify server. Users have a role either an admin or a normal user.Only an admin can create, update or delete a resource. All the routes starts with "/api/auth" The routes include:

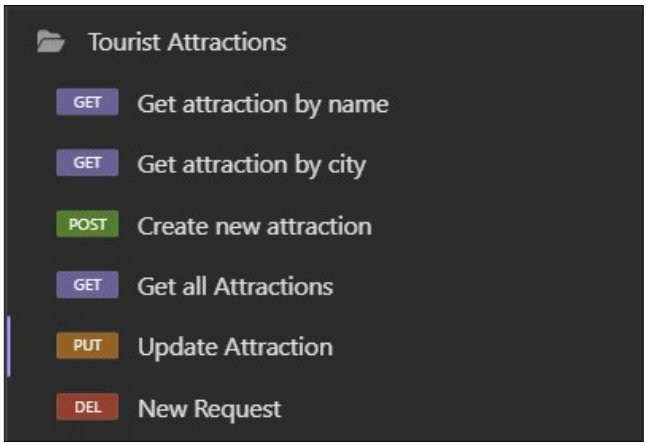
- A POST request to "/register" which is used to register a new user to the database.
- A POST request to "/login" which is used to sign in a user and provides access and refresh tokens.
- A PUT request to update user parameters (name, email, password).
- A Delete request to delete current user by user id.
- A GET request to get a list of all users.



2.6.3 Attractions Routes

Attractions can be landmarks, historical sites, museums, beaches, and parks.
While any user can get all attractions saved in the database or specific one by name or city, only users with the role admin can post, put or delete attractions.

- A POST request to create a new tourist attraction.
- A PUT request to update an existing one
- A Delete request to delete attraction by id.
- A GET request to get a list of all attractions.
- A GET request to get an attraction by name.
- A GET request to get an attraction by city.



2.6.4 Cultural Experience Routes

Cultural Experience can be music, an art,or tradition. The endpoints of the cultural experience are as follows.

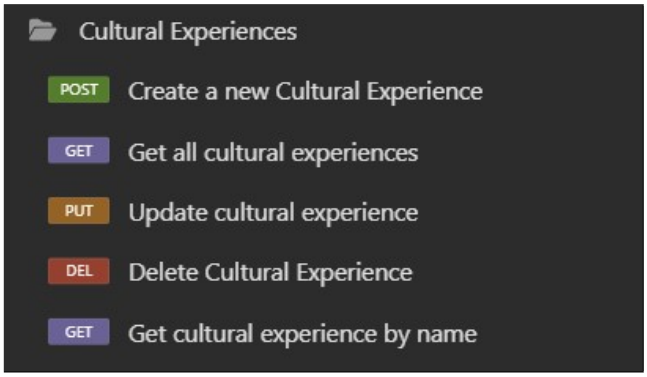
A POST request to create a new cultural experience.

A PUT request to update an existing one

A Delete request to delete experience by id.

A GET request to get a list of all experiences.

A GET request to get an experience by name.



2.6.5 Accommodation Routes

The types of accommodations include hotel, guesthouse, hostel, resort.

A POST request to create a new accommodation.

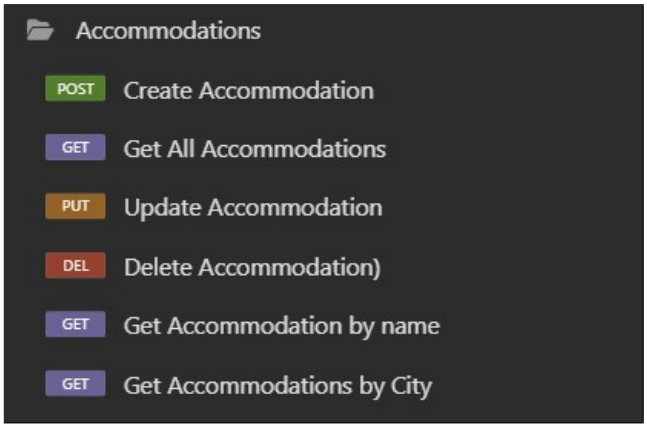
A PUT request to update an existing one

A Delete request to delete accommodation by id.

A GET request to get a list of all accommodations.

A GET request to get an accommodation by name.

A GET request to get an accommodation by city.



2.6.6 Events Routes

Events can be concerts, exhibitions or a gallery, or even a stand-up comedy event.

A POST request to create a new event.

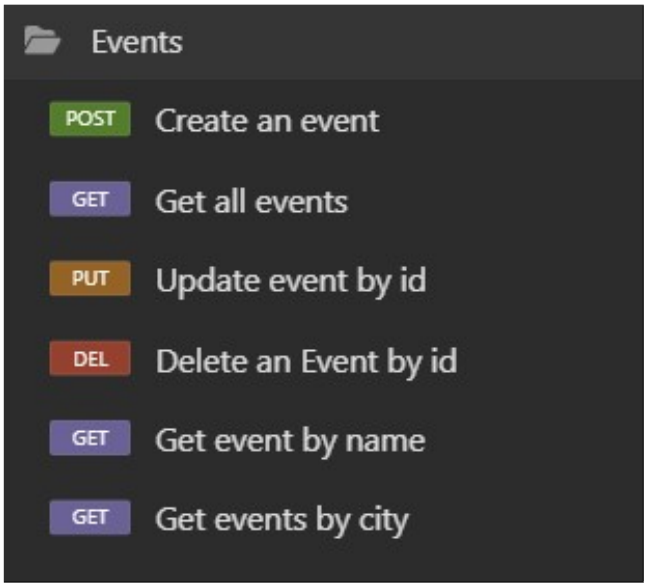
A PUT request to update an existing one

A Delete request to delete event by id.

A GET request to get a list of all events.

A GET request to get an event by name.

A GET request to get an event by city.



2.6.7 Cuisines Routes

The types of cuisines include Tunisian, Mediterranean, Italian, French, Asian.

A POST request to create a new cuisine.

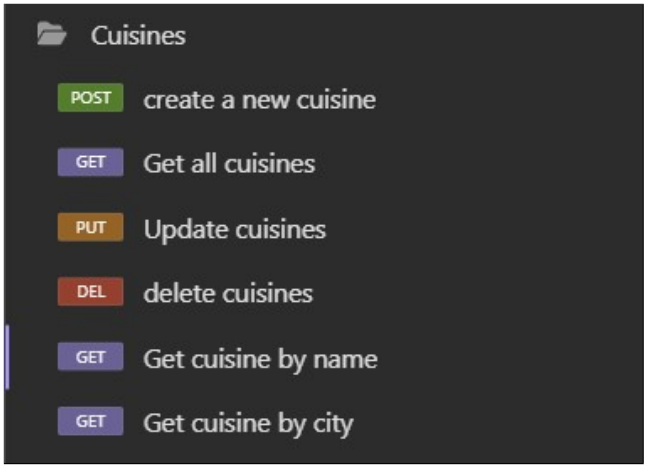
A PUT request to update an existing one

A Delete request to delete a restaurant by id.

A GET request to get a list of all restaurants.

A GET request to get a restaurant by name.

A GET request to get all restaurants in a given city.



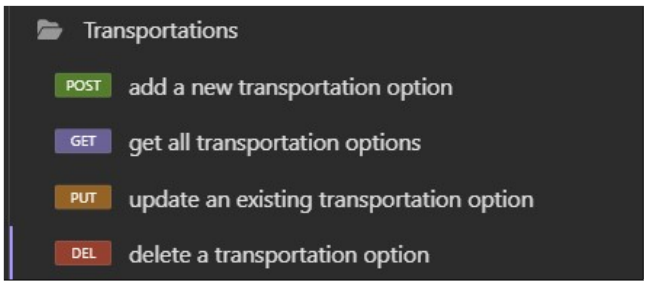
2.6.8 Transportation Routes

A POST request to create a new transportation.

A PUT request to update an existing one using id.

A Delete request to delete transportation by id.

A GET request to get a list of all transportation in the system.



2.6.9 Local Services Routes

A POST request to create a new service.

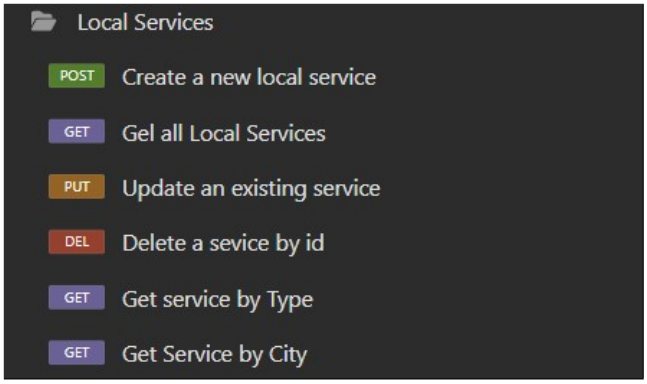
A PUT request to update an existing one

A Delete request to delete service by id.

A GET request to get a list of all services.

A GET request to get a service by type.

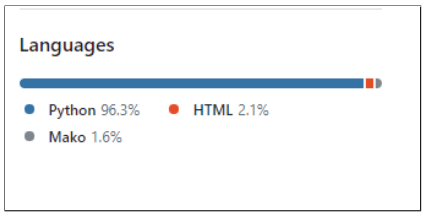
A GET request to get all services available in a given by city.



2.7 Git Contribution

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. [6]

I have used Git-Version Control to save changes and coordinate between the different repositories that I had (local and remote).



2.8 Configuration

Python code:

```python
def create_app(db_url=None):
    app = Flask(__name__)

    # Configure the application
    app.config["API_TITLE"] = "Wanderly Discover Tunisia API"
    app.config["API_VERSION"] = "v1"
    app.config["OPENAPI_VERSION"] = "3.0.3"
    app.config["OPENAPI_URL_PREFIX"] = "/"
    app.config["OPENAPI_SWAGGER_UI_PATH"] = "/swagger-ui"
    app.config["OPENAPI_SWAGGER_UI_URL"] = "https://cdn.jsdelivr.net/npm/sw
    app.config["SQLALCHEMY_DATABASE_URI"] = db_url or "sqlite:///data.db"
    app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False
    app.config["PROPAGATE_EXCEPTIONS"] = True

    # Initialize database
    db.init_app(app)

    # Initialize the API with Flask-Smorest
    api = Api(app)
```

2.9 Technology Stack

2.9.1 Flask

Flask - Web Microframework for Python.

Flask-restful - Extension for flask for quickly building REST APIs.

Flask-sqlalchemy - This is an extension of flask that add supports for SQLAlchemy. - marshmallow - An integration layer for flask and marshmallowF

2.9.2 Node.js

Node.js [7] is a cross-platform, open-source server environment that can run on Windows, Linux, Unix, macOS, and more. Node.js is a back-end JavaScript runtime environment, runs on the V8 JavaScript Engine, and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting.

2.9.3 Typescript

TypeScript [8] is a free and open source programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript and adds optional static typing to the language. It was developed and maintained by Microsoft. It is designed for the development of large applications and transpiles to JavaScript. As it is a superset of JavaScript, existing JavaScript programs are also valid TypeScript programs.

2.9.4 Fastify

Fastify [9] is a web framework highly focused on providing the best developer experience with the least overhead and powerful plugin architecture, inspired by Hapi and Express. As far as we know, it is one of the fastest web frameworks in town. It was created by Matteo Collina and Gabriele Lana in 2017 as an alternative to popular web frameworks such as Express.js. The developers aimed to create a framework that would allow for faster performance and more efficient use of resources when building web applications.

2.9.5 Swagger

Swagger is an open-source framework for designing, building, and documenting RESTful APIs. It provides a set of tools for creating and managing API documentation, testing, and client code generation. Swagger was created by Tony Tam and his team at Reverb Technologies in 2011. The framework was later donated to the OpenAPI Initiative, which is now maintained by the Linux Foundation. Swagger is widely used to design and document APIs in various programming languages, and it allows for a more efficient and streamlined development process by providing an easy-to-use interface for creating and managing API documentation.

2.9.6 JSON Web Tokens

JSON Web Token is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key. JWT was created by the company Auth0, which was founded by Matias Woloski and Eugenio Pace. JWT is widely adopted and supported by many programming languages and frameworks, which makes it a popular choice for securing web applications and APIs.

2.9.7 Zod

Zod is a TypeScript library for data validation and type-safe input validation. It provides a simple and expressive syntax for defining complex validation rules and constraints. Zod allows you to define a schema for an object and then validate that an object matches that schema. It was created by Colin McDonnell. Zod is designed to simplify the process of data validation in TypeScript and make it more robust and less error-prone, by taking advantage of TypeScript's type checking capabilities, it can provide a lot more feedback to developers about errors and make the development process smoother.

2.9.8 bcrypt

bcrypt is a password-hashing function designed by Niels Provos and David Mazières, based on the Blowfish cipher and presented at USENIX in 1999.[1] Besides incorporating a salt to protect against rainbow table attacks, bcrypt is an adaptive function: over time, the iteration count can be increased to make it slower, so it remains resistant to brute-force search attacks even with increasing computation power.The bcrypt function is the default password hash algorithm for OpenBSD and was the default for some Linux distributions such as SUSE Linux.There are implementations of bcrypt in C, C++, C#, Embarcadero Delphi, Elixir, Go, Java, JavaScript, Perl, PHP, Python, Ruby, and other languages.

Chapter 3

Challenges

While developing this API, I faced several problems and tried to overcome them. Some of these challenges are as follows:

- One of the challenges was Data Management and Accuracy as I had to ensure that all the data (e.g., tourist attractions, events, local services) is up-to-date, accurate, and relevant. I overcame that by implementing a system for regular updates and validation.
- Another aspect of the code where I had difficulties is Handling Large Volumes of Data as managing and efficiently querying a large database, especially with multiple categories like tourist attractions, accommodation, and events can be quite tricky.My solution was to optimize database queries using indexing, pagination, and caching mechanisms.
- Another aspect of the code where I had difficulties is the implementation of Authentication and Authorization mechanisms to secure the API and control access to resources. Choosing the right authentication method (e.g., API keys, OAuth) and managing user permissions is critical. I used industry-standard authentication protocols and implemented token-based authentication. I used a code from the internet to secure the authentication and authorization methods.
- Handling errors effectively and providing meaningful error messages was also quite difficult. Consistent error responses must be defined and communicated. Thus ,I had to be careful as I implemented standardized error responses and provided detailed error messages.

Chapter 4

Evaluation and Results

A potential development gap in the Discover Tunisia API lies in the limited scope of its current functionalities, as there is considerable room for expansion. One enhancement could involve the integration of real-time booking and reservation features for tourist attractions, accommodations, and events. This would allow users to make reservations seamlessly, regardless of their location, offering a more convenient and user-friendly experience without the need for phone calls or direct visits.

Additionally, enabling users to submit reviews and share experiences about tourist spots, cultural events, and services would foster a more interactive platform. This could be paired with a feature that provides location-based services, such as showing the nearest attractions, accommodations, and restaurants, along with the estimated travel time to reach them.

Another valuable feature could be the introduction of personalized recommendations, leveraging user preferences, past visits, and interactions to suggest relevant tourist spots and activities. This personalized approach would enhance the user experience, making it more tailored and engaging, thus encouraging repeat usage.

Overall, the Wanderly API lays a strong foundation for a comprehensive tourism platform, there is ample opportunity for further development to create a more dynamic and enriching experience for both travelers and local businesses.

Chapter 5

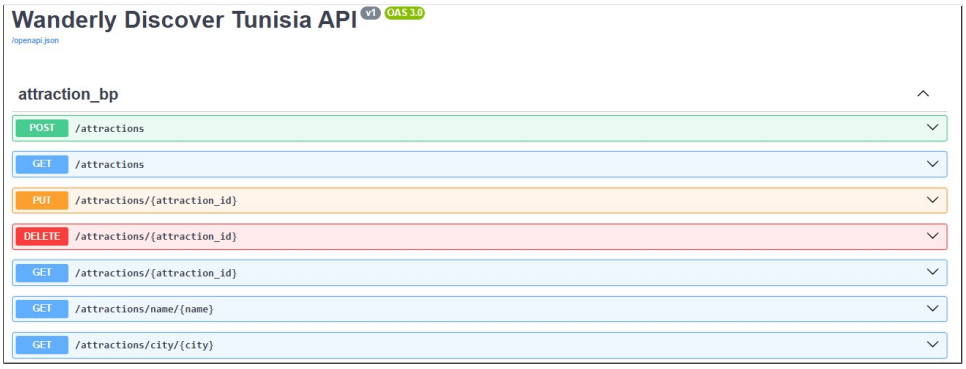
Conclusion

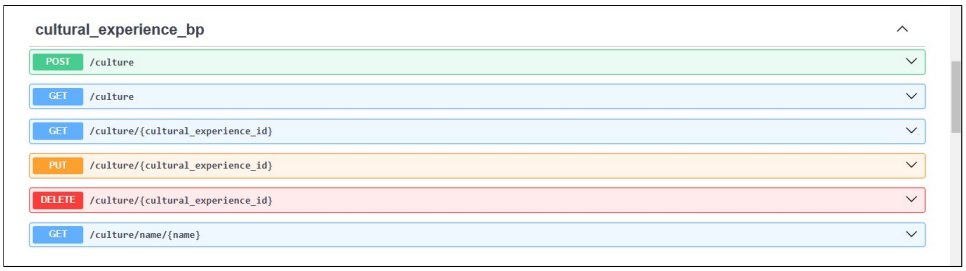
In summary, this technical report serves as a guide for tourists and local Tunisians seeking to leverage the potential of an online Tourism and Travel API. The Wanderly API represents a transformative step toward revitalizing Tunisia's tourism industry and reshaping global perceptions of the country. By providing a centralized, comprehensive, and accessible platform for exploring Tunisia's cultural, historical, and natural assets, the API has the potential to attract a broader audience and foster deeper engagement with Tunisia's rich heritage. This initiative not only addresses the current challenges of fragmented information and underrepresentation but also aligns with the goals of sustainable tourism and digital innovation. As Tunisia continues to unveil its hidden treasures to the world, the Discover Tunisia API stands as a testament to the power of technology in unlocking the true potential of this remarkable destination. Overall, Tunisia is a beautiful country and has a lot to offer and I sincerely hope that through this project I helped bettering our country's future and facilitating our lives through efficient technologies and I hope that that was conveyed throughout my work.

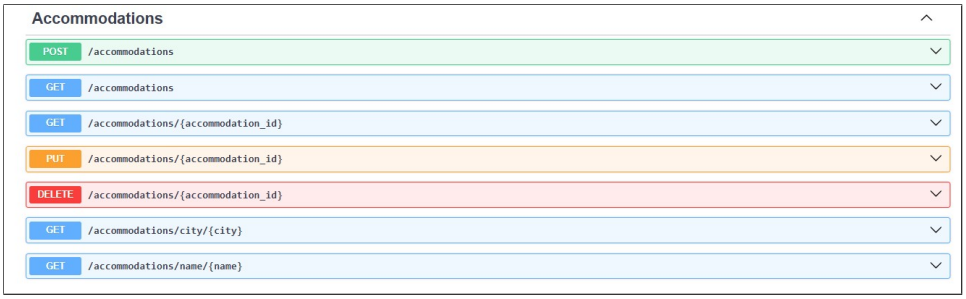
Mariem Ferchichi

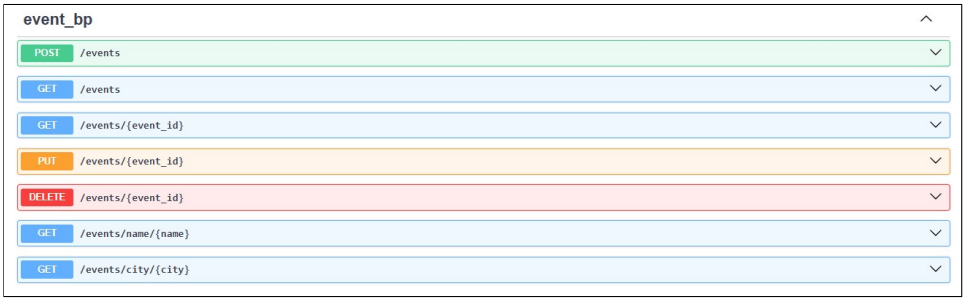
Appendix A

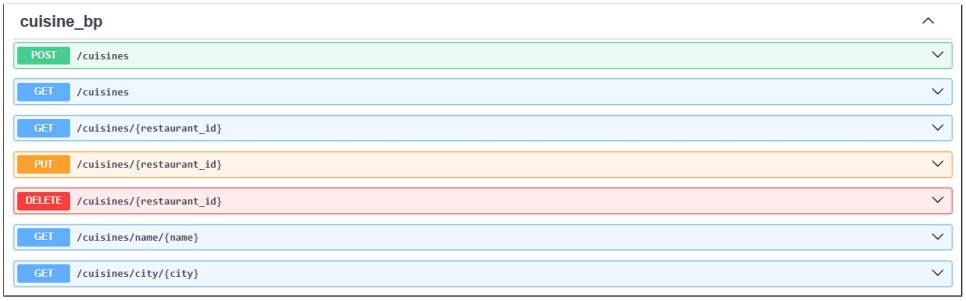
Screenshots of Swagger Documentation

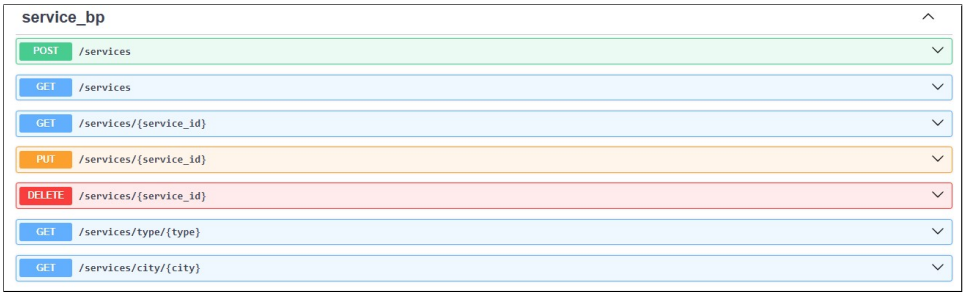


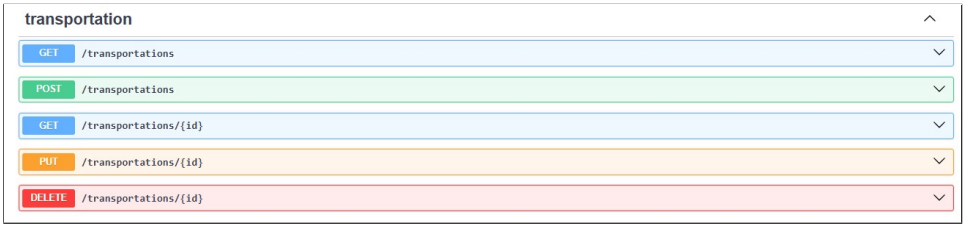


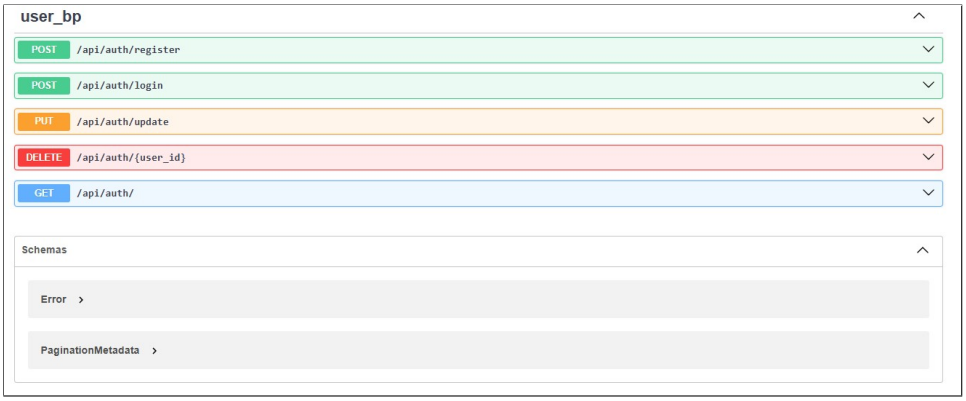



event_bp
POST /events
GET /events
GET /events/{event_id}
PUT /events/{event_id}
DELETE /events/{event_id}
GET /events/name/{name}
GET /events/city/{city}


cuisine_bp
POST /cuisines
GET /cuisines
GET /cuisines/{restaurant_id}
PUT /cuisines/{restaurant_id}
DELETE /cuisines/{restaurant_id}
GET /cuisines/name/{name}
GET /cuisines/city/{city}


service_bp
POST /services
GET /services
GET /services/{service_id}
PUT /services/{service_id}
DELETE /services/{service_id}
GET /services/type/{type}
GET /services/city/{city}


transportation
GET /transportations
POST /transportations
GET /transportations/{id}
PUT /transportations/{id}
DELETE /transportations/{id}


user_bp
POST /api/auth/register
POST /api/auth/login
PUT /api/auth/update
DELETE /api/auth/{user_id}
GET /api/auth/
Schemas
Error
PaginationMetadata

Bibliography

[1] A. B. Youssef, "Digital transformation in tunisia: Under which conditions could the digital economy benefit everyone?," in *ERF Working Papers Series*, pp. 1–42, Nov 2021.

[2] N. Schraepel, "Digital innovations create jobs and support transparent and effective public administration," *GIZ deutsche gesellschaft für internationale zusammenarbeit*, p. 1, 2020.

[3] "Jwt." `https://en.wikipedia.org/wiki/JSON_Web_Token`. [Online; accessed 23-Jan-2022].

[4] GeeksForGeeks, "Use ejs as template engine in node.js," *GeeksForGeeks*, p. 1, Jul 2021.

[5] G. Romero, "What is insomnia api test," *encora*, p. 1, June 2021.

[6] "Git." `https://git-scm.com/`. [Online; accessed 23-Jan-2022].

[7] "Node.js." https://en.wikipedia.org/wiki/Node.js. [Online; accessed 19-Jan-2024].

[8] "Typescript." https://en.wikipedia.org/wiki/TypeScript.

[9] "Fastify." https://github.com/fastify/fastify.