RestoConnect

A Restaurant REST API

Mariem Ferchichi

January,2024

Overview

1. Introduction

2. What is RestoConnect ?

3. Scalability

4. Challenges

5. Future Milestones

Introduction

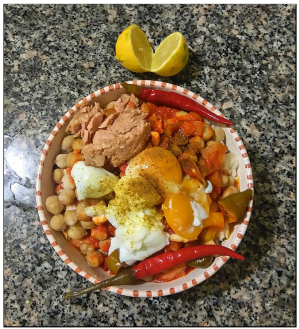
Introduction

Digitisation in Tunisia

Introduction

Restaurants in Tunisia

Food is a big part of the Tunisian culture. We eat at festivals, at holidays both civil and religious, at weddings, even at funerals. No matter the occasion, you will find that food is always there.



Introduction

RestoConnect API

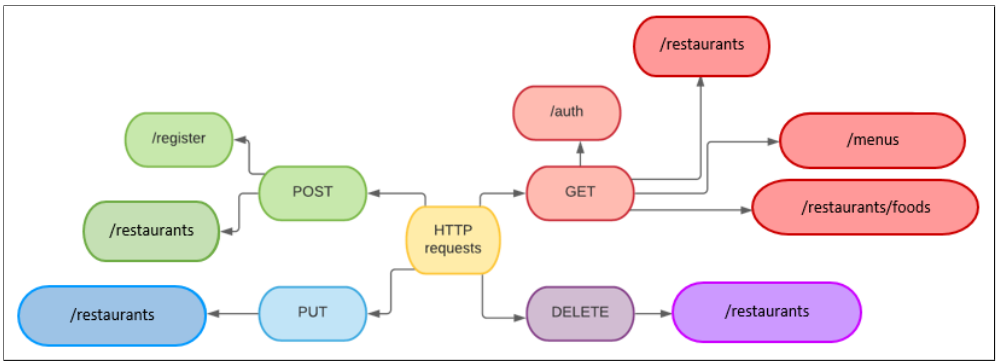
RestoConnect is **an online restaurant API** which displays restaurant options, with their menus and food items. It integrates data for all local restaurants and diets, including Mediterranean, Keto and vegan/vegetarian.

What is RestoConnect ?

What is RestoConnect ?

Definition, Functions and Features

RestoConnect is **an online restaurant REST API**, it enables its users to send the following requests :

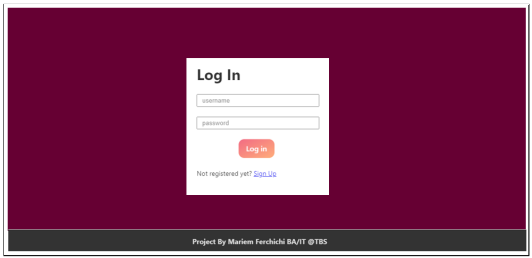


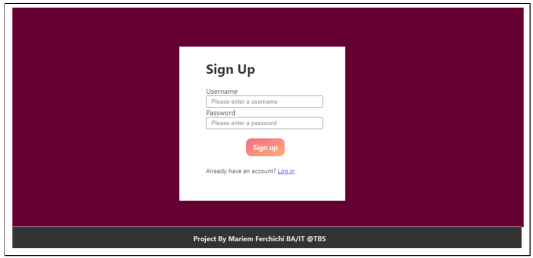
What is RestoConnect ?

Interface

What is RestoConnect ?

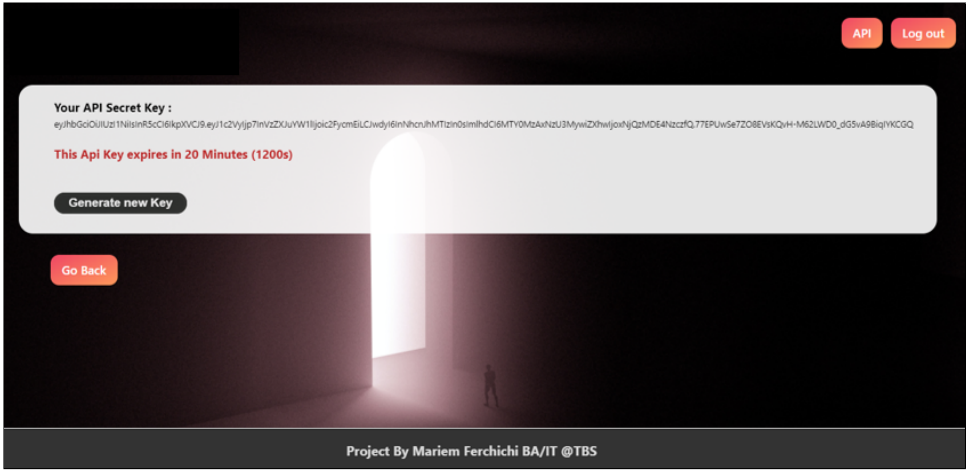
Interface





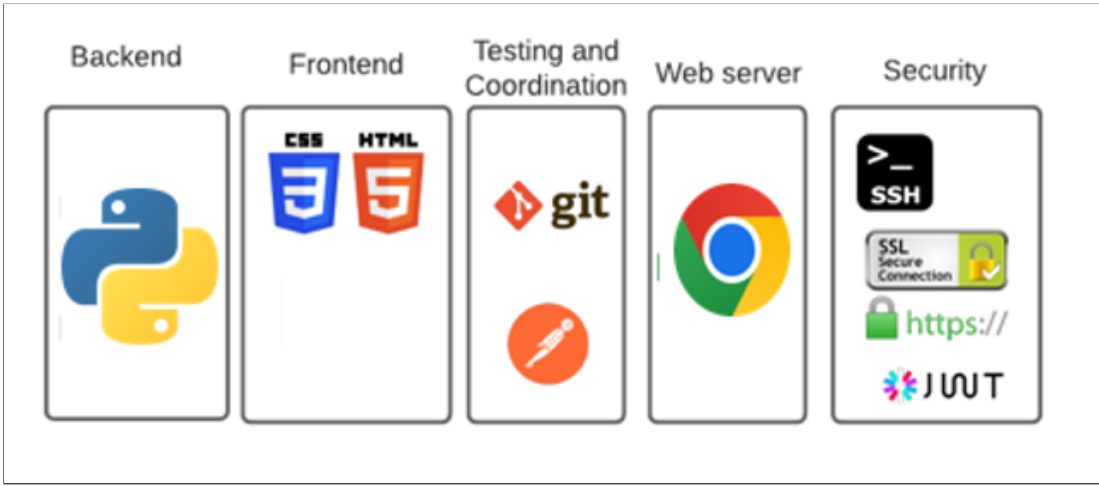
What is RestoConnect ?

Interface



What is RestoConnect ?

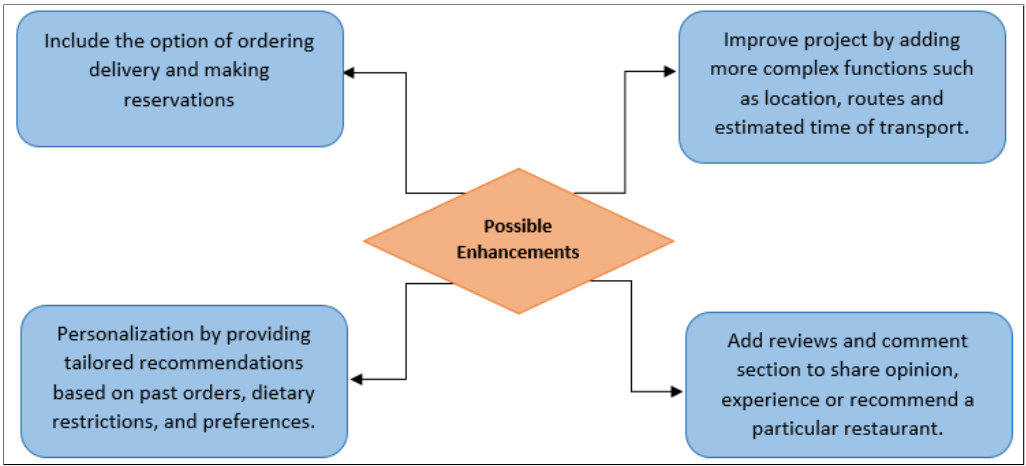
Technically



Scalability

Scalability

Possible Enhancements



Challenges

Challenges

Problems that I faced and how I overcame them

- Passing headers, parameters, and body objects in the browser
  Solution: ModHeader Chrome Extension
- Authentication and Authorization
  Solution:Use industry-standard authentication protocols and implement token-based authentication.
- Error Handling and Reporting
  Solution:Implement standardized error responses, use HTTP status codes appropriately, and provide detailed error messages.

Future Milestones

Future Milestones

Next Step

Now that the development phase is over. We move on to the next step. We start by monitoring customer response and analyse the feedback to improve our project even further. In addition, we start on the possible improvements previously mentioned.

```
import httplib2
import json

def getGeocodeLocation(inputString):
    google_api_key = "HIDDEN_USE_YOUR_GOOGLEAPI_KEY_HERE"
    locationString = inputString.replace(" ", "+")
    url = ('https://maps.googleapis.com/maps/api/geocode/json?address=%s&key=%s' % (locationString, google_api_k
    h = httplib2.Http()
    response, content = h.request(url, 'GET')
    result = json.loads(content)
    latitude = result['results'][0]['geometry']['location']['lat']
    longitude = result['results'][0]['geometry']['location']['lng']
    return (latitude, longitude)
```

```
import json
import httplib2
import sys
import codecs

sys.stdout = codecs.getwriter('utf8')(sys.stdout)
sys.stderr = codecs.getwriter('utf8')(sys.stderr)

client_id = "YOUR_ID_HERE"
client_secret = "YOUR_SECRET_HERE"
google_api_key = "YOUR_KEY_HERE"


def getGeocodeLocation(inputString):
    locationString = inputString.replace(" ", "+")
    url = ('https://maps.googleapis.com/maps/api/geocode/json?address=%s&key=%s' % (locationString, google_api_k
    h = httplib2.Http()
    result = json.loads(h.request(url, 'GET')[1])
    latitude = result['results'][0]['geometry']['location']['lat']
    longitude = result['results'][0]['geometry']['location']['lng']
    return (latitude, longitude)
```

Appendix

security.py > authenticate

```python
from flask import Flask, render_template, request, jsonify, session
from flask_session import Session
import psycopg2
import jwt
from uuid import uuid4
from router import router

app = Flask(__name__)
app.config['SESSION_TYPE'] = 'filesystem'
app.config['SECRET_KEY'] = str(uuid4())
Session(app)


# SIGNUP PAGE
@app.route("/signup")
def signup():
    return render_template('signup')

# LOGIN PAGE
@app.route("/login")
def login():
    return render_template('login')

# LOGIN (get a token)
@app.route('/auth', methods=['GET'])
def authenticate():
    username = request.args.get('username')
    password = request.args.get('password')

    try:
        cursor = pool.cursor()
        cursor.execute("SELECT * FROM users WHERE username = %s", (username,))
```

config.py > ...

```python
class BaseConfig:
    SQLALCHEMY_ECHO = False
    SQLALCHEMY_TRACK_MODIFICATIONS = True
    SQLALCHEMY_DATABASE_URI = "sqlite:///app.db"
    REDIS_HOST = "localhost"
    REDIS_PASSWORD = "61a62a841f54d08ec165faf2ad20544e9f6fb8d37d10bf429b6e9f6e2807e0c4"
    REDIS_PORT = 6379

class DevelopmentConfig(BaseConfig):
    SQLALCHEMY_DATABASE_URI = "sqlite:///dev.db"

class TestingConfig(BaseConfig):
    SQLALCHEMY_DATABASE_URI = "sqlite:///test.db"

class ProductionConfig(BaseConfig):
    SQLALCHEMY_DATABASE_URI = "sqlite:///base.db"


PresentConfig = BaseConfig

```

Database Configuration + Postman Snippets

db.py > ...

```python
import psycopg2
from psycopg2 import pool

# Define database connection parameters
db_params = {
    "user": "gyqhjwwg",
    "password": "olRD08yXC3tPEnG-2KoTv9tjEY42ELiW",
    "database": "xwnxumoi",
    "host": "arjuna.db.elephantsql.com",
    "port": 5432
}

# Create a PostgreSQL connection pool
connection_pool = psycopg2.pool.SimpleConnectionPool(
    minconn=1,
    maxconn=10,
    **db_params
)

# Function to get a connection from the pool
def get_connection():
    return connection_pool.getconn()

# Function to release a connection back to the pool
def release_connection(conn):
    connection_pool.putconn(conn)

# Function to close all connections in the pool
def close_all_connections():
    connection_pool.closeall()
```

Params | Authorization | Headers (10) | Body | Pre-request Script | Tests | Settings

```javascript
pm.test("Response time is less than 200ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(500);
});
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

Conclusion

Conclusion

In summary, the purpose of this project was to build an API with not only a practical purpose, but also a social and economic one. It also serves a touristic purpose as it would be a helpful addition for both locals and tourists and easy to use for all users to discover various local restaurants and try new and traditional dishes.