



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.



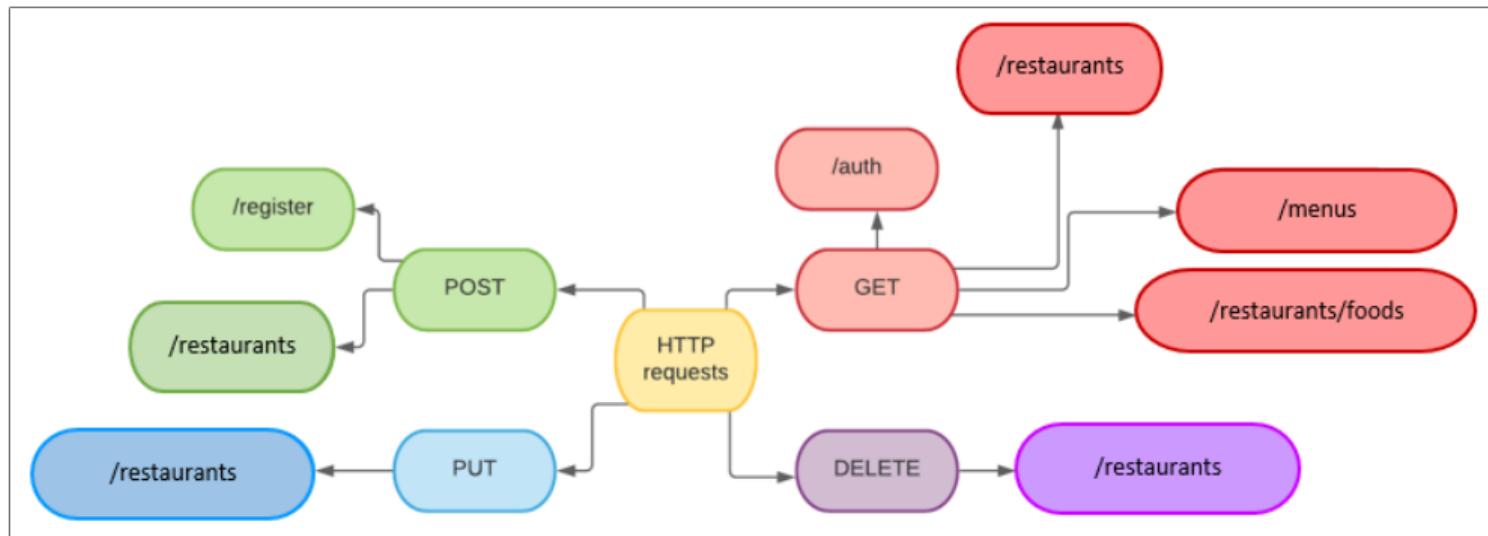


What is RestoConnect ?

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

The screenshot shows a dark red background with a white rectangular overlay containing the text "Tunisian Restaurant REST API". At the top right, there are two orange rounded rectangular buttons labeled "Login" and "Sign Up". At the bottom, a dark grey footer bar contains the text "Project By Mariem Ferchichi BA/IT @TBS".

Tunisian
Restaurant
REST API

Login Sign Up

Project By Mariem Ferchichi BA/IT @TBS



What is RestoConnect ?

Interface



Log In

username
password

Log in

Not registered yet? [Sign Up](#)

Project By Mariem Ferchichi BA/IT @TBS

This screenshot shows the login interface. It features a white rectangular form with two input fields: 'username' and 'password'. Below the fields is a red rounded rectangle containing the text 'Log in'. At the bottom left of the form, there is a blue link 'Not registered yet? [Sign Up](#)'. The entire interface is set against a dark grey background.

Sign Up

Username
Please enter a username

Password
Please enter a password

Sign up

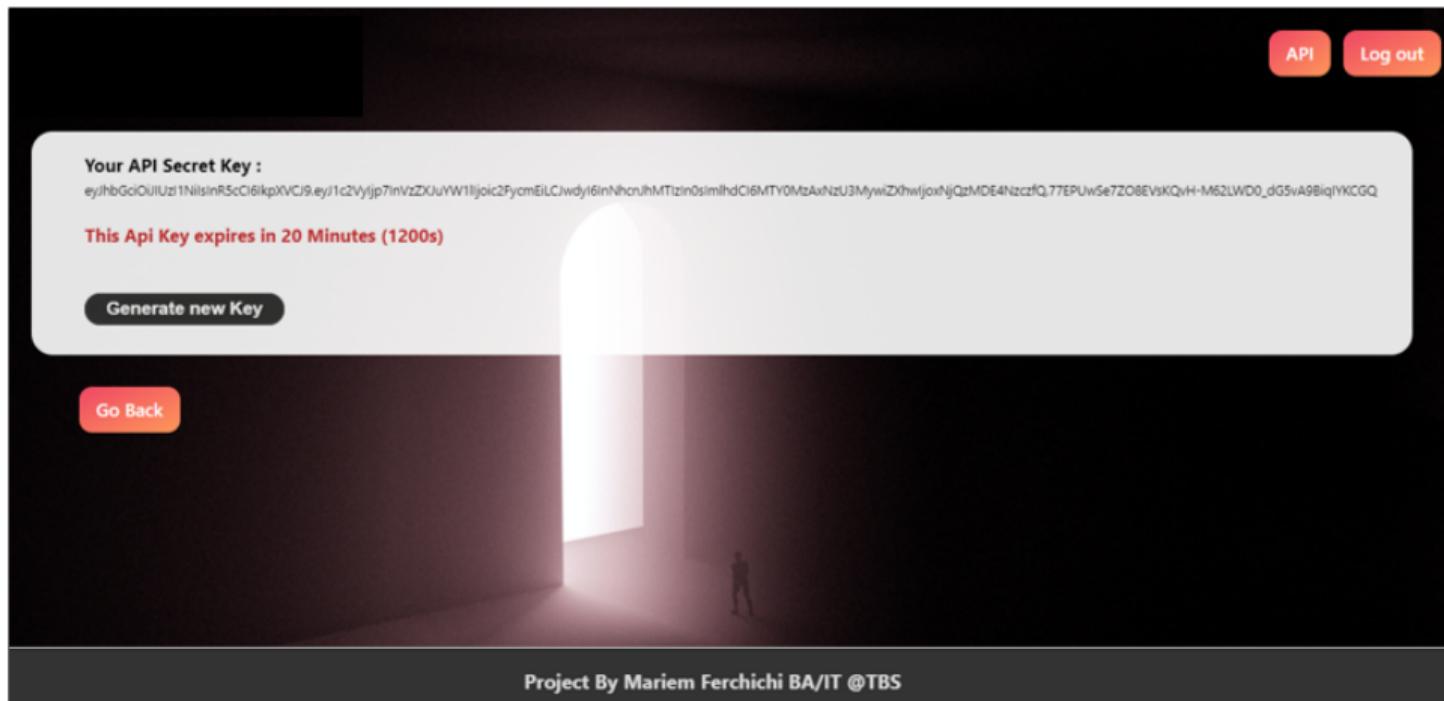
Already have an account? [Log in](#)

Project By Mariem Ferchichi BA/IT @TBS

This screenshot shows the sign-up interface. It features a white rectangular form with two input fields: 'Username' and 'Password'. Both fields have placeholder text: 'Please enter a username' and 'Please enter a password'. Below the fields is a red rounded rectangle containing the text 'Sign up'. At the bottom left of the form, there is a blue link 'Already have an account? [Log in](#)'. The entire interface is set against a dark grey background.

What is RestoConnect ?

Interface



Your API Secret Key :

```
eyJhbGciOiUzI1NilsinR5cIi6ikpXVCJ9eyJ1c2Vyljp7InVzZXJuYW1ljoic2FycmEiLCjwdyI6inNhcnJhMTizin0simlhCj6MTY0MzAxNzU3MywiZXhwIjoxNjQzMDE4NzczfQ.77EPUwSe7ZO8EVsKQvH-M62LWD0_dG5vA9BiqjYKCGQ
```

This Api Key expires in 20 Minutes (1200s)

[Generate new Key](#)

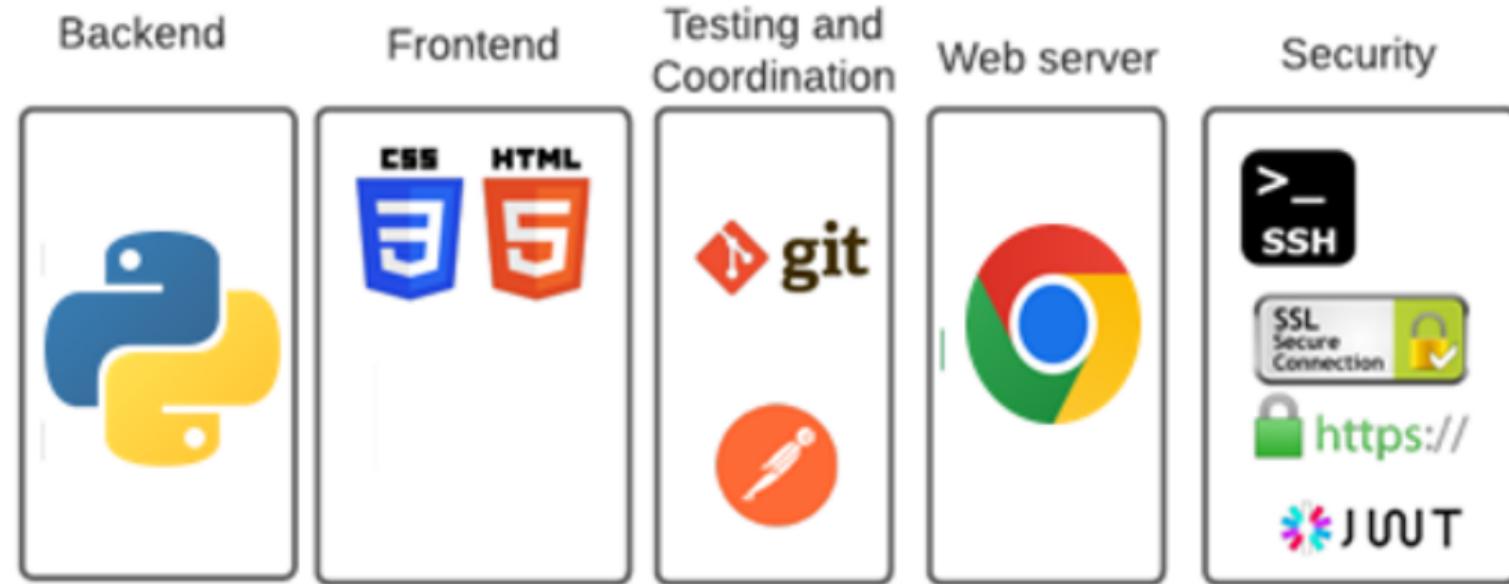
[Go Back](#)

API Log out

Project By Mariem Ferchichi BA/IT @TBS

What is RestoConnect ?

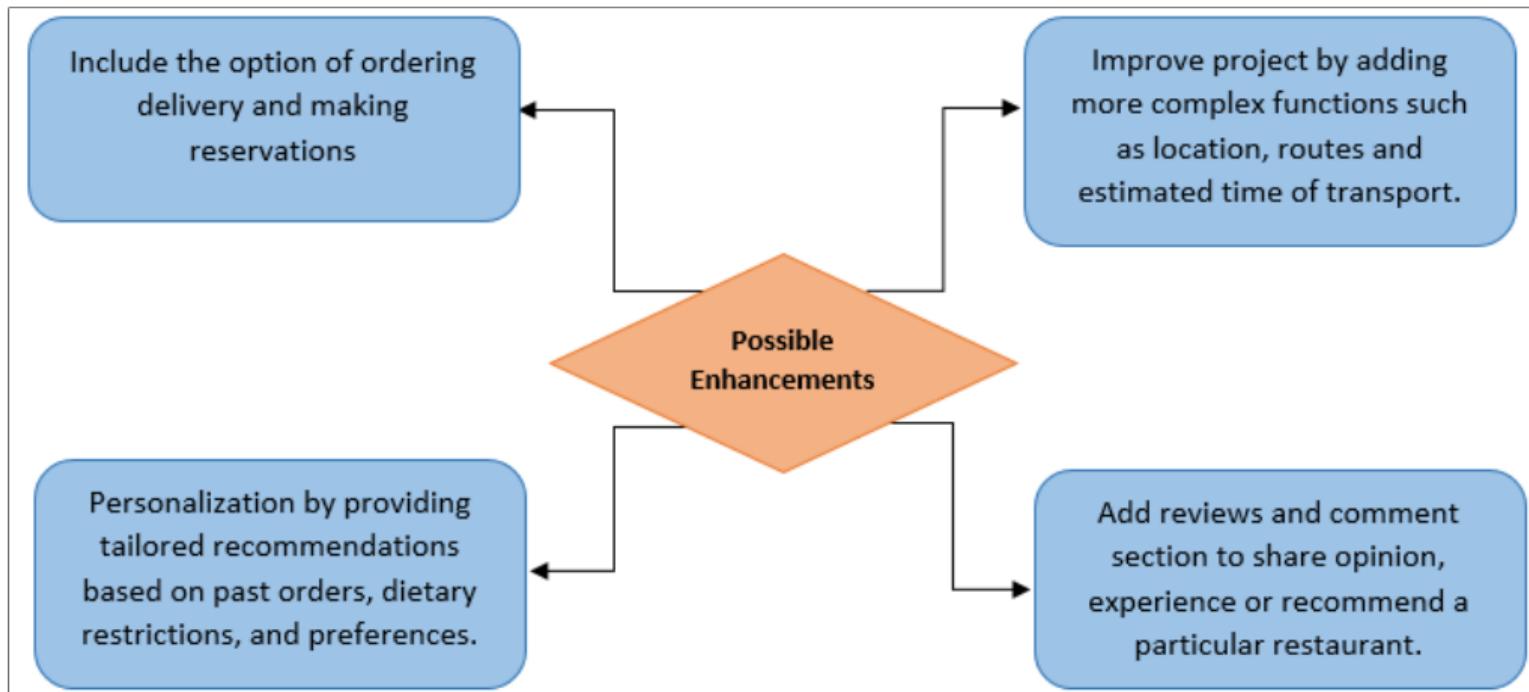
Technically



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_key))
    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_key))
    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



```
❸ security.py > authenticate
 1 from flask import Flask, render_template, request, jsonify, session
 2 from flask_session import Session
 3 import psycopg2
 4 import jwt
 5 from uuid import uuid4
 6 from router import router
 7
 8 app = Flask(__name__)
 9 app.config['SESSION_TYPE'] = 'filesystem'
10 app.config['SECRET_KEY'] = str(uuid4())
11 Session(app)
12
13
14 # SIGNUP PAGE
15 @app.route("/signup")
16 def signup():
17     |    return render_template('signup')
18
19 # LOGIN PAGE
20 @app.route("/login")
21 def login():
22     |    return render_template('login')
23
24 # LOGIN (get a token)
25 @app.route('/auth', methods=['GET'])
26 def authenticate():
27     username = request.args.get('username')
28     password = request.args.get('password')
29
30     try:
31         cursor = pool.cursor()
32         cursor.execute("SELECT * FROM users WHERE username = %s", (username,))
```

```
❸ config.py > ...
 1 class BaseConfig:
 2     SQLALCHEMY_ECHO = False
 3     SQLALCHEMY_TRACK_MODIFICATIONS = True
 4     SQLALCHEMY_DATABASE_URI = "sqlite:///app.db"
 5     REDIS_HOST = "localhost"
 6     REDIS_PASSWORD = "61a62a841f54d08ec165faf2ad20544e9f6fb8d37d10bf429b6e9f6e2807e0c4"
 7     REDIS_PORT = 6379
 8
 9
10 class DevelopmentConfig(BaseConfig):
11     SQLALCHEMY_DATABASE_URI = "sqlite:///dev.db"
12
13 class TestingConfig(BaseConfig):
14     SQLALCHEMY_DATABASE_URI = "sqlite:///test.db"
15
16 class ProductionConfig(BaseConfig):
17     SQLALCHEMY_DATABASE_URI = "sqlite:///base.db"
18
19 PresentConfig = BaseConfig
20
21
```

Database Configuration + Postman Snippets



```
* db.py > ...
1 import psycopg2
2 from psycopg2 import pool
3
4 # Define database connection parameters
5 db_params = {
6     "user": "gyqhjwmg",
7     "password": "o1RD08yXC3tPEnG-2KoTv9tjEY42ELiW",
8     "database": "xwmxumoi",
9     "host": "arjuna.db.elephantsql.com",
10    "port": 5432
11 }
12
13 # Create a PostgreSQL connection pool
14 connection_pool = psycopg2.pool.SimpleConnectionPool(
15     minconn=1,
16     maxconn=10,
17     **db_params
18 )
19
20 # Function to get a connection from the pool
21 def get_connection():
22     return connection_pool.getconn()
23
24 # Function to release a connection back to the pool
25 def release_connection(conn):
26     connection_pool.putconn(conn)
27
28 # Function to close all connections in the pool
29 def close_all_connections():
30     connection_pool.closeall()
```

Params	Authorization	Headers (10)	Body	Pre-request Script	Tests	Settings
					<pre>1 pm.test("Response time is less than 200ms", function () { 2 pm.expect(pm.response.responseTime).to.be.below(500); 3 }); 4 pm.test("Status code is 200", function () { 5 pm.response.to.have.status(200); 6 });</pre>	

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.