# Awra9i - Tunisian Municipalities Document Management System (DMS) REST API

IT325 Web Services Final Project

by

Khalil Elachkham

January 2023

IT/BA Junior Student


Information Technology Major

**Tunis Business School**

Ben Arous,TUNISIA.

2021-2022

# Declaration of Academic Honesty

I declare that the work submitted and presented in this report is my own original work, in my own words, and any sources used have been properly cited.

I understand that the principales of academic honesty are of the utmost importance and that any form of plagiarism, cheating or fabricated source will not be tolerated.

Date: January 15th, 2023 Khalil ELachkham

# Abstract

The adoption of digital technologies in the world has reached a crucial stage today. Digitization in Tunisia has greatly accelerated in recent years, allowing it to be one of the fastest-growing sectors. Despite this progress, there are still areas where we lag behind.

A clear and irrefutable example is the management of documents in municipalities. This lack of digitisation in document management causes inefficiency and inaccuracy in the process of storing and retrieving important documents related to civilians. Awra9i, our municipality document management system, aims to address this issue by providing an API platform for municipalities to effectively store, manage, and retrieve these documents.

Keywords - Digitization, Tunisia, Municipalities, Document management, Awra9i.

# Contents

# Chapter 1

# Introduction

My final project for the IT325 Web Services course this semester consists of a RESTful API developed using Flask, Pymongo, JWT, and it has been tested on Insomnia. This API uses MongoDB as the database with the implementation of PyMongo.

This program implements all CRUD operations and uses JWT token authentication for added security.

This API was designed to address the lack of digitization in document management in municipalities, which causes inefficiency and inaccuracy in the process of storing and retrieving important documents.

its main functionality is providing civilians-related data and documents.

# Chapter 2

# A summary of the work accomplished

## 2.1 Python/Flask Contribution

**Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. [1]**

**Flask is a web framework, it's a Python module that lets you develop web applications easily. It's has a small and easy-to-extend core: it's a microframework that doesn't include an ORM (Object Relational Manager) or such features. [2]**

I have used the flask framework for Python to develop my app.

Implementation code :

```
from flask import Flask
```

## 2.2 JWT Contribution

**JSON Web Token or JWT is an open standard to create tokens This standard has become quite popular since it's very effective for Web Apps like Google APIs, where after the user authentication you make API requests. JSON Web Token is a type of token that includes a structure, which can be decrypted by the server that allows you to authenticate the identity of the user of that application. [3]**

I have used JSON Web Token authentication to secure the API access. Implementation code : And i have used the function token required to require the token generated :

```
import jwt
from jwt.exceptions import ExpiredSignatureError, InvalidSignatureError, DecodeError
```

```
def token_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        token = None
        if 'x-access-token' in request.headers:
            token = request.headers['x-access-token']

        if not token:
            return jsonify({'message': 'Token is missing!'}), 401

        try:
            data = jwt.decode(token, current_app.config['SECRET_KEY'], verify=True, algorithms=['HS256'])
        except ExpiredSignatureError:
            return jsonify({'message': 'Token expired!'}), 401
        except jwt.exceptions.InvalidSignatureError:
            return jsonify({'message': 'Token is invalid!'}), 401

        return f(*args, **kwargs)

    return decorated
```

## 2.3 All the HTTP Methods used

### 2.3.1 GET Requests

#### GET /persons

Get a list of the persons with it related docs.

```
@blp.route("/get_persons", methods=["GET"])
@token_required
def get_all_persons():
    persons = list(mongo.db.persons.find())
    if len(persons) == 0:
        return jsonify({"message": "No persons found"}), 404
    for person in persons:
        person["_id"] = str(person["_id"])
    return jsonify(persons)
```

**GET /person/id**

Get a specified person by his object id in the database

```
@blp.route("/get_person/<id>", methods=["GET"])
@token_required
def get_person(id):
    if not ObjectId.is_valid(id):
        return jsonify({"message": "Invalid id"}), 400
    person = mongo.db.persons.find_one({"_id": ObjectId(id)})
    if person:
        person["_id"] = str(person["_id"])
        return jsonify(person)
    else:
        return jsonify({"message": "Person not found"}), 404
```

### 2.3.2 POST Requests

**POST /signin**

Register in the API after specifying a valid email and password in the body of the request using a json format and provide an access token.

```
@blp.route('/signin', methods=['POST'])
def signin():
    email = request.json['email']
    password = request.json['password']
    user = mongo.db.users.find_one({'email': email})
    if user:
        return jsonify({'error': 'email already used'}), 401
    hashed_password = generate_password_hash(password)
    mongo.db.users.insert_one({"email": email, "password": hashed_password})


    exp = datetime.utcnow() + timedelta(minutes=480)
    token = jwt.encode({"email": email, "exp": exp}, current_app.config['SECRET_KEY'])
    return jsonify({'user created here is your token': token}), 200
```

**POST /login**

Generate for user existing in the database an access token.

```
@blp.route('/login', methods=['POST'])
def login():
    email = request.json['email']
    password = request.json['password']

    # Verify email and password
    user = mongo.db.users.find_one({'email': email})
    if not user or not check_password_hash(user["password"], password):
        return jsonify({'error': 'Invalid email or password'}), 401

    exp = datetime.utcnow() + timedelta(minutes=480)
    token = jwt.encode({"email": email, "exp": exp}, current_app.config["SECRET_KEY"])
    return jsonify({'token': token }), 200
```

**POST /person**

Add a new person by specifying the name and the id.

```
@blp.route("/add_person", methods=["POST"])
@token_required
def add_person():
    # Extract person data from request
    person = request.json
    person_id = mongo.db.persons.insert_one(person).inserted_id
    return jsonify({"message": "Person added successfully"}), 201
```

### 2.3.3 PUT Requests

**PUT /person/id**

Update a person's name, id, and set of documents.

```python
@blp.route("/update_person/<id>", methods=["PUT"])
@token_required
def update_person(id):
    if not ObjectId.is_valid(id):
        return jsonify({"message": "Invalid id"}), 400
    person = mongo.db.persons.find_one({"_id": ObjectId(id)})
    if person:
        update_data = {}
        if 'name' in request.form:
            update_data["name"] = request.form.get("name")
        if 'id' in request.form:
            update_data["id"] = request.form.get("id")
        if 'birth_certificate' in request.form:
            update_data["birth_certificate"] = request.form.getlist("birth_certificate")
        if 'id_card' in request.form:
            update_data["id_card"] = request.form.getlist("id_card")
        if 'passeport' in request.form:
            update_data["passeport"] = request.form.getlist("passeport")
        if 'other_docs' in request.form:
            update_data["other_docs"] = request.form.getlist("other_docs")
        mongo.db.persons.update_one(
            {"_id": ObjectId(id)},
            {"$set": update_data }
        )
        return jsonify({"message": "Person updated successfully!"}), 200
    else:
        return jsonify({"message": "Person not found"}), 404
```

### 2.3.4 DELETE Requests

**DELETE /person/id**

Delete a specific person in the database.

```python
@blp.route("/delete_person/<id>", methods=["DELETE"])
@token_required
def delete_person(id):
    if not ObjectId.is_valid(id):
        return jsonify({"message": "Invalid id"}), 400
    person = mongo.db.persons.find_one({"_id": ObjectId(id)})
    if person:
        mongo.db.persons.delete_one({"_id": ObjectId(id)})
        return jsonify({"message": "Person deleted successfully!" })
    else:
        return jsonify({"message": "Person not found"}), 404
```

**DELETE /user**

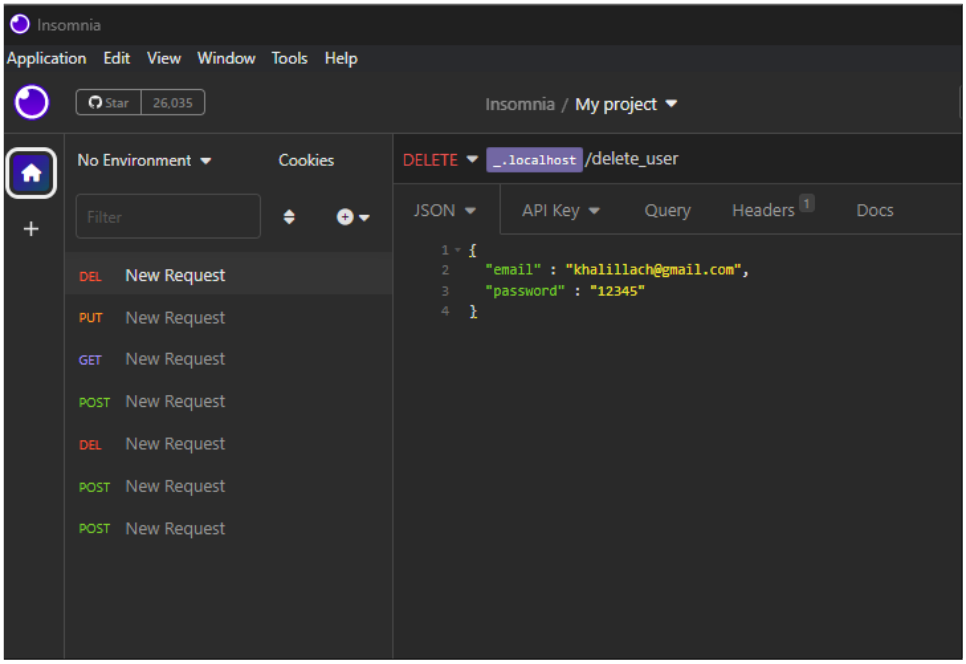
Delete a specific user in the database by providing in the body a json format of the user's email and password.

```
@blp.route('/delete_user', methods=['DELETE'])
@token_required
def deleteuser():
    email = request.json['email']
    password = request.json['password']
    user = mongo.db.users.find_one({'email': email})
    if not user or not check_password_hash(user["password"], password):
        return jsonify({'error': 'Invalid email or password'}), 401
    if user:
        mongo.db.users.delete_one({"email": email})
        return jsonify({"message":"user deleted successfully!" })
    else:
        return jsonify({"message": "user not found"}), 404
```

## 2.4 Insomnia Contribution

**Insomnia is a free, cross-platform desktop application that simplifies the interaction and design of HTTP-based APIs. Insomnia combines an easy-to-use interface with advanced features like authentication wizards, code generation, and environment variables.. [4]**

I have used Insomnia for the automatic testing of my API requests.



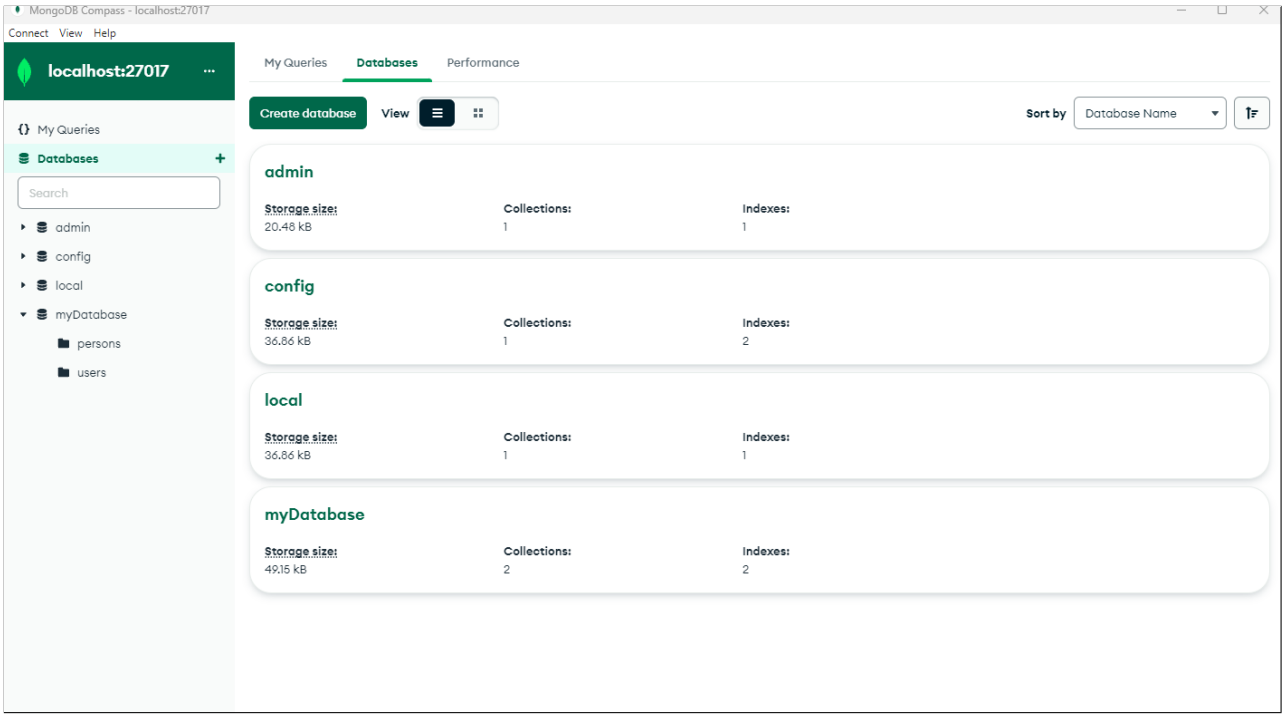

## 2.5 Mongodb Contribution

**MongoDB is an open source NoSQL database management program. NoSQL is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document-oriented information, store or retrieve information. [5]**

The app is hosted using MongoDB database.

## 2.6 Mongodb compass Contribution

**MongoDB Compass is a powerful GUI for querying, aggregating, and analyzing your MongoDB data in a visual environment. Compass is free to use and source available, and can be run on macOS, Windows, and Linux. [6]**




## 2.7 Database Structure

The database consists of 2 collections .

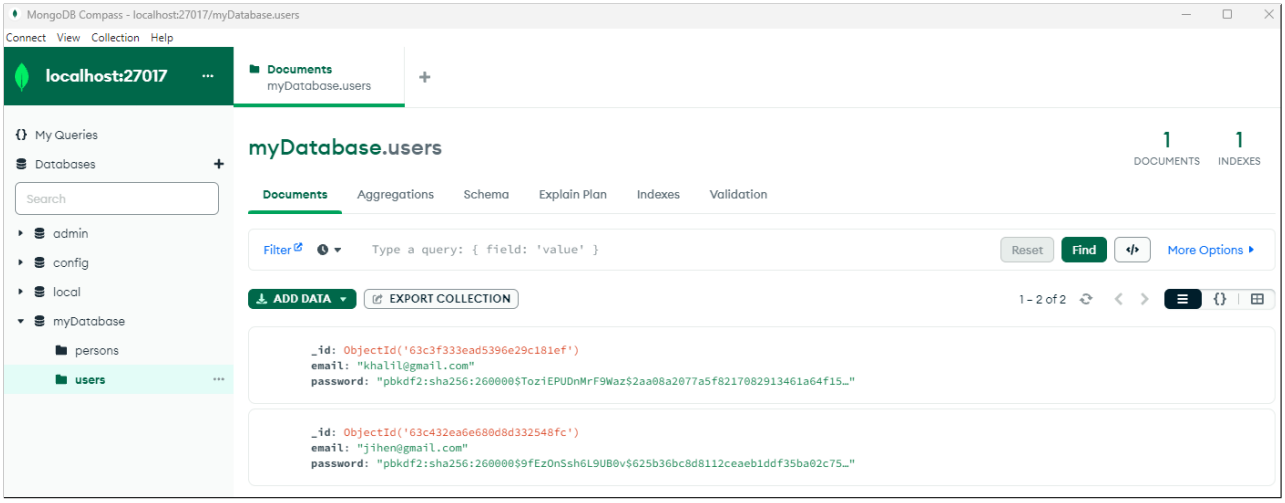
### 2.7.1 collections

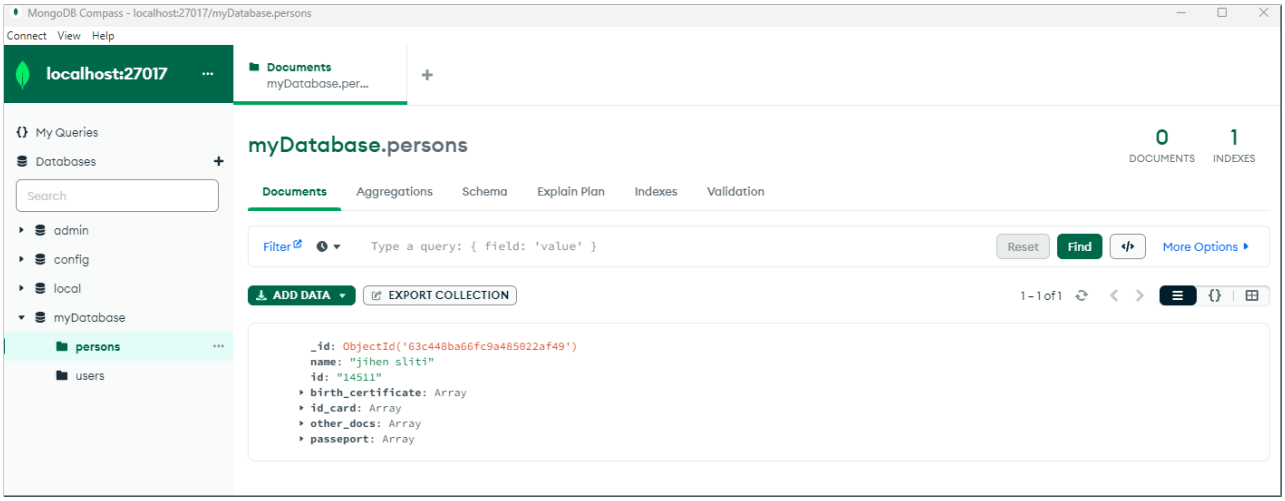
**collection "users"**

this collection stores all the users emails and passwords

**collection "persons"**

this collection stores all the persons with their id and set of documents


MongoDB Compass - localhost:27017/myDatabase.users
Connect View Collection Help
localhost:27017
Documents
myDatabase.users
My Queries
Databases
Search
admin
config
local
myDatabase
persons
users
myDatabase.users
1
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
Filter
Type a query: { field: 'value' }
Reset
Find
More Options
ADD DATA
EXPORT COLLECTION
1 - 2 of 2
_id: ObjectId('63c3f333ead5396e29c181ef')
email: "khalil@gmail.com"
password: "pbkdf2:sha256:260000$ToziEPUDnMrF9Waz$2aa08a2077a5f8217082913461a64f15…"
_id: ObjectId('63c432ea6e680d8d332548fc')
email: "jihen@gmail.com"
password: "pbkdf2:sha256:260000$9fEzOnSsh6L9UB0v$625b36bc8d8112ceaeb1ddf35ba02c75…"


MongoDB Compass - localhost:27017/myDatabase.persons
Connect View Help
localhost:27017
Documents
myDatabase.per...
My Queries
Databases
Search
admin
config
local
myDatabase
persons
users
myDatabase.persons
0
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
Filter
Type a query: { field: 'value' }
Reset
Find
More Options
ADD DATA
EXPORT COLLECTION
1 - 1 of 1
_id: ObjectId('63c448ba66fc9a485022af49')
name: "jihen sliti"
id: "14511"
birth_certificate: Array
id_card: Array
other_docs: Array
passeport: Array

# Chapter 3

# Conclusion

The purpose of this project is to build an API that serves a social purpose. Digitization in Tunisia is slowly developing and taking part in different sectors. As someone who is interested in improving the interactions between the government and the public, I have brainstormed and developed an idea that I believe will be innovative.

This project aims to facilitate the process of managing documents in municipalities as well as speeding it up. While this project may be considered as difficult to implement, I truly believe that it could be very influential in potentially implementing technology in every public sector in Tunisia.

I owe my knowledge and access to the necessary information to my professor DR Montassar Ben Messaoud who guided us through every step of this project.

*Khalil Elachkham*

# Appendix A

# Response examples

## A.1 GET Requests

### A.1.1 GET /persons


```
[
        {
                "_id": "63c448ba66fc9a485022af49",
                "birth_certificate": [
                        "missing"
                ],
                "id": "14511",
                "id_card": [
                        "https://drive.google.com/file/
                    d/1FjKMKFYG4gtkuuZbIwNXRu3pimF
                    mIRB9/view?usp=sharing"
                ],
                "name": "khalil lachkham",
                "other_docs": [
                        "https://drive.google.com/file/
                    d/1FjKMKFYG4gtkuuZbIwNXRu3pim
                    FmIRB9/view?usp=sharing"
                ],
```

```
        "passeport": [
                "https://drive.google.com/file/
            d/1FjKMKFYG4gtkuuZbIwNXRu3pim
            FmIRB9/view?usp=sharing"
        ]
    },
    {
        "_id": "63c4517f2ef334db18484671",
        "birth_certificate": [
                "missing"
        ],
        "id": "1477",
        "id_card": [
                "https://drive.google.com/file/
            d/1FjKMKFYG4gtkuuZbIwNXRu3pim
            FmIRB9/view?usp=sharing"
        ],
        "name": "jihen sliti",
        "passeport": [
                "https://drive.google.com/file/
            d/1FjKMKFYG4gtkuuZbIwNXRu3pim
            FmIRB9/view?usp=sharing"
        ]
    }
]
```


### A.1.2 GET /person/id


```
{
    "_id": "63c4517f2ef334db18484671",
    "birth_certificate": [
            "missing"
    ],
```

```
        "id": "1477",
        "id_card": [
                "https://drive.google.com/file/
            d/1FjKMKFYG4gtkuuZbIwNXRu3pim
            FmIRB9/view?usp=sharing"
        ],
        "name": "khalil lachkham",
        "passeport": [
                "https://drive.google.com/file/
            d/1FjKMKFYG4gtkuuZbIwNXRu3pim
            FmIRB9/view?usp=sharing"
        ]
}
```


## A.2 POST Requests

### A.2.1 POST /signin


```
{
        "user created here is your token":
       "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJlbWFpbCI6ImppaGVuQGdtYWlsLmNvbSIsImV4cCI6MTY3MzgzMTI3NH0.
        u1DxKIUIdGozBzigegEIFw2CozYlU7rN6BUIf_QNipE"
}
```


### A.2.2 POST /login


```
{
"token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
 eyJlbWFpbCI6ImtoYWxpbEBnbWFpbC5jb20iLCJleHAiOjE2NzM4Mzk2NzF9.
 ctuf24MJZZ5MVc54OT320BZ72GC8e3MU68BFcgMYq74"
}
```

### A.2.3 POST /addperson


```
{
        "message": "Person added successfully"
}
```


## A.3 PUT Requests

### A.3.1 PUT /person/id


```
{
        "message": "Person updated successfully!"
}
```


## A.4 DELETE Requests

### A.4.1 DELETE /person/id


```
{
        "message": "Person deleted successfully!"
}
```


### A.4.2 DELETE /user/


```
{
        "message": "user deleted successfully!"
}
```

# Bibliography


[1] "What is python?." https://www.python.org/doc/essays/blurb/. [Online; accessed 15-Jan-2023].

[2] "What is flask python." https://pythonbasics.org/what-is-flask-python/. [Online; accessed 15-Jan-2023].

[3] "Jwt." https://4geeks.com/lesson/what-is-JWT-and-how-to-implement-with- [Online; accessed 15-Jan-2023].

[4] "Api testing tools: Insomnia." https://abstracta.us/blog/software-testing/api-testing-tools-insomnia/#:~:text=Insomnia%20is%20a%20free%2C%20cross,code%20generation%2C%20and%20environment%20variables. [Online; accessed 15-Jan-2023].

[5] "Mongodb." https://www.techtarget.com/searchdatamanagement/definition/MongoDB. [Online; accessed 15-Jan-2023].

[6] "What is mongodb compass?." https://www.mongodb.com/docs/compass/current/#:~:text=MongoDB%20Compass%20is%20a%20powerful,macOS%2C%20Windows%2C%20and%20Linux. [Online; accessed 15-Jan-2023].