WORKSHOP N° 3
SOFTWARE ENGINEERING II

Presented by:

Michael Stiven Betancourt Gelves

Professor:
Carlos Andres Sierra Virguez

November 20th 2025

Universidad Nacional de Colombia
Engineering Faculty
2025

Workshop 3: Project Implementation and Integration

Overview

This document outlines the deliverables and requirements for **Workshop 3**. The objective is to finalize the database implementation, develop backend services, create a web frontend, and ensure successful integration across all components.

1. Database Implementation

- **SQL Scripts/Migrations:** Each of the microservices presented in the previous workshops are designed to have their own database with one or more tables to make it easier to manage each microservice. We are using PostgreSQL as the primary DB for all Microservices.

- **Sample Data:** Below is a set of sample data for each of the databases:

  Auth Service DB:

id	name	email	username	password_hash	role	reset_token	reset_token_expiry	created_at
2	Juan Alvarez	jua…le.com	juancho	$2a$...zmoe	ADMIN			2025-...2
3	Pepito Perez	pep…le.com	pepito	$2a$...rxcm	RESTAURANT_MANAGER			2025-...5
1	Cristhian Cely	cri…le.com	cristancho	$2a$...aT24.	ADMIN			2025-...9

Geo Service DB:

id	name	type	latitude	longitude	geometry
1	Pizza Planet	RESTAURANT	40,73061	-73,935242	POINT (-73.935242 40.73061)
2	McDonalds 53	RESTAURANT	52,73061	-72,935242	POINT (-72.935242 52.73061)
3	McDonalds Salitre	RESTAURANT	52,74757	-72,952145	POINT (-72.952145 52.74757)

- **Documentation:** All schemas are fully documented in the `README.md` file.

2. Backend Services

- **Source Code:** Source code for frontend and backend are located in ~/Click&MunchApp

- **Structure:**
  Frontend can be located in ~/Click&MunchApp/frontend
  Backtend can be located in ~/Click&MunchApp/backend

as follows:

```
./Click&MunchApp/

|

├── backend/

|       ├── src/

|       ├── .env

|       └── README.md

|
├── frontend/
|       ├── public/
|       ├── src/
|       └── package.json
|
├── .gitignore # Ignores node_modules in both folders
└── README.md # Main project documentation
```

- **Configuration:** For database connection, there is a YAML file created to create the databases:

```yaml
version: '3.8'


services:
 auth-db:
   image: postgres:16
   container_name: auth-db
   restart: always
   environment:
     POSTGRES_DB: auth_db
     POSTGRES_USER: mike
     POSTGRES_PASSWORD: secret
   ports:
     - "5433:5432"
   volumes:
     - auth_data:/var/lib/postgresql/data
```

```yaml
restaurant-db:

  image: postgres:16

  container_name: restaurant-db

  restart: always

  environment:

    POSTGRES_DB: restaurant_db

    POSTGRES_USER: mike

    POSTGRES_PASSWORD: secret

  ports:

    - "5434:5432"

  volumes:

    - restaurant_data:/var/lib/postgresql/data



geo-db:

  image: postgis/postgis:16-3.4

  container_name: geo-db

  restart: always

  environment:

    POSTGRES_DB: geo_db

    POSTGRES_USER: mike

    POSTGRES_PASSWORD: secret

  ports:

    - "5435:5432"

  volumes:

    - geo_data:/var/lib/postgresql/data
```

```yaml
volumes:

 auth_data:

 restaurant_data:

 geo_data:
```

This YAML file creates a series of containers based on public postgres images available in docker.

- **API Documentation:** The following is the documentation for all endpoints for the available microservices:
  - AuthService Endpoints:
    - POST Register
      localhost:8081/api/auth/register

      ### Request Headers
      Content-Type: application/json

      ### Body
      ```
      {
       "name": "Pepito Perez",
       "email": "pepo@example.com",
       "username": "pepito",
       "password": "789456",
       "role": "RESTAURANT_MANAGER"
      }
      ```

    - POST Login
      localhost:8081/api/auth/login

      ### Request Headers
      Content-Type: application/json

      ### Body
      ```
      {
       "username": "juancho",
       "password": "789456"
      }
      ```

- POST Reset
  localhost:8081/auth/password-reset/request

  ### Request Headers
  Content-Type: application/json

  ### Body
  ```
  {
   "email": "cristhian@example.com"
  }
  ```

- POST Confirm
  localhost:8081/auth/password-reset/confirm

  ### Request Headers
  Content-Type: application/json

  ### Body
  ```
  {
     "resetToken":
  "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJjcmlzdGFuY2hvIiwiaWF0IjoxNzYzNj
  g2NzEzLCJleHAiOjE3NjM2OTAzMTN9.aYlz2bMuFWPlR25Ydts0TBPNg
  bHAZ41W-NZEwVpuR9M",
     "newPassword": "jamaica"
  }
  ```

- GeoService Endpoints:

  - POST Add Location
    localhost:8083/api/geo/locations

    ### Request Headers
    Content-Type: application/json

    ### Body
    ```
    {
     "name": "McDonalds Salitre",
     "type": "RESTAURANT",
     "latitude": 52.74757,
     "longitude": -72.952145
    }
    ```

  - POST Find Nearby
    localhost:8083/api/geo/nearby

Request Headers
Content-Type: application/json

Body
```
{
  "latitude": 52.75214,
  "longitude": -72.886547,
  "radiusInKm": 5
}
```

3. Web Frontend

- Due to some inconveniences with the resources available, there was a delay in this part of the project, however, this important part is being worked on to provide the best possible user experience.

4. Unit Testing

- **Backend Tests:** All tests are included in the code snippets inside the directory

  We used Mockito to conduct unit testing

Results:

Since the available microservices are developed in Java with SpringBoot, Unit Tests are built in the project using framework decorators for testing and Mockito library. All tests were created to verify special conditions and REST API calls.

```
Task :test
BUILD SUCCESSFUL in 4s
4 actionable tasks: 4 executed
```

5. Integration Evidence

- **Interaction Logic:** Since the front-end is still under construction, the interaction logic between frontend and backend is still to be documented.

References

EngAndres. (n.d.). *unal_public* [Folder: Software Engineering 2_Morning (G3)/slides]. GitHub. https://github.com/EngAndres/unal_public/tree/main/Software%20Engineering%202_Morning%20(G3)/slides