Software Engineering Seminar

ProTicket

Backend Implementation, Database Connection, REST API Integration Unit Testing

---

Tutor: Carlos Andres Sierra

Students:

- Juan Daniel Vanegas Mayorquín — *Cod:* 20222020077
- Juan Sebastian Villalba Roa — *Cod:* 20201020066
- Daniel Santiago Arcila Martínez — *Cod:* 20191020075

Universidad Francisco José de Caldas
School of Engineering
Systems Engineering

Bogotá D.C. 2025

Source Code for Backends

Auth Backend - Java Spring Boot

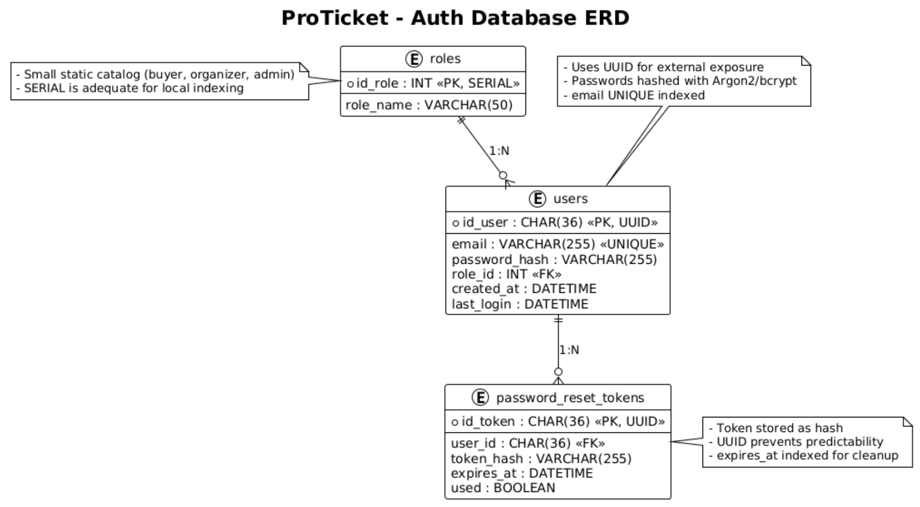

ProTicket - Auth Database ERD
roles
id_role : INT «PK, SERIAL»
role_name : VARCHAR(50)
- Small static catalog (buyer, organizer, admin)
- SERIAL is adequate for local indexing
- Uses UUID for external exposure
- Passwords hashed with Argon2/bcrypt
- email UNIQUE indexed
1:N
users
id_user : CHAR(36) «PK, UUID»
email : VARCHAR(255) «UNIQUE»
password_hash : VARCHAR(255)
role_id : INT «FK»
created_at : DATETIME
last_login : DATETIME
1:N
password_reset_tokens
id_token : CHAR(36) «PK, UUID»
user_id : CHAR(36) «FK»
token_hash : VARCHAR(255)
expires_at : DATETIME
used : BOOLEAN
- Token stored as hash
- UUID prevents predictability
- expires_at indexed for cleanup

Figure 1: ProTicket Auth Database Diagram.

The **Auth Service** was developed using **Spring Boot (Java 17)** with layered architecture principles to manage user authentication, registration, and password reset securely through JWT-based tokens.

- **Config:** Security and token configuration classes, including `SecurityConfig`, `JwtService`, and password encoder setup.
- **Controller:** Contains REST controllers (`AuthController`) for handling login, registration, and password reset endpoints.
- **DTO:** Defines request/response structures for `RegisterRequest`, `LoginRequest`, `PasswordResetStartRequest`, and `PasswordResetConfirmRequest`.
- **Entity:** Contains persistent entities such as `User` and `Role`, mapped to the authentication database.
- **Repository:** Provides access to persistence layer via `UserRepository`.
- **Service:** Implements the business logic for authentication, JWT generation, validation, and password reset.
- **Test:** Includes `JUnit 5` tests validating controller and service layers.

Database Configuration

Connection settings for the Auth Service are defined in the Spring Boot configuration file:

```yaml
spring:
  datasource:
    url: jdbc:mysql://mysql_auth:3306/${MYSQL_DATABASE}?useSSL=false&a
    ↪  llowPublicKeyRetrieval=true&serverTimezone=UTC
    username: ${MYSQL_USER}
    password: ${MYSQL_PASSWORD}
    driver-class-name: com.mysql.cj.jdbc.Driver

  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQL8Dialect

server:
  port: 8080
  servlet:
    context-path: /auth

jwt:
  secret: ${JWT_SECRET}
  expiration: 3600000
```

—

REST API Documentation

The following endpoints are defined under the base path `/auth`.

0.1 POST /auth/register

Description: Registers a new user. **Request Schema:** `RegisterRequest`

```json
{
  "email": "user@example.com",
  "password": "string",
  "role": "string"
}
```

Response (200 OK):

```
{
  "token": "string",
  "userId": "string",
  "role": "string"
}
```

—

0.2 POST /auth/login

Description: Authenticates a user with email and password. **Request Schema:** `LoginRequest`

```
{
  "email": "user@example.com",
  "password": "securePassword"
}
```

Response (200 OK):

```
{
  "token": "string",
  "userId": "string",
  "role": "USER"
}
```

—

0.3 POST /auth/password/start

Description: Initiates password reset process by sending a token to the user's registered email. **Request Schema:** `PasswordResetStartRequest`

```
{
  "email": "user@example.com"
}
```

Response (200 OK):

```
{
  "message": "Password reset email sent successfully"
}
```

Response (404 Not Found):

```
{
  "error": "User not found"
}
```

—

0.4 POST /auth/password/confirm

Description: Confirms password reset with a valid token and new password.
Request Schema: `PasswordResetConfirmRequest`

```
{
  "token": "string",
  "newPassword": "newSecurePassword123"
}
```

Response (200 OK):

```
{
  "message": "Password reset successfully"
}
```

Response (400 Bad Request):

```
{
  "error": "Invalid or expired token"
}
```

—

Data Models (Schemas)

RegisterRequest

```
{
  "email": "string",
  "password": "string",
  "role": "string"
}
```

LoginRequest

```
{
  "email": "string",
  "password": "string"
}
```

JwtResponse

```
{
  "token": "string",
  "userId": "string",
  "role": "string"
}
```

PasswordResetStartRequest

```json
{
  "email": "string"
}
```

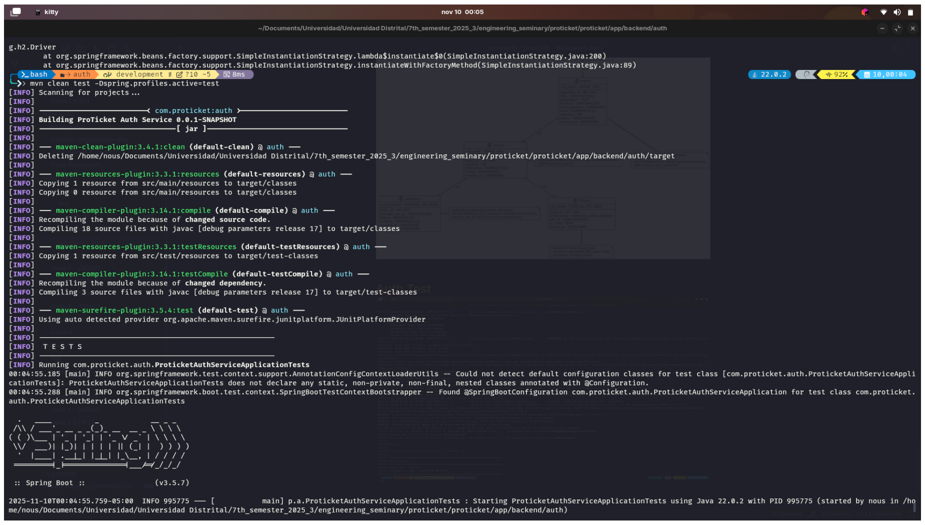
PasswordResetConfirmRequest

```json
{
  "token": "string",
  "newPassword": "string"
}
```

—

Unit Test Results and Code

Unit testing for the Auth Service was performed using **JUnit 5** and **MockMvc**. All controller endpoints were tested to ensure correct HTTP responses and JWT validation.


kitty
nov 10 00:05
~/Documents/Universidad/Universidad Distrital/7th_semester_2025_3/engineering_seminary/proticket/proticket/app/backend/auth

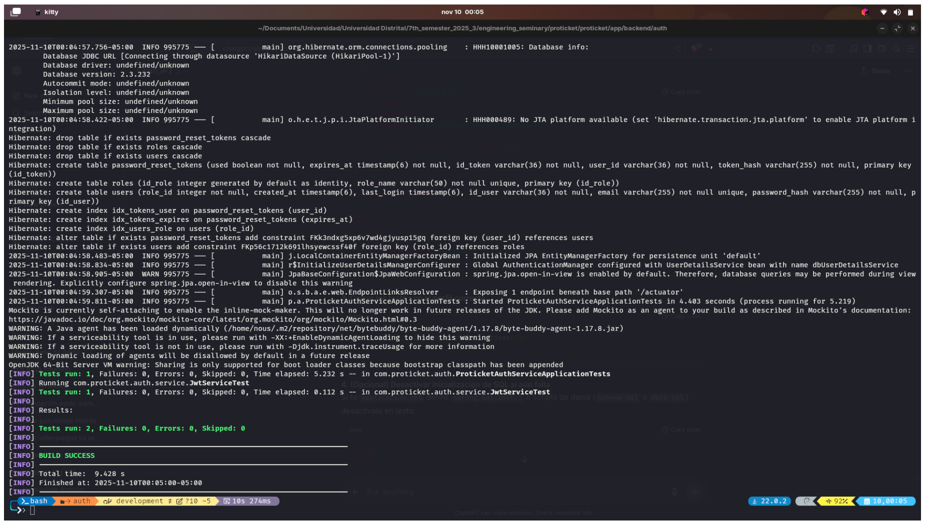
```
g.h2.Driver
        at org.springframework.beans.factory.support.SimpleInstantiationStrategy.lambda$instantiate$0(SimpleInstantiationStrategy.java:200)
        at org.springframework.beans.factory.support.SimpleInstantiationStrategy.instantiateWithFactoryMethod(SimpleInstantiationStrategy.java:89)
 bash  auth  development ≢ ?10 ~5  8ms                                                                    22.0.2   92%  10,00:04
╰─> mvn clean test -Dspring.profiles.active=test
[INFO] Scanning for projects...
[INFO]
[INFO] ─────────────────────────< com.proticket:auth >─────────────────────────
[INFO] Building ProTicket Auth Service 0.0.1-SNAPSHOT
[INFO] ────────────────────────────────[ jar ]─────────────────────────────────
[INFO]
[INFO] ── maven-clean-plugin:3.4.1:clean (default-clean) @ auth ──
[INFO] Deleting /home/nous/Documents/Universidad/Universidad Distrital/7th_semester_2025_3/engineering_seminary/proticket/proticket/app/backend/auth/target
[INFO]
[INFO] ── maven-resources-plugin:3.3.1:resources (default-resources) @ auth ──
[INFO] Copying 1 resource from src/main/resources to target/classes
[INFO] Copying 0 resource from src/main/resources to target/classes
[INFO]
[INFO] ── maven-compiler-plugin:3.14.1:compile (default-compile) @ auth ──
[INFO] Recompiling the module because of changed source code.
[INFO] Compiling 18 source files with javac [debug parameters release 17] to target/classes
[INFO]
[INFO] ── maven-resources-plugin:3.3.1:testResources (default-testResources) @ auth ──
[INFO] Copying 1 resource from src/test/resources to target/test-classes
[INFO]
[INFO] ── maven-compiler-plugin:3.14.1:testCompile (default-testCompile) @ auth ──
[INFO] Recompiling the module because of changed dependency.
[INFO] Compiling 3 source files with javac [debug parameters release 17] to target/test-classes
[INFO]
[INFO] ── maven-surefire-plugin:3.5.4:test (default-test) @ auth ──
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] ───────────────────────────────────────────────────────
[INFO]  T E S T S
[INFO] ───────────────────────────────────────────────────────
[INFO] Running com.proticket.auth.ProticketAuthServiceApplicationTests
00:04:55.185 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils -- Could not detect default configuration classes for test class [com.proticket.auth.ProticketAuthServiceAppli
cationTests]: ProticketAuthServiceApplicationTests does not declare any static, non-private, non-final, nested classes annotated with @Configuration.
00:04:55.288 [main] INFO org.springframework.boot.test.context.SpringBootTestContextBootstrapper -- Found @SpringBootConfiguration com.proticket.auth.ProticketAuthServiceApplication for test class com.proticket.
auth.ProticketAuthServiceApplicationTests

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.5.7)

2025-11-10T00:04:55.759-05:00  INFO 995775 --- [           main] p.a.ProticketAuthServiceApplicationTests : Starting ProticketAuthServiceApplicationTests using Java 22.0.2 with PID 995775 (started by nous in /ho
me/nous/Documents/Universidad/Universidad Distrital/7th_semester_2025_3/engineering_seminary/proticket/proticket/app/backend/auth)
```

Figure 2: JUnit test suite results for Spring Boot Auth Service.



```
2025-11-10T00:04:57.756-05:00  INFO 995775 --- [           main] org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
	Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
	Database driver: undefined/unknown
	Database version: 2.3.232
	Autocommit mode: undefined/unknown
	Isolation level: undefined/unknown
	Minimum pool size: undefined/unknown
	Maximum pool size: undefined/unknown
2025-11-10T00:04:58.422-05:00  INFO 995775 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform i
ntegration)
Hibernate: drop table if exists password_reset_tokens cascade
Hibernate: drop table if exists roles cascade
Hibernate: drop table if exists users cascade
Hibernate: create table password_reset_tokens (used boolean not null, expires_at timestamp(6) not null, id_token varchar(36) not null, user_id varchar(36) not null, token_hash varchar(255) not null, primary key
(id_token))
Hibernate: create table roles (id_role integer generated by default as identity, role_name varchar(50) not null unique, primary key (id_role))
Hibernate: create table users (role_id integer not null, created_at timestamp(6), last_login timestamp(6), id_user varchar(36) not null, email varchar(255) not null unique, password_hash varchar(255) not null, p
rimary key (id_user))
Hibernate: create index idx_tokens_user on password_reset_tokens (user_id)
Hibernate: create index idx_tokens_expires on password_reset_tokens (expires_at)
Hibernate: create index idx_users_role on users (role_id)
Hibernate: alter table if exists password_reset_tokens add constraint FKk3ndxg5xp6v7wd4gjyusp15gq foreign key (user_id) references users
Hibernate: alter table if exists users add constraint FKp56c1712k691lhsyewcssf40f foreign key (role_id) references roles
2025-11-10T00:04:58.483-05:00  INFO 995775 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-11-10T00:04:58.834-05:00  INFO 995775 --- [           main] r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManager configured with UserDetailsService bean with name dbUserDetailsService
2025-11-10T00:04:58.905-05:00  WARN 995775 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view
 rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2025-11-10T00:04:59.307-05:00  INFO 995775 --- [           main] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 1 endpoint beneath base path '/actuator'
2025-11-10T00:04:59.811-05:00  INFO 995775 --- [           main] p.a.ProticketAuthServiceApplicationTests : Started ProticketAuthServiceApplicationTests in 4.403 seconds (process running for 5.219)
Mockito is currently self-attaching to enable the inline-mock-maker. This will no longer work in future releases of the JDK. Please add Mockito as an agent to your build as described in Mockito's documentation:
https://javadoc.io/doc/org.mockito/mockito-core/latest/org.mockito/org/mockito/Mockito.html#0.3
WARNING: A Java agent has been loaded dynamically (/home/nous/.m2/repository/net/bytebuddy/byte-buddy-agent/1.17.8/byte-buddy-agent-1.17.8.jar)
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a future release
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 5.232 s -- in com.proticket.auth.ProticketAuthServiceApplicationTests
[INFO] Running com.proticket.auth.service.JwtServiceTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.112 s -- in com.proticket.auth.service.JwtServiceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  9.428 s
[INFO] Finished at: 2025-11-10T00:05:00-05:00
[INFO] ------------------------------------------------------------------------
 bash  auth  development ≢ ?10 ~5  10s 274ms                                                         22.0.2   92%  10,00:05
>
```

Figure 3: JUnit test suite results for Spring Boot Auth Service.

Business Logic Backend - Python Backend

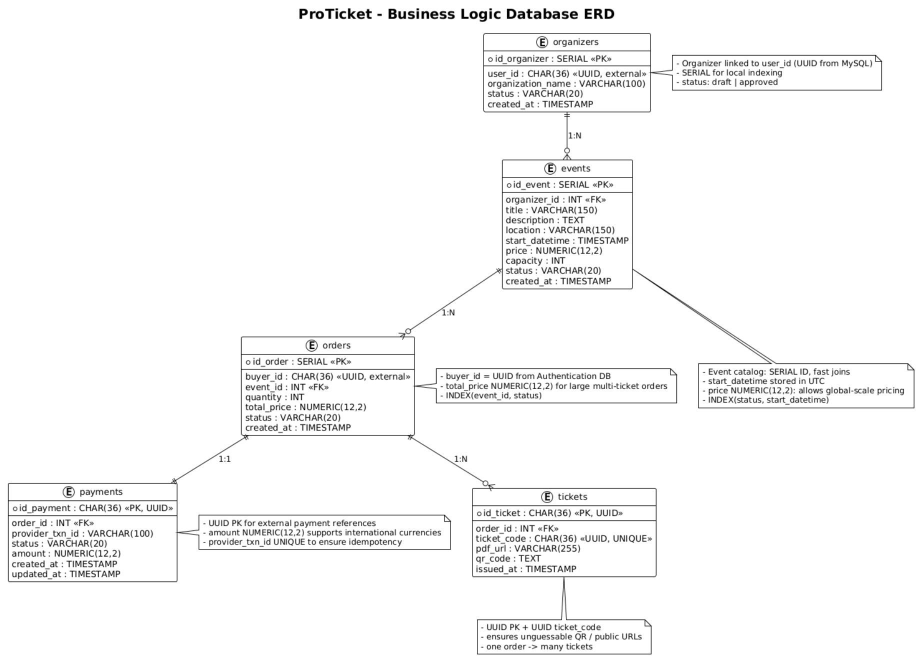

ProTicket - Business Logic Database ERD
organizers
id_organizer : SERIAL «PK»
user_id : CHAR(36) «UUID, external»
organization_name : VARCHAR(100)
status : VARCHAR(20)
created_at : TIMESTAMP
- Organizer linked to user_id (UUID from MySQL)
- SERIAL for local indexing
- status: draft | approved
1:N
events
id_event : SERIAL «PK»
organizer_id : INT «FK»
title : VARCHAR(150)
description : TEXT
location : VARCHAR(150)
start_datetime : TIMESTAMP
price : NUMERIC(12,2)
capacity : INT
status : VARCHAR(20)
created_at : TIMESTAMP
- Event catalog: SERIAL ID, fast joins
- start_datetime stored in UTC
- price NUMERIC(12,2): allows global-scale pricing
- INDEX(status, start_datetime)
1:N
orders
id_order : SERIAL «PK»
buyer_id : CHAR(36) «UUID, external»
event_id : INT «FK»
quantity : INT
total_price : NUMERIC(12,2)
status : VARCHAR(20)
created_at : TIMESTAMP
- buyer_id = UUID from Authentication DB
- total_price NUMERIC(12,2) for large multi-ticket orders
- INDEX(event_id, status)
1:1
1:N
payments
id_payment : CHAR(36) «PK, UUID»
order_id : INT «FK»
provider_txn_id : VARCHAR(100)
status : VARCHAR(20)
amount : NUMERIC(12,2)
created_at : TIMESTAMP
updated_at : TIMESTAMP
- UUID PK for external payment references
- amount NUMERIC(12,2) supports international currencies
- provider_txn_id UNIQUE to ensure idempotency
tickets
id_ticket : CHAR(36) «PK, UUID»
order_id : INT «FK»
ticket_code : CHAR(36) «UUID, UNIQUE»
pdf_url : VARCHAR(255)
qr_code : TEXT
issued_at : TIMESTAMP
- UUID PK + UUID ticket_code
- ensures unguessable QR / public URLs
- one order -> many tickets

Figure 4: ProTicket Business Logic Database Diagram.

For our Business Logic Backend, we structured on 7 folders with one responsability for anyone.

- **Core**: Is where config of our database is saved, there we obtain our database url for the .env file and where we get connected to it.
- **Crud**:Is where every CRUD operation is realize, on this folder we separate CRUD for every table of our business logic database.
- **Models**: Is where the structure of every table of our database using SQLAlchemy.
- **Routes**: Is where we defined every single endpoint for any of our tables of the database.
- **Schemas**: Is where we defined the Pydantic schemas to control input and output data of our endpoints.
- **Services**: Is where the connection of auth database is defined.
- **Tests**: Is where tests for every endpoint are defined through pytest.

Database Connection Scripts/Configuration

Business Logic Backend Connection Scripts/Configuration

First one, we configure variable ‘BUSINESS_DATABASE_URL‘ in .env file:

```
BUSINESS_DATABASE_URL=postgresql://postgres:postgres@localhost:5432/pr
↪  oticket_business_logic
```

Create the database

```
CREATE DATABASE proticket_business_logic;
```

Initialize the database

```
python
>>> from core.database import Base, engine
>>> Base.metadata.create_all(bind=engine)
>>> exit()
```

REST API Documentation

Python Backend

1 Events Endpoints

The `/events` endpoints manage event creation, retrieval, and deletion.

1.1 POST /events/

Description: Creates a new event. Only users with the role `organizer` are authorized to perform this action.
Request:

```
POST /events/
Content-Type: application/json

{
  "title": "Rock Festival 2025",
  "description": "Annual rock festival with live bands.",
  "location": "Bogotá",
  "start_datetime": "2025-12-15T20:00:00",
  "price": 150.0,
  "capacity": 1000,
  "organizer_id": 1
}
```

Response 200 OK:

```
{
  "id_event": 1,
  "organizer_id": 1,
  "title": "Rock Festival 2025",
  "description": "Annual rock festival with live bands.",
  "location": "Bogotá",
  "start_datetime": "2025-12-15T20:00:00",
  "price": 150.0,
  "capacity": 1000
}
```

Response 403 Forbidden:

```
{
  "detail": "User not authorized to create events"
}
```

—

1.2 GET /events/

Description: Retrieves all available events.
Request:

```
GET /events/
```

Response 200 OK:

```
[
  {
    "id_event": 1,
    "organizer_id": 1,
    "title": "Rock Festival 2025",
    "description": "Annual rock festival with live bands.",
    "location": "Bogotá",
    "start_datetime": "2025-12-15T20:00:00",
    "price": 150.0,
    "capacity": 1000
  },
  {
    "id_event": 2,
    "organizer_id": 1,
    "title": "Tech Summit 2025",
    "description": "Technology and innovation conference.",
    "location": "Medellín",
    "start_datetime": "2025-09-10T09:00:00",
    "price": 120.0,
    "capacity": 500
  }
]
```

—

1.3 GET /events/{event_id}

Description: Retrieves details for a specific event by its ID.
Request:

```
GET /events/1
```

Response 200 OK:

```
{
  "id_event": 1,
  "organizer_id": 1,
  "title": "Rock Festival 2025",
  "description": "Annual rock festival with live bands.",
  "location": "Bogotá",
  "start_datetime": "2025-12-15T20:00:00",
  "price": 150.0,
  "capacity": 1000
}
```

Response 404 Not Found:

```
{
  "detail": "Event not found"
}
```

—

1.4 DELETE /events/{event_id}

Deletes an existing event by ID. **Description:** If the event does not exist, a 404 error is returned.
Request:

```
DELETE /events/1
```

Response 200 OK:

```
{
  "message": "Event deleted successfully",
  "deleted_event_id": 1
}
```

Response 404 Not Found:

```
{
  "detail": "Event not found"
}
```

2 Orders Endpoints

2.1 POST /orders/

Description: Creates a new order for a given event.

Request:

```
POST /orders/
Content-Type: application/json

{
  "event_id": 1,
  "quantity": 2
}
```

Response (201 Created):

```
{
  "id_order": 5,
  "buyer_id": "123e4567-e89b-12d3-a456-426614174000",
  "event_id": 1,
  "quantity": 2,
  "total_price": 200.00,
  "status": "pending",
  "created_at": "2025-11-09T15:45:00"
}
```

Response (400 Bad Request):

```
{
  "detail": "Insufficient tickets available"
}
```

—

2.2 GET /orders/{order_id}

Description: Retrieves a specific order by its ID.
Request:

```
GET /orders/5
```

Response (200 OK):

```
{
  "id_order": 5,
  "buyer_id": "123e4567-e89b-12d3-a456-426614174000",
  "event_id": 1,
  "quantity": 2,
  "total_price": 200.00,
  "status": "pending",
  "created_at": "2025-11-09T15:45:00"
}
```

Response (404 Not Found):

```
{
  "detail": "Orden not found"
}
```

—

2.3 GET /orders/user/{buyer_id}

Description: Returns all orders for a specific buyer (user).
Request:

```
GET /orders/user/123e4567-e89b-12d3-a456-426614174000
```

Response (200 OK):

```
[
  {
    "id_order": 5,
    "buyer_id": "123e4567-e89b-12d3-a456-426614174000",
    "event_id": 1,
    "quantity": 2,
    "total_price": 200.00,
    "status": "pending",
    "created_at": "2025-11-09T15:45:00"
  },
  {
    "id_order": 6,
    "buyer_id": "123e4567-e89b-12d3-a456-426614174000",
    "event_id": 3,
    "quantity": 1,
    "total_price": 150.00,
    "status": "confirmed",
    "created_at": "2025-11-09T16:00:00"
  }
]
```

Response (404 Not Found):

```
{
  "detail": "User has no orders"
}
```

—

2.4 PUT /orders/{order_id}/status

Description: Updates the status of an existing order.
Request:

```
PUT /orders/5/status
Content-Type: application/json

{
  "status": "confirmed"
}
```

Response (200 OK):

```
{
  "id_order": 5,
  "buyer_id": "123e4567-e89b-12d3-a456-426614174000",
  "event_id": 1,
  "quantity": 2,
  "total_price": 200.00,
  "status": "confirmed",
  "created_at": "2025-11-09T15:45:00"
}
```

Response (400 Bad Request):

```
{
  "detail": "Invalid status value"
}
```

3 Organizers Endpoints

3.1 POST /organizers/

Description: Creates a new organizer associated with the authenticated user.
Request:

```
POST /organizers/
Content-Type: application/json

{
  "organization_name": "Music Events Co.",
  "status": "active"
}
```

Response (200 OK):

```
{
  "id_organizer": 1,
  "user_id": "123e4567-e89b-12d3-a456-426614174000",
  "organization_name": "Music Events Co.",
  "status": "active",
  "created_at": "2025-11-09T15:45:00"
}
```

—

3.2 GET /organizers/{organizer_id}

Description: Retrieves details for a specific organizer by ID.
Request:

```
GET /organizers/1
```

Response (200 OK):

```
{
  "id_organizer": 1,
  "user_id": "123e4567-e89b-12d3-a456-426614174000",
  "organization_name": "Music Events Co.",
  "status": "active",
  "created_at": "2025-11-09T15:45:00"
}
```

Response (404 Not Found):

```
{
  "detail": "Organizer not found"
}
```

—

3.3 GET /organizers/

Description: Retrieves a paginated list of all organizers.

Request:

```
GET /organizers/?skip=0&limit=2
```

Response (200 OK):

```
[
  {
    "id_organizer": 1,
    "user_id": "123e4567-e89b-12d3-a456-426614174000",
    "organization_name": "Music Events Co.",
    "status": "active",
    "created_at": "2025-11-09T15:45:00"
  },
  {
    "id_organizer": 2,
    "user_id": "987e6543-e21b-43d3-c654-123614178999",
    "organization_name": "Festival Group",
    "status": "draft",
    "created_at": "2025-11-09T16:10:00"
  }
]
```

—

3.4 PUT /organizers/{organizer_id}

Description: Updates the details of an existing organizer.

Request:

```
PUT /organizers/1
Content-Type: application/json

{
  "organization_name": "Music Events International",
  "status": "active"
}
```

Response (200 OK):

```
{
  "id_organizer": 1,
  "user_id": "123e4567-e89b-12d3-a456-426614174000",
  "organization_name": "Music Events International",
  "status": "active",
  "created_at": "2025-11-09T15:45:00"
}
```

—

3.5 DELETE /organizers/{organizer_id}

Description: Deletes a specific organizer by ID.
Request:

```
DELETE /organizers/1
```

Response (200 OK):

```
{
  "message": "Organizer deleted successfully",
  "deleted_organizer_id": 1
}
```

4 Payments Endpoints

4.1 POST /payments/

Description: Creates a new payment record associated with an existing order.
Request:

```
POST /payments/
Content-Type: application/json

{
  "order_id": 1,
  "provider_txn_id": "TXN-987654321",
  "amount": 150.00
}
```

Response (201 Created):

```
{
  "id_payment": "550e8400-e29b-41d4-a716-446655440000",
  "order_id": 1,
  "provider_txn_id": "TXN-987654321",
  "status": "initiated",
  "amount": 150.00,
  "created_at": "2025-11-09T16:30:00",
  "updated_at": "2025-11-09T16:30:00"
}
```

Response (400 Bad Request):

```
{
  "detail": "Transaction with this provider_txn_id already exists"
}
```

—

4.2 GET /payments/{payment_id}

Description: Retrieves payment details by its unique UUID.

Request:

```
GET /payments/550e8400-e29b-41d4-a716-446655440000
```

Response (200 OK):

```
{
  "id_payment": "550e8400-e29b-41d4-a716-446655440000",
  "order_id": 1,
  "provider_txn_id": "TXN-987654321",
  "status": "initiated",
  "amount": 150.00,
  "created_at": "2025-11-09T16:30:00",
  "updated_at": "2025-11-09T16:30:00"
}
```

Response (404 Not Found):

```
{
  "detail": "Payment not found"
}
```

—

4.3 PUT /payments/{payment_id}/status

Description: Updates the status of a payment (e.g., from `initiated` to `completed` or `failed`).

Request:

```
PUT /payments/550e8400-e29b-41d4-a716-446655440000/status
Content-Type: application/json

{
  "status": "completed"
}
```

Response (200 OK):

```
{
  "id_payment": "550e8400-e29b-41d4-a716-446655440000",
  "order_id": 1,
  "provider_txn_id": "TXN-987654321",
  "status": "completed",
  "amount": 150.00,
  "created_at": "2025-11-09T16:30:00",
  "updated_at": "2025-11-09T17:00:00"
}
```

Response (400 Bad Request):

```
{
  "detail": "Invalid status transition"
}
```

5 Tickets Endpoints

5.1 POST /tickets/

Description: Creates a new ticket associated with an existing order.
Request:

```
POST /tickets/
Content-Type: application/json

{
  "order_id": 1,
  "pdf_url": "https://example.com/ticket_1.pdf",
  "qr_code": "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAA..."
}
```

Response (201 Created):

```
{
  "id_ticket": "b56d3e6f-3c5f-4bcb-9a1a-835b3adbe001",
  "order_id": 1,
  "ticket_code": "0d7c6c42-8acb-4a33-9f37-4b2e5d45a6dd",
  "pdf_url": "https://example.com/ticket_1.pdf",
  "qr_code": "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAA...",
  "issued_at": "2025-11-09T16:30:00"
}
```

Response (400 Bad Request):

```
{
  "detail": "Error creating ticket: order_id not found"
}
```

—

5.2 GET /tickets/{ticket_id}

Description: Retrieves a specific ticket by its unique identifier (`id_ticket`).
Request:

```
GET /tickets/b56d3e6f-3c5f-4bcb-9a1a-835b3adbe001
```

Response (200 OK):

```
{
  "id_ticket": "b56d3e6f-3c5f-4bcb-9a1a-835b3adbe001",
  "order_id": 1,
  "ticket_code": "0d7c6c42-8acb-4a33-9f37-4b2e5d45a6dd",
  "pdf_url": "https://example.com/ticket_1.pdf",
  "qr_code": "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAA...",
  "issued_at": "2025-11-09T16:30:00"
}
```

Response (404 Not Found):

```
{
  "detail": "Ticket not found"
}
```

—

5.3 GET /tickets/order/{order_id}

Description: Retrieves all tickets associated with a specific order.

Request:

```
GET /tickets/order/1
```

Response (200 OK):

```
[
  {
    "id_ticket": "b56d3e6f-3c5f-4bcb-9a1a-835b3adbe001",
    "order_id": 1,
    "ticket_code": "0d7c6c42-8acb-4a33-9f37-4b2e5d45a6dd",
    "pdf_url": "https://example.com/ticket_1.pdf",
    "qr_code": "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAA...",
    "issued_at": "2025-11-09T16:30:00"
  },
  {
    "id_ticket": "d97f93f4-4e1f-41a0-b65d-5b3f0c89a22f",
    "order_id": 1,
    "ticket_code": "b5f0cbbd-cc3b-4e1d-a91a-223d7a06a2dc",
    "pdf_url": "https://example.com/ticket_2.pdf",
    "qr_code": "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAA...",
    "issued_at": "2025-11-09T16:32:00"
  }
]
```

Response (404 Not Found):

```
{
  "detail": "No tickets found for this order"
}
```

—

5.4 GET /tickets/code/{ticket_code}

Description: Retrieves a ticket by its unique ticket code (QR-based).
Request:

```
GET /tickets/code/0d7c6c42-8acb-4a33-9f37-4b2e5d45a6dd
```

Response (200 OK):

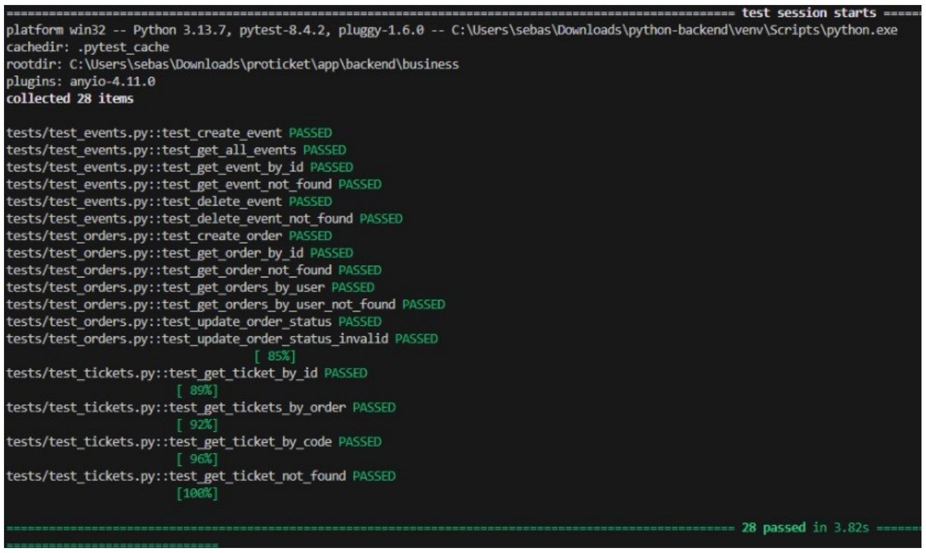
```
{
  "id_ticket": "b56d3e6f-3c5f-4bcb-9a1a-835b3adbe001",
  "order_id": 1,
  "ticket_code": "0d7c6c42-8acb-4a33-9f37-4b2e5d45a6dd",
  "pdf_url": "https://example.com/ticket_1.pdf",
  "qr_code": "data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAA...",
  "issued_at": "2025-11-09T16:30:00"
}
```

Response (404 Not Found):

```
{
  "detail": "Ticket not found"
}
```

Unit Test Results and Code

For python-backend we used pytest to validate correct functionality of every endpoint, this took us to develop 28 test in total. There is the proof of our FastAPI pass every test.



```
==================================================================================================== test session starts ======
platform win32 -- Python 3.13.7, pytest-8.4.2, pluggy-1.6.0 -- C:\Users\sebas\Downloads\python-backend\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\sebas\Downloads\proticket\app\backend\business
plugins: anyio-4.11.0
collected 28 items

tests/test_events.py::test_create_event PASSED
tests/test_events.py::test_get_all_events PASSED
tests/test_events.py::test_get_event_by_id PASSED
tests/test_events.py::test_get_event_not_found PASSED
tests/test_events.py::test_delete_event PASSED
tests/test_events.py::test_delete_event_not_found PASSED
tests/test_orders.py::test_create_order PASSED
tests/test_orders.py::test_get_order_by_id PASSED
tests/test_orders.py::test_get_order_not_found PASSED
tests/test_orders.py::test_get_orders_by_user PASSED
tests/test_orders.py::test_get_orders_by_user_not_found PASSED
tests/test_orders.py::test_update_order_status PASSED
tests/test_orders.py::test_update_order_status_invalid PASSED
                                    [ 85%]
tests/test_tickets.py::test_get_ticket_by_id PASSED
                         [ 89%]
tests/test_tickets.py::test_get_tickets_by_order PASSED
                         [ 92%]
tests/test_tickets.py::test_get_ticket_by_code PASSED
                         [ 96%]
tests/test_tickets.py::test_get_ticket_not_found PASSED
                         [100%]

==================================================================================================== 28 passed in 3.82s ======
==============================
```

Figure 5: Unit testing for FastAPI endpoints from python-backend.

Evidence of Web GUI Integration

Demostrative video of Web Application functionality: https://youtu.be/rrIqo7G_2m4