UML

February 11, 2025

0.1 1 - Cadastro e Autenticação de Usuários

0.1.1 1.1 - User

Atributos
- `id`: Identificador único do usuário.
- `name`: Nome do usuário.
- `email`: Email usado para login.
- `password`: Senha protegida (criptografada).
- `role`: Define se o usuário é um **PARTICIPANTE** ou **ORGANIZADOR**.

Métodos
- `register()`: Cria um novo usuário no sistema.
- `login()`: Realiza a autenticação do usuário.

0.1.2 1.2 - Role (Enum)

Atributos
- `PARTICIPANTE`: Usuários que compram ingressos e participam dos eventos.
- `ORGANIZADOR`: Usuários que criam e gerenciam eventos.

0.1.3 1.3 - AuthService

Métodos
- `register(User user)`: Cria um usuário e salva no banco de dados.
- `authenticate(String email, String password)`: Valida login com email e senha.

0.1.4 1.4 - AuthController

Métodos
- `register(User user)`: Chama o `AuthService` para criar um usuário.
- `login(AuthRequest request)`: Processa um login de usuário.

0.1.5 1.5 - UserRepository

Métodos
- `findByEmail(String email)`: Busca um usuário pelo email.

0.2 2 - Criação e Gerenciamento de Eventos

0.2.1 2.1 - Event

Atributos
- `id`: Identificador único do evento.
- `title`: Nome do evento.
- `description`: Detalhes sobre o evento.
- `date`: Data e hora em que o evento acontecerá.
- `location`: Local onde o evento ocorrerá.

Métodos
- `createEvent()`: Cria um novo evento.
- `updateEvent()`: Edita os detalhes do evento.
- `deleteEvent()`: Remove o evento do sistema.

0.2.2 2.2 - EventService

Métodos
- `createEvent(EventDTO event)`: Cria um evento validando os dados.
- `updateEvent()`: Permite que um organizador edite seu evento.
- `listEvents()`: Retorna todos os eventos cadastrados.

0.2.3 2.3 - EventController

Métodos
- `createEvent(Event event)`: Permite que organizadores criem eventos.
- `getAllEvents()`: Retorna todos os eventos cadastrados.
- `getEventById(Long id)`: Busca um evento pelo ID.

0.2.4 2.4 - EventRepository

Métodos
- `JpaRepository<Event, Long>`: Métodos básicos como `save()` e `findById()`.

0.3 4 - Agenda e Programação do Evento

0.3.1 4.1 - Activity

Atributos
- `id`: Identificador único da atividade.
- `name`: Nome da atividade (ex: "Palestra sobre IA").
- `description`: Detalhes sobre a atividade.
- `schedule`: Data e hora da atividade dentro do evento.

Métodos
- `addActivity()`: Adiciona uma nova atividade ao evento.
- `updateActivity()`: Edita os detalhes da atividade.
- `removeActivity()`: Remove uma atividade do evento.

0.3.2 4.2 - ActivityService

Métodos
- `addActivity(Activity activity)`: Adiciona uma nova atividade ao evento.
- `getActivitiesByEvent(Long eventId)`: Retorna todas as atividades de um evento.

0.3.3 4.3 - ActivityController

Métodos
- `addActivity(Activity activity)`: Permite adicionar atividades ao evento.
- `getActivitiesByEvent(Long eventId)`: Retorna a programação de um evento.

0.3.4 4.4 - ActivityRepository

Métodos
- `findByEventId(Long eventId)`: Busca todas as atividades de um evento.

0.3.5 OBS: A numeração é de acordo com o que entendi do documento do prof e de acordo com os requisitos

1 Arquitetura do Sistema de Gestão de Eventos

1.0.1 Backend

- Linguagem: Java 17 ou 21
- Framework: Spring Boot
- APIs: REST (Spring Web) e assíncronas (Spring Async + CompletableFuture)
- Banco de Dados: MySQL com JDBC (Spring Data JDBC)
- Autenticação: Spring Security + JWT com OAuth2
- Mensageria: RabbitMQ (Spring AMQP)
- Testes: JUnit 5 + Mockito

- Pagamento: Stripe API (Webhooks para confirmação)

1.0.2 Frontend

- Linguagem: JavaScript
- Framework: React (JS, HTML, CSS)
- Chamada de APIs: Axios (requisições assíncronas)
- Gerenciamento de estado: Redux Toolkit ou Context API
- UI: Material UI ou Tailwind CSS

1.0.3 Infraestrutura

- CI/CD: GitHub Actions
- Documentação: Swagger
- Monitoramento: Spring Actuator (se houver necessidade)

1.0.4 Código para gerar o Diagrama UML

[1]:
```
from PIL import Image, ImageDraw, ImageFont


width, height = 1600, 1200
img = Image.new("RGB", (width, height), "white")
draw = ImageDraw.Draw(img)

box_width, box_height = 280, 220
x_gap, y_gap = 60, 70

elements = [
    ("User", ["- id: Long", "- name: String", "- email: String", "- password:␣
 ↪String", "- role: Role"],
     ["+ register()", "+ login()"]),
    ("Event", ["- id: Long", "- title: String", "- description: String", "-␣
 ↪date: LocalDateTime", "- location: String"],
     ["+ createEvent()", "+ updateEvent()", "+ deleteEvent()"]),
    ("Activity", ["- id: Long", "- name: String", "- description: String", "-␣
 ↪schedule: LocalDateTime"],
     ["+ addActivity()", "+ updateActivity()", "+ removeActivity()"]),
    ("Role (Enum)", ["- PARTICIPANTE", "- ORGANIZADOR"], []),
    ("AuthService", [], ["+ register(User)", "+ authenticate(String, String)"]),
    ("AuthController", [], ["+ register(User)", "+ login(AuthRequest)"]),
    ("UserRepository", [], ["+ findByEmail(String)"]),
    ("EventService", [], ["+ createEvent(EventDTO)", "+ updateEvent()", "+␣
 ↪listEvents()"]),
    ("EventController", [], ["+ createEvent(Event)", "+ getAllEvents()", "+␣
 ↪getEventById(Long)"]),
    ("EventRepository", [], ["+ JpaRepository<Event, Long>"]),
```

```
    ("ActivityService", [], ["+ addActivity(Activity)", "+␣
 ↪getActivitiesByEvent(Long)"]),
    ("ActivityController", [], ["+ addActivity(Activity)", "+␣
 ↪getActivitiesByEvent(Long)"]),
    ("ActivityRepository", [], ["+ findByEventId(Long)"]),
]

try:
    font = ImageFont.truetype("arial.ttf", 18)
except IOError:
    font = ImageFont.load_default()

x_start, y_start = 30, 30

for i, (class_name, attributes, methods) in enumerate(elements):
    x = x_start + (i % 4) * (box_width + x_gap)
    y = y_start + (i // 4) * (box_height + y_gap)

    draw.rectangle([x, y, x + box_width, y + box_height], outline="black",␣
 ↪width=2)
    draw.rectangle([x, y, x + box_width, y + 40], fill="lightgray",␣
 ↪outline="black")

    draw.text((x + 5, y + 10), class_name, font=font, fill="black")

    attr_y = y + 50
    for attr in attributes:
        draw.text((x + 5, attr_y), attr, font=font, fill="black")
        attr_y += 20

    method_y = attr_y + 10
    for method in methods:
        draw.text((x + 5, method_y), method, font=font, fill="black")
        method_y += 20

uml_diagram_resized_path = "uml.png"
img.save(uml_diagram_resized_path)
uml_diagram_resized_path
```

[1]: 'uml.png'