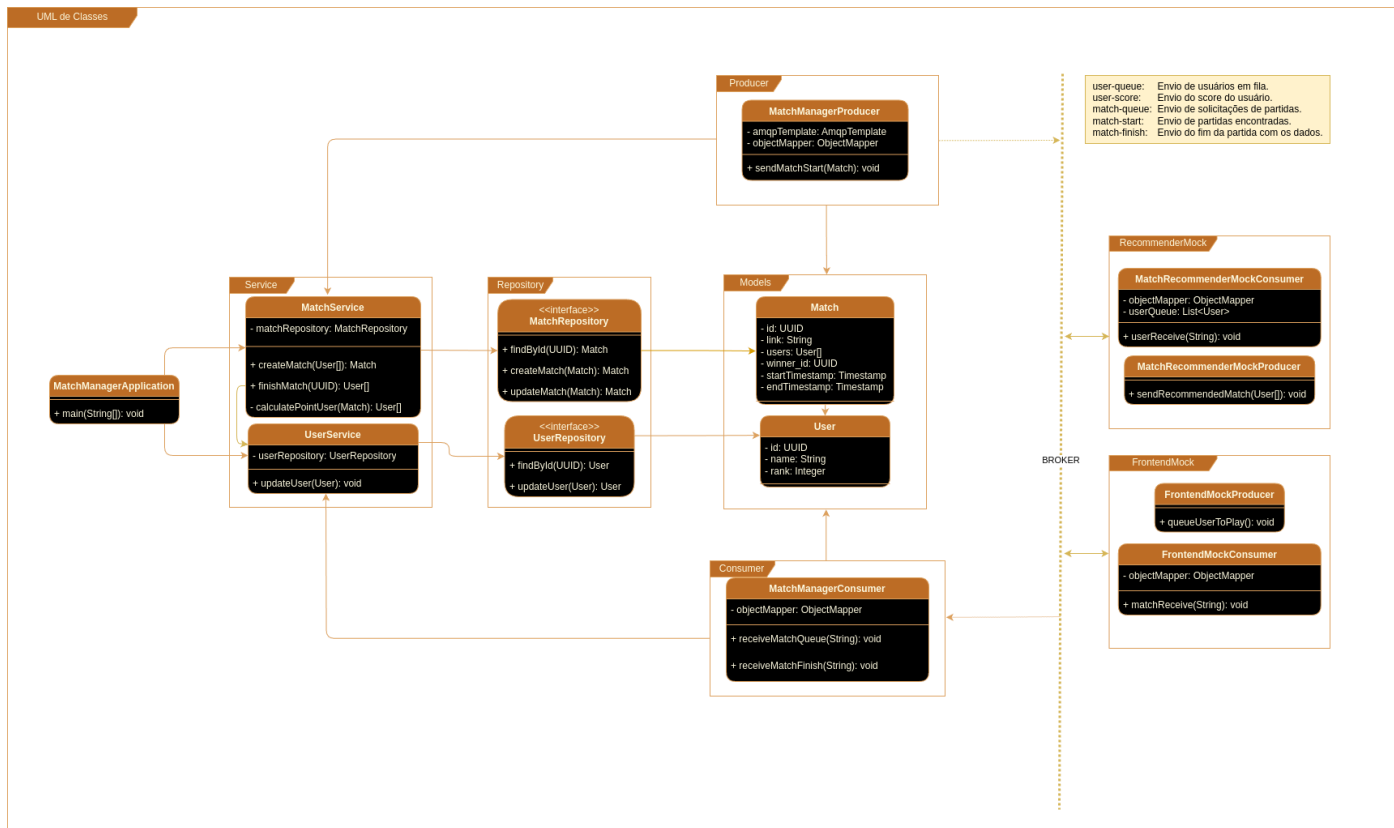
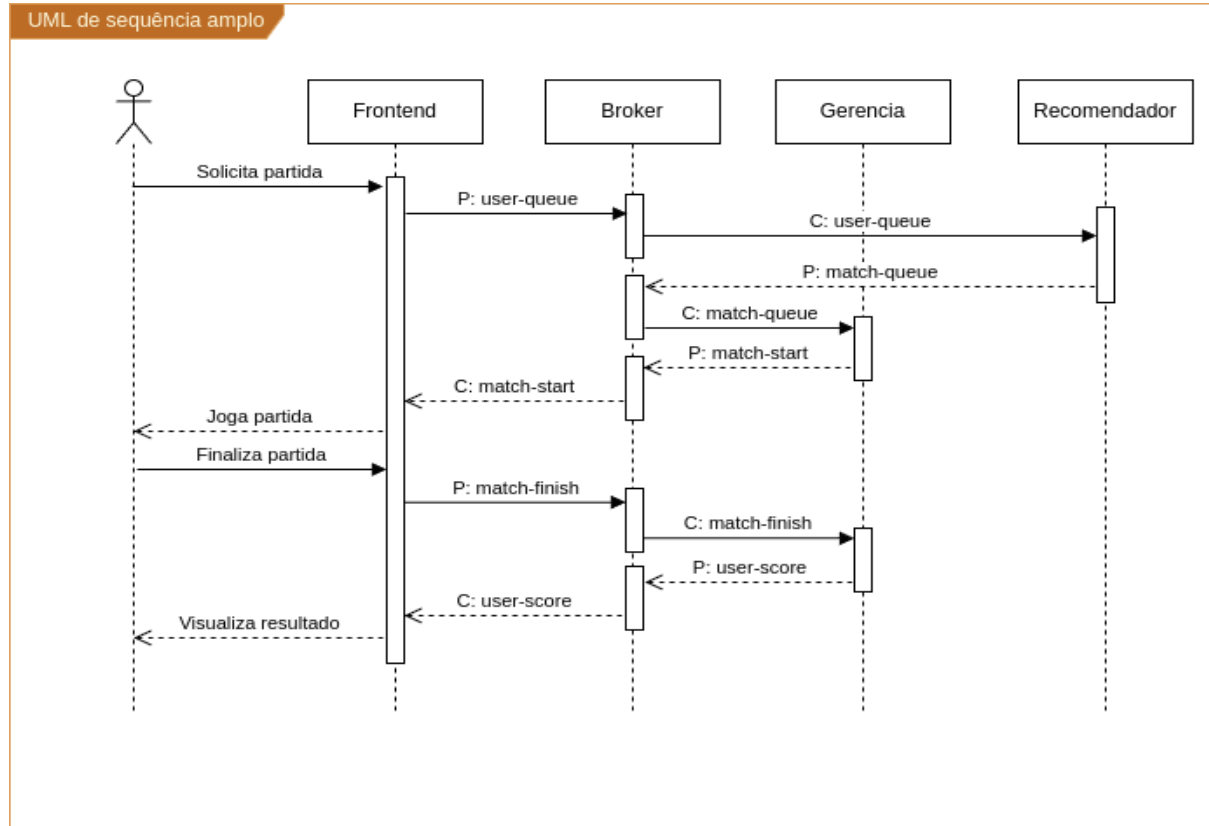


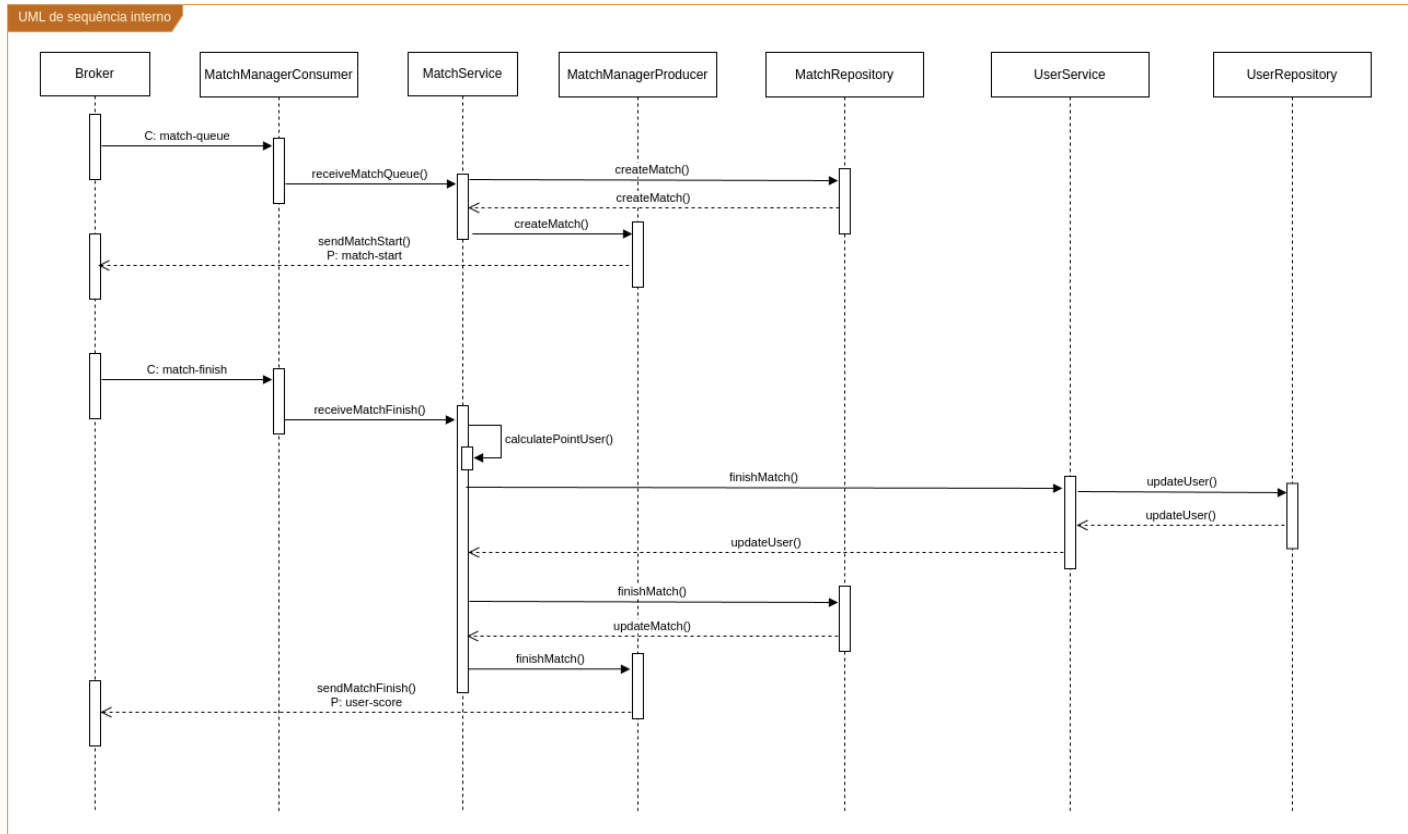
# UML de Classes



# UML de Sequência amplo



# UML de Sequência interno



## Design Pattern: Singleton/Decorator

```
@Service
public class MatchService {
    @Autowired
    private MatchRepository matchRepository;
}
```

# Design Pattern: Builder/Decorator

```
@Table(name = "matches")
@Entity(name = "match")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class Match {
    @Id
    @GeneratedValue(strategy = GenerationType.UUID)
    private UUID id;

    @ManyToOne
    @JoinColumn(name = "user1_id")
    private User user1;

    @ManyToOne
    @JoinColumn(name = "user2_id")
    private User user2;
```

## Design Pattern: Observer/Decorator

```
@Component
public class MatchManagerConsumer {
    private final ObjectMapper objectMapper = new ObjectMapper();

    @RabbitListener(queues = "user-queue")
    public void userReceive(String message) {
```

# Documentação

## How to run

1. Clone the repository

```
git clone https://github.com/InatelS203/GerenciaDePartidas.git
```

2. Then open a command line tool in the following path: "matchManager/matchManager"

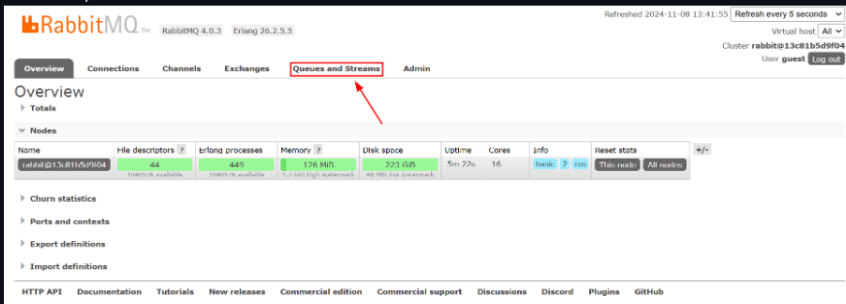
Run Postgres and RabbitMQ first

```
docker-compose up -d postgres rabbitmq
```

3. Now, into your browser enter in the URL: "<http://localhost:15672/>"

Login with username="guest" and password="guest"

4. Go to the queue tab



The screenshot shows the RabbitMQ 4.0.3 management interface. The 'Queues and Streams' tab is highlighted with a red box and a red arrow. The interface displays an overview of the cluster, including node statistics and various tabs for management.

Name	File descriptors	Erlang processes	Memory	Link space	Uptime	Cores	Info	Reset stats
rabbit@13c81b5d9f04	44	440	176 MB	371 GiB	5m 22s	16	<a href="#">View</a> <a href="#">Info</a>	<a href="#">Reset stats</a>

5. Create the "user-queue" following the image