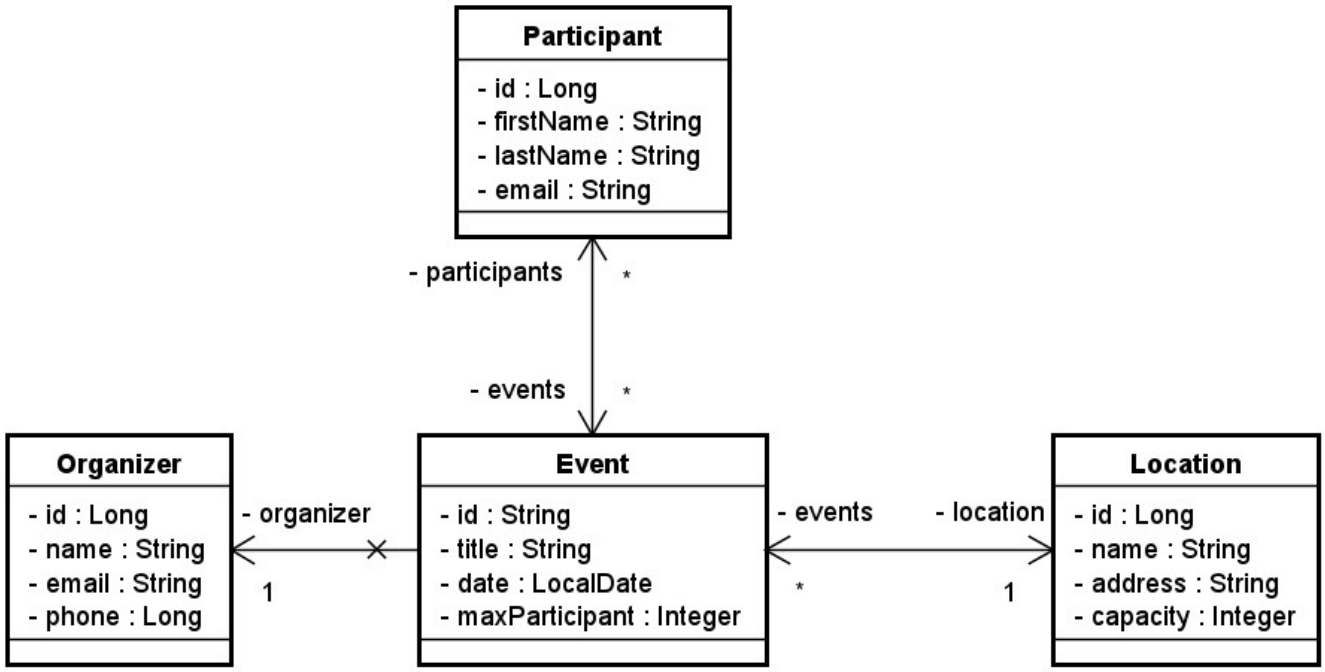
Event Management System

Create a Spring Boot REST-API with a database for an event management system. This system manages `events`, `locations`, `organizers`, and `participants`.

The following class diagram is provided:



1. Create Entities using JPA

Create a new database named `eventDB` and make sure to create all entities as shown in the class diagram. Create the mappings to the tables in the database and use `validations` for some of the attributes.

2. Create Repositories

Create repositories for all the entities and implement the following queries in the correct repositories.

Derived Queries

- Find all events after a defined *date*.
- Find all locations where there are events taking place with more than a maximum of 1000 participants.
- Find all participants which are registered for at least *x* events.

JPQL Queries

- Find all events with less than *x* free tickets.
- Find all organizers which have organzied more than 5 events at a defined *location*.
- Find all participants which participate at any event within the next 7 days.

Native Queries

- Find the 3 most loved events (most participants) ordered by the number of participants.

- Find all locations, which have more than 2 events on the same day.
- Find all participants, which are registered for events in different locations.

3. Create Services

Define a clear business logic by creating the services for each entity and make sure that all `CRUD` operations are present.

4. DTOs + Mapper

No entity should be sent to the frontend. Create DTOs and the mappers needed. Make sure to not show all data: e.g.: `EventDTO` only contains `id`, `title`, and `date`; `EventDetailDTO` shows everything, ...

5. Create Controllers

Create all the controllers with useful endpoints. Make sure to also offer pagination and sorting and use all types of passing data to the backend.

6. Validation

Make sure to validate data, before something is added to the table or processed. Use `@Valid` in the controller to do so.

7. Exception Handling

Create a `GlobalExceptionHandler` and create your own Exceptions for the different use cases. Also catch all the other Errors with the message: *Ups, an error occurd!*.

8. Data

Make sure that you test all your functionality with differnt data. Provide an import script for some test data and make sure that all queries deliver valid results. All queries must be testable with `Postman`.