EVENT MANAGEMENT SYSTEM (BACKEND)

Sandali Vithanage

[sandalisamadara95@gmail.com](mailto:sandalisamadara95@gmail.com)

Contents

[Problem Description 3](#_Toc116260371)

[Technologies Used 3](#_Toc116260372)

[Database Design 4](#_Toc116260373)

[Assumptions 4](#_Toc116260374)

[API Specification 5](#_Toc116260375)

[Future Modifications 6](#_Toc116260376)

# Problem Description

Backend Task

Create a locally runnable backend application for Event management (such as concerts,

meetups etc.) in your preferred JVM language. Feel free to use any major framework (Spring

Boot is most welcome). The application should have a persistence layer and a REST API.

User management can be skipped for simplicity.

**Domain Requirements:**

-

The user should be able to create, read and delete an event. Also to list all available

events.

-

An event should have a name, date, city, country, a guest list and a weather forecast.

-

Weather forecast should be filled while creating an event by calling an external API

(https://openweathermap.org/api)

**Technical Requirements:**

-

The application should be runnable either by calling a Gradle task, Maven or Docker

container. There should be a README outlining how to do it.

-

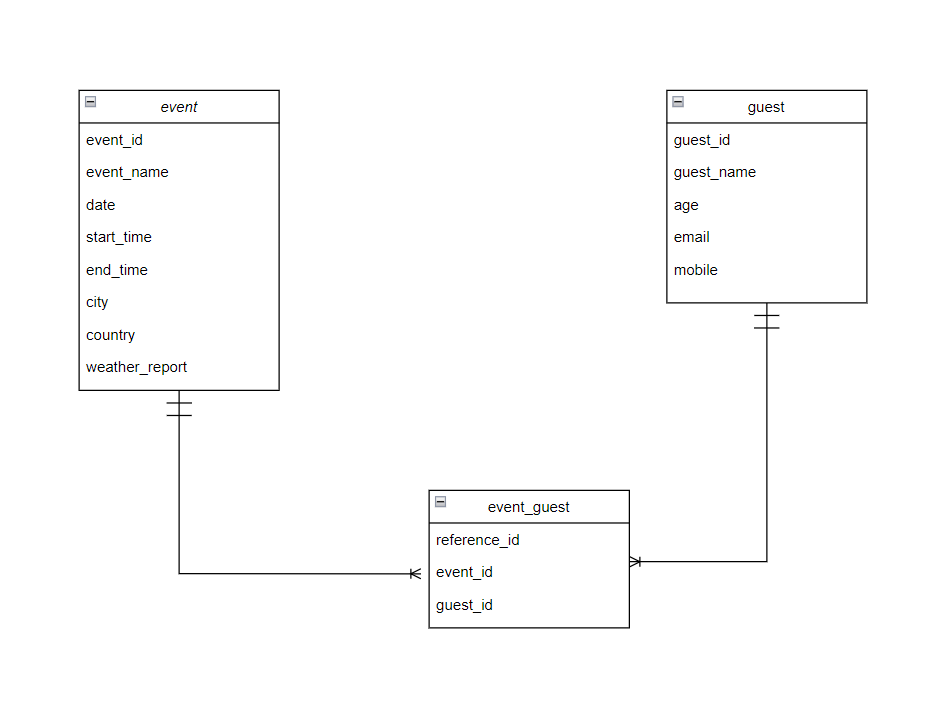
The endpoints should return data in JSON format.

# Technologies Used

1. Java 8
2. Spring Boot framework.
3. Hibernate.
4. JPA.
5. MySQL Database.
6. Lombok.
7. IDE (IntelliJ Idea, MySQL Workbench)
8. Postman

# Database Design

I assumed that Event and Guest has many-to-many relationship and designed the database.



DB scripts are in src/main/java/resources folder.

# Assumptions

1. User Management is skipped.
2. Guests are assumed to be users of the system who are registered before. So, guests are assumed to be already there in the database. New guest can be inserted when a guest is registered to the system. We can add existing guests to events using API endpoints in EventsController.
3. To get the coordinates of the location <https://openweathermap.org/api/geocoding-api> is used. Getting the coordinates by city and country. (ISO 3166 country codes should be used)
4. To get weather a free API from OpenWeather is used. I used 5 day/3hour forecast data.

<https://openweathermap.org/forecast5>

1. My openweathermap key is hard coded in the code. Please replace it if needed.

*(Change APP\_ID value with your key in EventServiceImpl.java.)*

1. Since 5 day/3hour forecast is used, event creation date range is restricted. User can add events for 5 days from current date only.
2. As weather forecast weather during the event (per 3 hours) is added to database column as a comma separated value.

*Eg: Returned weather from API is (3:00 – Clouds, 6:00 – Clouds, 9:00 – Rain)*

*If event time is 3:30 – 8:30 weather will be Clouds, Rain.*

*If event time is 3:30 – 4:30 weather will be Clouds.*

1. **How to start the application and run, how to call REST API are described in the README.md file inside event.management folder.**

# API Specification

Use Postman for following API endpoints.

1. To add a event use following url with POST request.

<http://localhost:8080/eventManagement/events>

set request body as raw with JSON format.

    {

        "event\_name": "M2",

        "date": "2022-10-10",

        "start\_time" : "03:30:00",

        "end\_time" : "18:30:00",

        "city": "Berlin",

        "country": "DE",

        "guestSet": [

            {

                "guest\_id": 4,

                "guest\_name": "Harry Potter",

                "age": 17,

                "email": "harry.potter@gmail.com",

                "mobile": "0000011014"

            },

            {

                "guest\_id": 5,

                "guest\_name": "Ron Weasley",

                "age": 17,

                "email": "ron.w@gmail.com",

                "mobile": "000001105"

            },

            {

                "guest\_id": 6,

                "guest\_name": "Hermoine Granger",

                "age": 17,

                "email": "hermoine.g@gmail.com",

                "mobile": "000001106"

            }

        ]

}

1. To list all events use following url with GET request.

<http://localhost:8080/eventManagement/events>

1. To get events by id, use following url with GET request.

[http://localhost:8080/eventManagement/events/<id>](http://localhost:8080/eventManagement/events/%3cid%3e)

1. To delete a event, use following url with DELETE request.

[http://localhost:8080/eventManagement/events/<id>](http://localhost:8080/eventManagement/events/30)

1. Replace <id> with an id.
2. **Postman collection is in event.management folder.**

# Future Modifications

1. Add more database constraints.
2. Improvements in exception handling and error/exception messages.
3. Unit Testing.
4. Performance Testing.
5. Predict more accurate weather by calling a weather API that predict hourly weather.
6. Add Guest management endpoints and User management endpoints.