Information Management & Systems Engineering

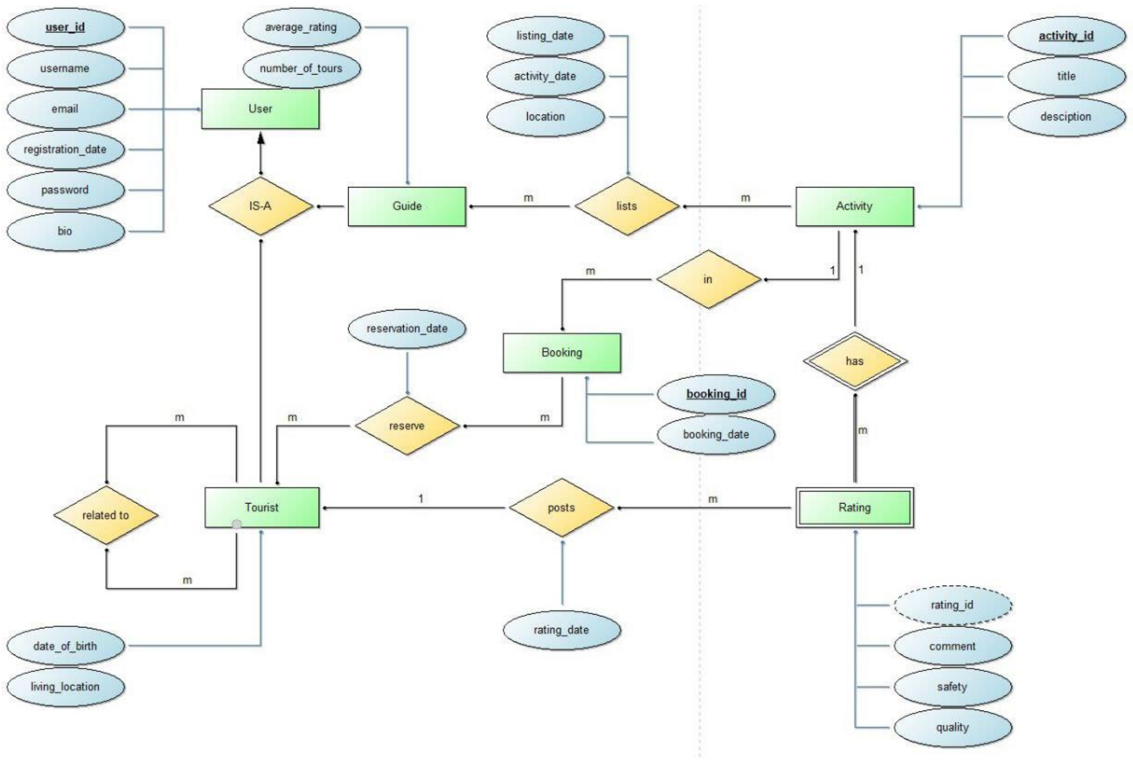
Milestone 2

Arsen Keshishyan 01576524

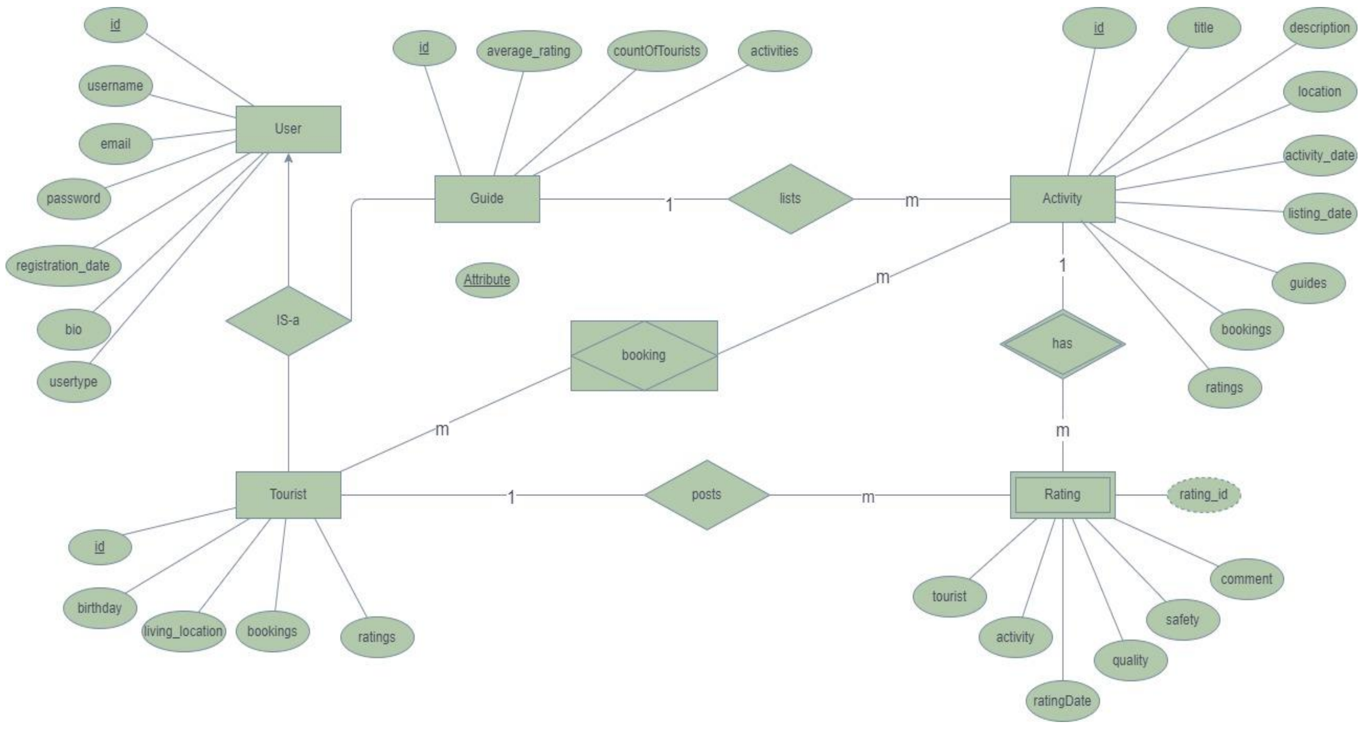
Logical/Physical database design of your RDBMS

ER model of Milestone 1:

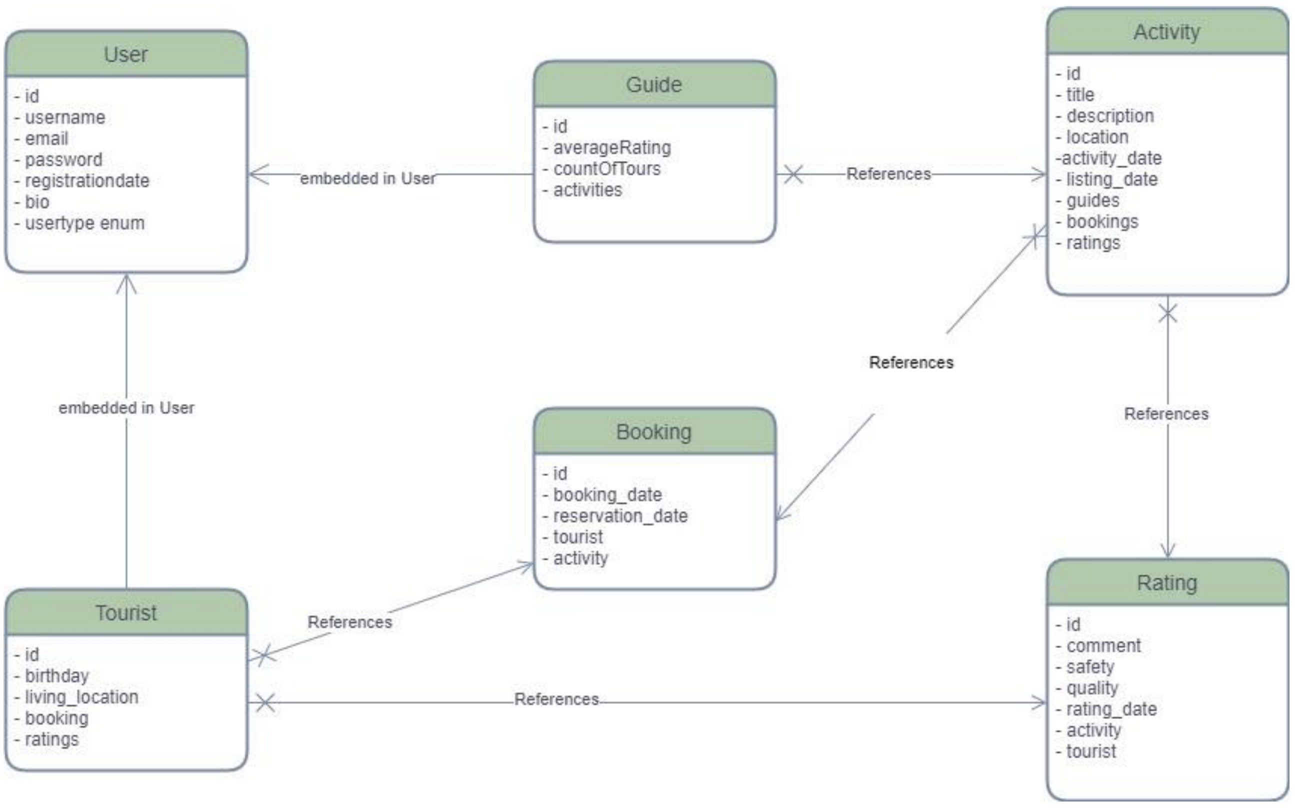
ER



model of Milestone 2:



NoSQL schema:



Json schema:

```
{
   "$schema":"https://json-schema.org/draft/2022-06/schema",
   "$id":"https://user.schema.json",
   "user":{
      "title":"Record of user",
      "description":"This document records the details of an user",
      "type":"object",
      "properties":{
         "id":{
            "description":"A unique identifier for an user",
            "type":"number"
         },
         "username":{
            "description":"Full name of the user",
            "type":"string"
         },
         "email":{
            "description":"Email of the user",
            "type":"string"
         },
         "registration_date":{
            "description":"Registration Date of the user",
            "type":"string"
         },
         "password":{
```

```
        "description":"Password of the user",
        "type":"number"
      },
      "bio":{
        "description":"Personal description of the user",
        "type":"string"
      },
      "user_type":{
        "description":"Type of the user",
        "type":"string"
      },
      "birthday":{
        "description":"Birthday of the user",
        "type":"string"
      },
      "living_location":{
        "description":"Living location of the user",
        "type":"string"
      },
      "average_rating":{
        "description":"Average rating of the user",
        "type":"number"
      },
       "counts_of_tours":{
        "description":"Number of tours an user guide",
        "type":"number"
      }
    }
  },


  "activity":{
    "title":"Record of tourism activities ",
    "description":"This document records the details of an activity",
    "type":"object",
    "properties":{
      "id":{
        "description":"A unique identifier for an activity",
        "type":"number"
      },
      "title":{
        "description":"The title for an activity",
        "type":"string"
      },
      "description":{
        "description":"The description for an activity",
        "type":"string"
      },
      "location":{
        "description":"The location for an activity",
        "type":"string"
      },
      "activity_date":{
        "description":"The title for an activity",
        "type":"string"
      },

      "listing_date":{
        "description":"The listing date for an activity",
        "type":"string"
      },
      "guide_id":
        {
          "description":"An unique identifier for the user(guide)",
          "type":"number"
        }
    }
  },

  "booking": {
    "title":"Record of tourist's booking",
    "description":"This document records the details of an booking",
    "type":"object",
    "properties":{
      "booking_id":{
        "description":"A unique identifier for a booking",
        "type":"number"
      },
      "booking_date":{
        "description":"certain date for a booking",
        "type":"number"
      },
      "reservation_date":{
        "description":"certain date for an activity reservation",
        "type":"number"
      },
      "tourist_id":{
          "user_id":"Tourist inherits from user table",
          "description":"The tourist who creates activity",
          "type":"number"
      },

      "activity_id":{
          "activity_id":"A unique identifier for an activity",
          "description":"The description for an activity",
          "type":"number"
      }

    }
  },

  "rate":{
     "title":"Record of tourist's ratings for activities",
     "description":"This document records the details of an rating",
     "type":"object",
     "properties":{
       "rating_id":{
         "description":"A unique identifier for a rating",
         "type":"number"
       },
       "rating_date":{
         "description":"certain date for a rating",
         "type":"string"
       },
       "comment":{
         "description":"certain comment of rating",
         "type":"string"
       },
       "safety":{
         "description":"certain safty of rating",
         "type":"string"
       },
       "quality":{
         "description":"quality for an activity",
         "type":"string"
       },
       "activity_id":{
         "description":"A unique identifier for an activity",
         "type":"number"
       },
       "tourist_id":{
         "description":"A unique identifier for an tourist",
         "type":"number"
       }
     }
   }
 }
```

Technology stack:

Backend**:**

Rest-Backend is built with Spring Boot. It makes Rest-Endpoints easy to create. Moreover, Spring Boot connects to our PostgreSQL database as well as MongoDB without boilerplate code.

SQL

For the relational database, PostgreSQL is used in combination with Spring Data JPA (Hibernate ORM) to create tables based on Java classes.

NoSQL

Spring Data MongoDB is used for non-relational databases in the java-backend, which allows for object mapping and mongo operations.

Frontend:

The frontend is made with the JavaScript React library, which allows us to build interactive web pages.

NoSQL design

The main 4 collections are: Users, Activities, Booking, Rating. Although there are 2 types of users as guide & tourist, they all can be contained inside user collections and the additional attributes of guide and the tourist can be contained inside the user collection as well. This collection has the all information about the users . Details of users are frequently accessed; therefore, it is beneficial to having one collection rather than making joins.
Activities are created by the “guide”. The other users like tourists do not have permission to create activities. Booking is a collection where the reservations of the activities made by the “tourist”. Bookings and Actives should be recorded separately because there are many to many relationships between the tourist and the activities. Activities will only exist in the Database untill a tourist make booking.

In the system the rating will be generated after bookings are made. Also the tourist should rate particular activity . It means the storing together details that are not inserted or retrieved often is not an efficient way . Ratings of the activates queried less frequently as it is not compulsory for every user to rate all the activities. Therefore, ratings are inserted and retrieved only if needed. There is no need to iterate throughout all the activities for inserting/retrieving ratings.

NoSQL indexing

_Id -> index : Most of the queries are using the _Id.
Username -> unique index: The unique property for an index causes MongoDB to reject duplicate values for the indexed field. Username of the user collection should be an unique index so that the mongoDB contains only unique user names. Also it prevents from inserting non-unique usernames to the DB.

Location -> Geographical index: If this system is further developing with map functionalities and the location using coordinates, to support efficient queries of geospatial coordinate data, MongoDB provides two special indexes: 2d indexes that uses planar geometry when returning results and 2dsphere indexes that use spherical geometry to return results.

Description -> Text index : when the tourist search for activities, description field of the activity collection can be applied . Therefore, the fast query search can be done for the keywords in the text fields.

Booking collection -> TTL index : we can delete data after a specific period of time regarding booking information. As an example: the bookings(reservations) within the last year is important for generating reports. The TTL index can be used for booking collection. TTL will delete the documents automatically if it is older then one year .

Show and compare the SELECT statements of both main and elaborate use cases from to MongoDBquery statements

Reporting use case: `SELECT users.id, users.name, activity.title, activity.location, FROM users, activities`