

Assignment 3 - Landmarks

GitLab Repository: https://gitlab.com/bicocca_projects/2023_assignment3_landmarks

Group components: {givenName} {familyName} - {badgeNumber}:

- Nicol Emanuele - 919020
- Nicolas Guarini - 918670

Project

This project focuses on developing the backend of a social application for saving points of interest (*landmarks*) around the world. Each user can save points of interest which include the coordinates (latitude, longitude, altitude), a name, and the description relating to that point. A user can also "follow" other users, in order to see (via a hypothetical frontend which can be a web app or a mobile app) the points of interest that they save at any time.

At the implementation level, [JPA](#) and [Hibernate](#) was used to manage synchronization and mapping with the PostgreSQL database, while the [Spark](#) framework was used to manage the API endpoints, which allows fast and functional management of the endpoints and related handlers.

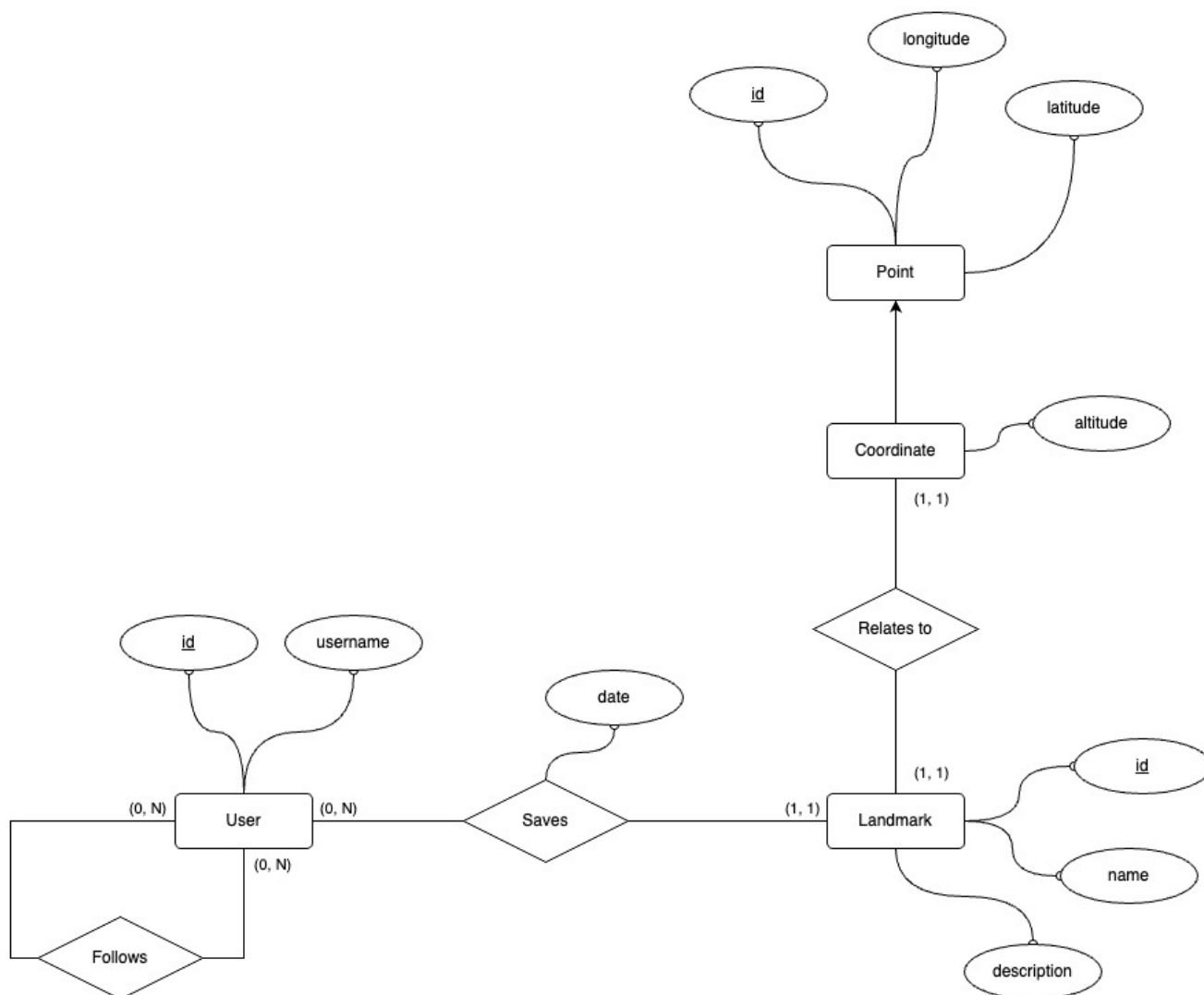
ER Diagram

We have four entities:

- user
- landmark
- coordinate
- point

and three relations:

- "follows": Many-to-Many, self-loop on User entity;
- "relates to": One-to-One, between Landmark and Coordinate;
- "saves": One-to-Many, between User and Landmark.



Project Structure

The application source code is structured in the following folders:

- **controller**: Controllers handle incoming HTTP requests, process them, and provide an appropriate response. They act as an intermediary between the client (hypothetical user interface) and the application's business logic;
- **model**: Models represent the application's data and business logic. They encapsulate the structure of data entities and provide methods for interacting with the data;
- **service**: Services contain the business logic of the application. They act as an intermediary between controllers and repositories, handling complex operations, and providing a higher-level API;
- **repository**: Repositories are responsible for data access and database interactions. They provide methods for querying, saving, and deleting entities;
- **util**: Utility classes contain common functionalities that can be reused across the application. The main class in this folder is **PersistenceManager**, which manages the lifecycle of **EntityManagerFactory** and provides methods for initializing, obtaining, and closing it.

By adhering to the MVC pattern, the project's structure encourages separation of concerns, making the codebase modular, maintainable, and scalable.