ERD Design and Development

The first step in developing my application was to design the Entity-Relationship Diagram (ERD). I used a modeling tool to visually represent the hospital's data model. The ERD is a crucial part of any information system because it lays out the data structure and defines how different entities in the system relate to each other. In this case, I identified key entities such as employees (doctors and nurses), departments, wards, and patients.

Each employee, whether a doctor or a nurse, has a unique employee number, surname, first name, address, and telephone number. Departments are characterized by a unique code, name, building location, and a director who is a doctor. Wards within these departments have a unique number, the number of beds, and a supervising nurse. Nurses are assigned to only one department, while doctors are not restricted to a specific department but are instead associated with their specialty. Patients are represented by a unique number, name, surname, address, and phone number and can be assigned to a specific ward and bed, along with their diagnosis details.

Creating JPA Entities

Once the ERD was finalized, the next step was to create the JPA entities based on this model. I used the Java Persistence API (JPA) to map these entities to corresponding database tables. Each entity in the Java code represents a table in the database. For example, there is an Employee class, from which Doctor and Nurse classes inherit. This class hierarchy reflects the hospital's real-world structure, where every doctor and nurse is also an employee. The Department and Ward classes have relationships with Employee to indicate the doctor who directs a department and the nurse who supervises a ward, respectively. The Patient class has relationships with both Ward and Doctor to show which ward the patient is in and which doctor is treating them.

Database Schema Generation

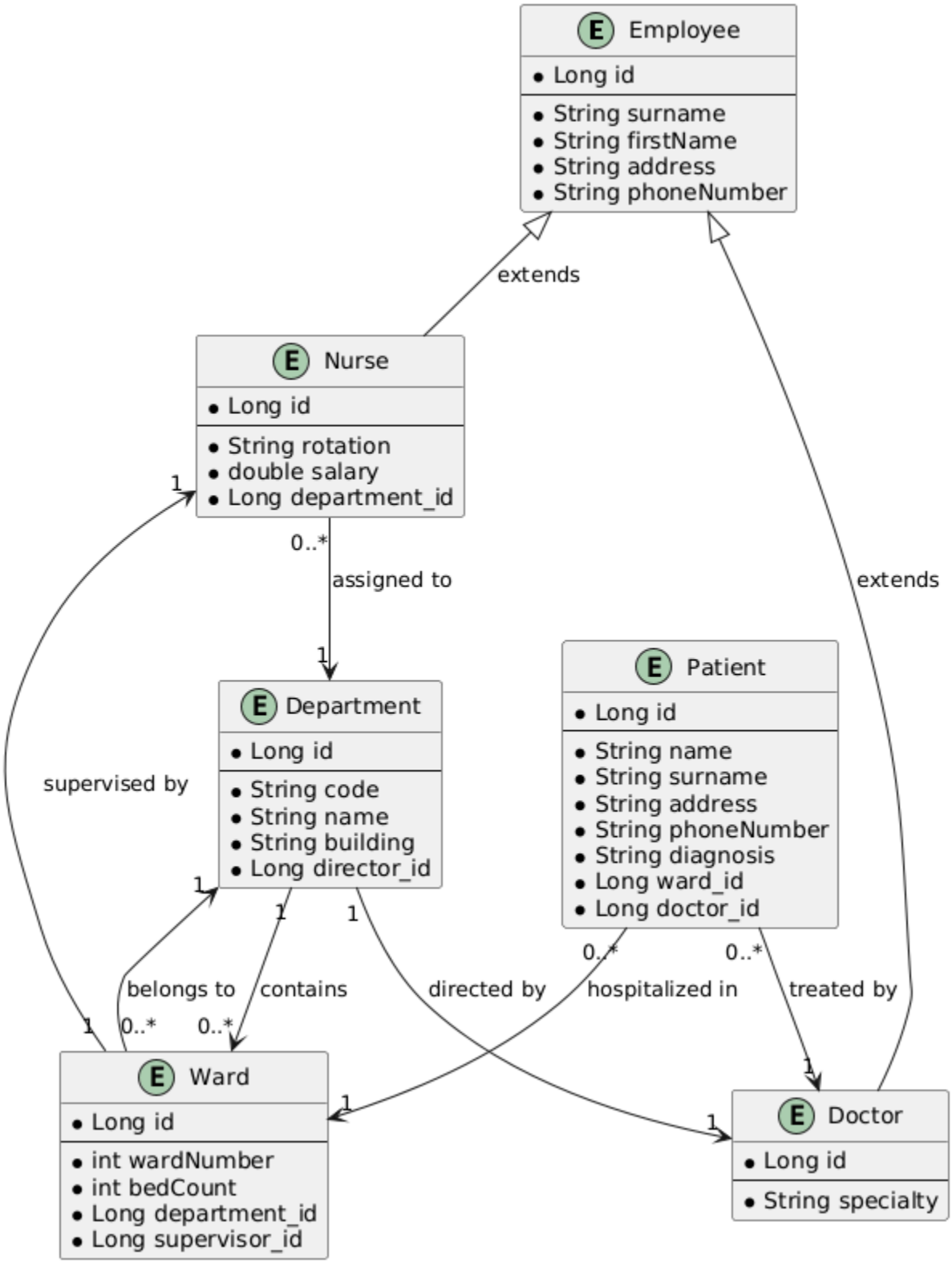
After defining the JPA entities, I configured my application to automatically generate the database schema using either PostgreSQL or MySQL. This auto-generation is facilitated by Spring Boot JPA's ability to create tables, columns, and relationships based on the Java entities. As I ran the application, the schema was created in the database, aligning closely with the original ERD. This automated process saves time and ensures consistency between the application code and the database schema.

Comparing Initial ERD with Generated ERD

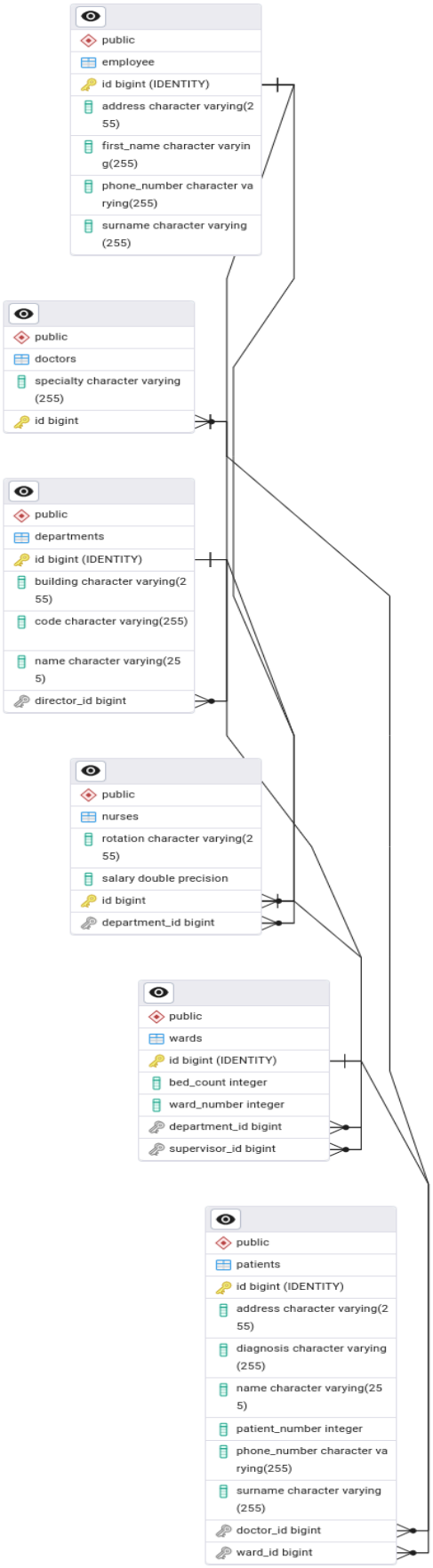
Once the database schema was generated, I compared it with my initial ERD design. This step is crucial to ensure that the implementation matches the design specifications and that all relationships are correctly mapped. The comparison revealed that the generated schema was consistent with the ERD, confirming that my entities and relationships were correctly implemented in the code. However, in some instances, minor adjustments were needed, such as

Proposed ERD diagram

tweaking fetch types or refining cascading operations, to optimize performance and ensure data integrity.



Proposed ERD diagram



Proposed ERD diagram

Actual ERD for the project

Proposed ERD diagram