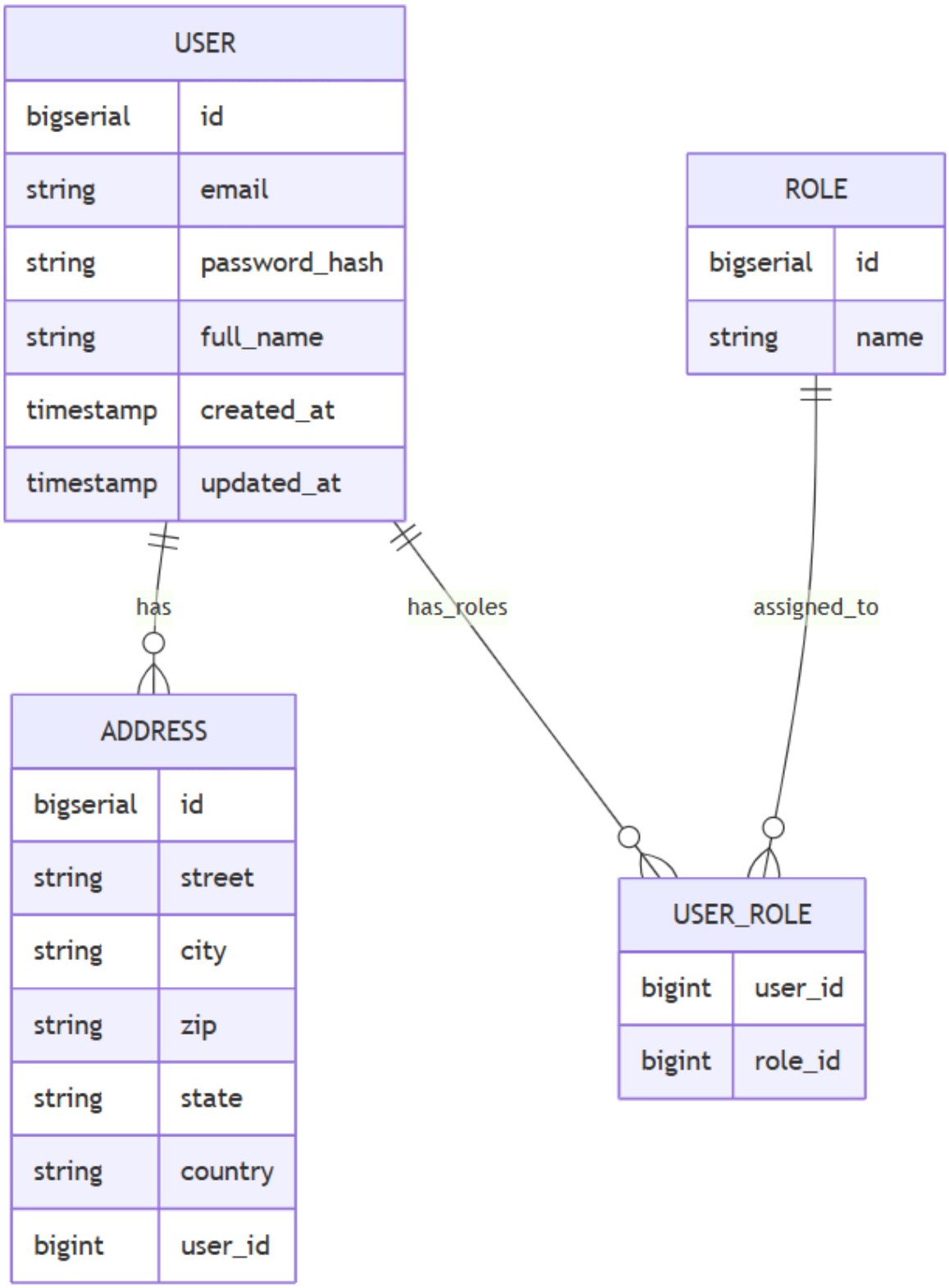
Database Schema

Each microservice has its **own database** (PostgreSQL) to enforce loose coupling. The schemas below outline the main tables and relationships for each service.

User Service Schema

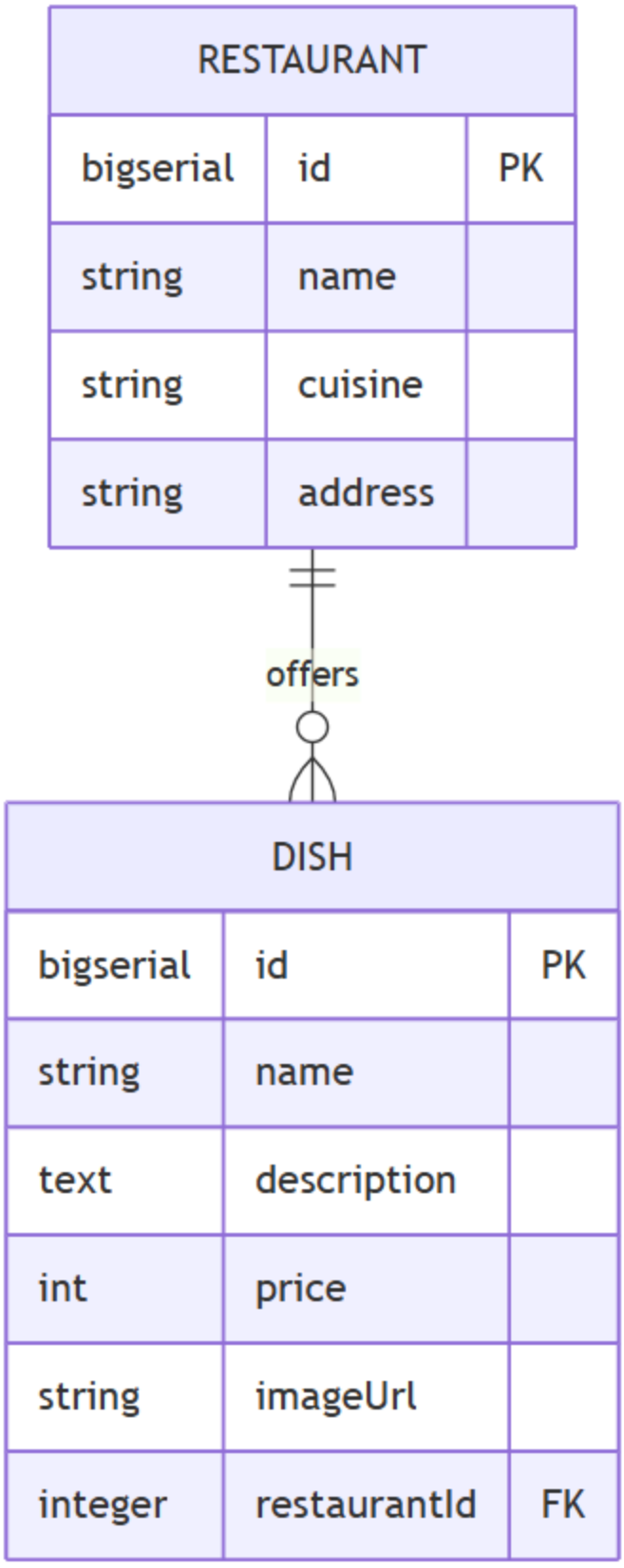


USER: Stores user accounts (id, email, encrypted password, name).

ADDRESS: Stores user addresses. Each user can have multiple addresses (one-to-many `USER -> ADDRESS`).

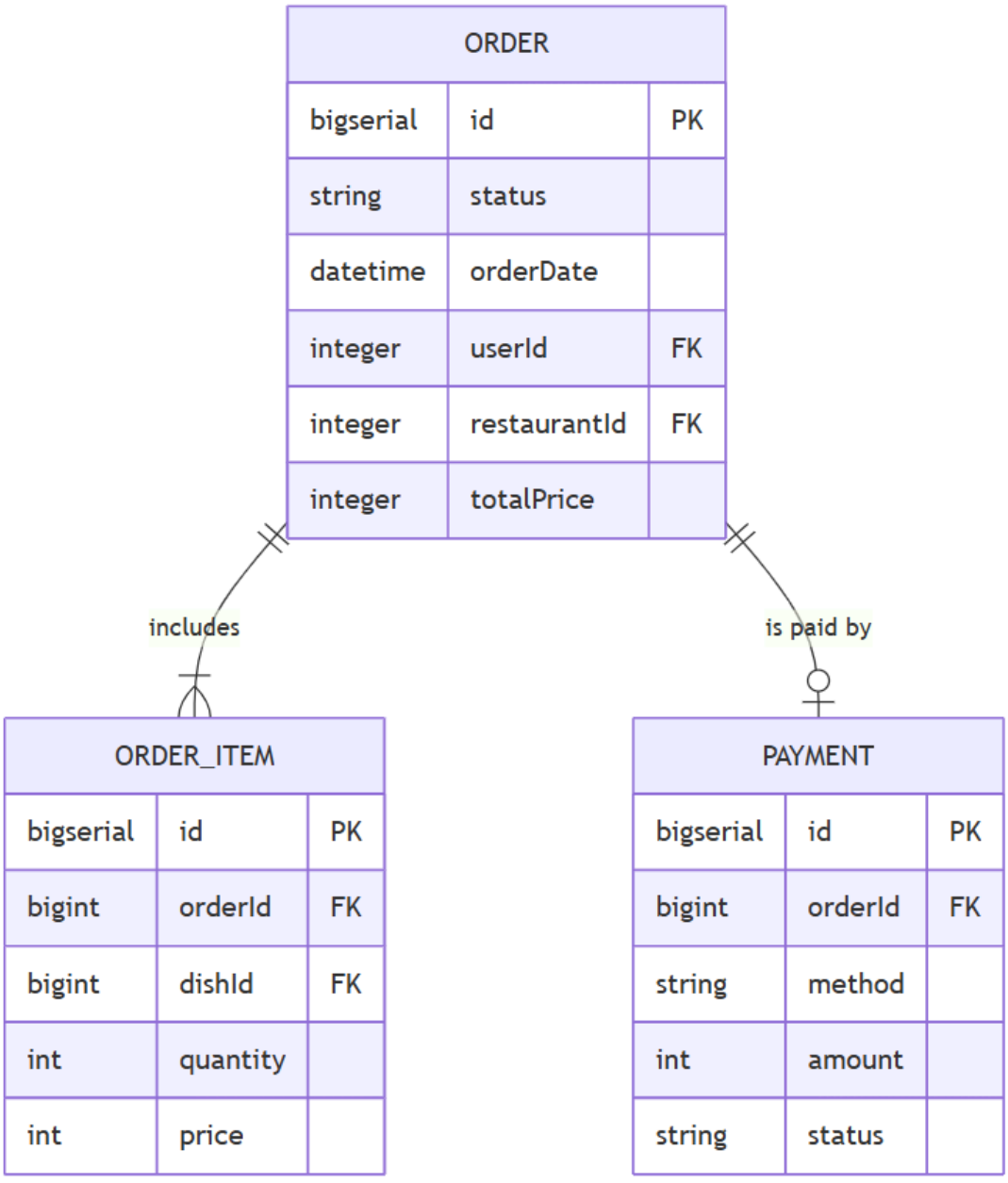
ROLE: Defines roles (`USER`, `ADMIN`). Users and roles have a many-to-many association (a user may have multiple roles).

Restaurant Service Schema


RESTAURANT
bigserial
id
PK
string
name
string
cuisine
string
address
offers
DISH
bigserial
id
PK
string
name
text
description
int
price
string
imageUrl
integer
restaurantId
FK

RESTAURANT: Each record has name, cuisine type, location, etc.

DISH: Menu items associated to a restaurant (`restaurantId` foreign key). A restaurant can have many dishes (one-to-many).



- **ORDER**: Stores each order's details (status, date, total price, user and restaurant IDs).
- **ORDER_ITEM**: Line items within an order. An order can have many items (one-to-many).
- **PAYMENT**: Records payment information for an order (method, amount, status). This is for the mock payment; in a real system it could be a separate Payment Service.

These schemas ensure normalized data with foreign keys. For example, each `ORDER` references a `USER` and a `RESTAURANT` by ID, but we rely on services to enforce integrity through their APIs (since services only access their own DB). Each microservice's database tables correspond to JPA entities annotated with `@Entity` and use repository interfaces for data access.