backend/

├─ app.js # 入口文件，启动服务

├─ package.json

├─ config/

│ ├─ db.js # 数据库连接配置

│ └─ default.js # 其他全局配置，例如端口、JWT密钥

├─ controllers/ # 控制器层，处理业务逻辑

│ ├─ authController.js

│ ├─ userController.js

│ ├─ permissionController.js

│ └─ reportController.js

├─ models/ # 数据模型，对应数据库表

│ ├─ userModel.js

│ ├─ permissionModel.js

│ └─ reportModel.js

├─ routes/ # 路由层，定义接口路径

│ ├─ authRoutes.js

│ ├─ userRoutes.js

│ ├─ permissionRoutes.js

│ └─ reportRoutes.js

├─ middleware/ # 中间件，如权限校验、日志、错误处理

│ ├─ authMiddleware.js

│ ├─ errorHandler.js

│ └─ logger.js

├─ utils/ # 工具函数

│ ├─ response.js # 统一返回格式

│ └─ token.js # JWT生成/解析

└─ seed/ # 初始数据/测试数据

└─ initData.js

 **app.js**

* 负责启动 Express 服务器
* 挂载全局中间件、路由

 **config/**

* 数据库配置、JWT密钥、默认端口等

 **models/**

* 对应 SQL Server 的表
* 可用 **sequelize** 或 **knex** 直接操作

 **controllers/**

* 处理具体业务逻辑
* 例如：用户登录、查询报告、权限校验

 **routes/**

* 定义 REST API 路径，例如：
* POST /login
* GET /users
* POST /users
* GET /reports
* 每个模块对应一个路由文件

 **middleware/**

* **authMiddleware.js** → 校验用户登录状态、Token
* **errorHandler.js** → 统一捕获错误
* **logger.js** → 请求日志

 **utils/**

* 放一些通用工具函数，比如统一返回、生成JWT

 **seed/**

* 测试或初始化数据，方便前端调接口