四川大學

SICHUAN UNIVERSITY

医院门诊挂号系统

数 据 库 原 理 实 验 期 末 报 告

任课教师 段磊

学　　院 计算机学院

专　　业 计算机科学与技术

组　　别 第四组

组　　长 郑仕博

成　　员 吴正博，高俊翔

2025 年 12 月 23 日

目录

1 引言

1.1 目的

本系统的目的是围绕真实门诊挂号场景构建一套可运行的示例系统，覆盖号源查询、挂号、支付/取消、就诊记录与站内消息等完整业务链，并在实现过程中突出数据库的核心能力：关系建模、主外键约束与事务一致性控制。

1.2 项目范围

系统覆盖门诊挂号的核心业务链：账号与角色、科室/医生信息、号源（排班）管理、在线挂号、余额支付与取消退款、就诊病历、站内消息与权限控制。系统包含四类角色：患者、医生、科室管理员、系统管理员。

1.3 需求分析

医院挂号系统旨在解决传统挂号模式中存在的排队时间长、号源管理混乱、信息查询不便等问题，通过数字化方式实现挂号流程的规范化、高效化，同时满足患者、医生、医院管理员等不同角色的核心业务需求，为医院诊疗工作的有序开展提供技术支撑。本系统需保证数据的安全性、操作的易用性以及功能的完整性，符合医院日常运营的业务逻辑。

1.4 文档概览

第 1 部分说明背景与范围；第 2 部分给出系统概览与典型流程；第 3 部分描述整体架构与关键实现策略；第 4 部分给出数据库 E-R 图、关系模式与约束；第 5 部分说明各业务模块的实现；第 6 部分概述人机界面设计；第 7 部分介绍测试与验收；第 8 部分为需求矩阵；第 9 部分总结与展望；第 10 部分为附录，包含项目目录说明、环境要求、配置文件说明、部署与运行、示例数据说明及演示账号与建议流程。

1.5 术语与缩略语

E-R 图（实体联系图）、PK（主键）、FK（外键）、ACID（事务特性）、REST（接口风格）、Slot（号源/时间段）、Reg（挂号记录）、RBAC（基于角色的访问控制，本文实现为简化版）。

2 系统概览

2.1 系统目标

以“数据库为中心”完成一个可运行的 Web 项目，能够体现数据库原理课程的核心点：关系模式设计、主外键约束、典型增删改查、关键业务规则的一致性维护（容量、金额、状态流转）以及面向多角色的权限控制。

2.2 系统定位与场景假设

本系统定位于教学场景下的门诊挂号流程演示，围绕“号源、挂号、支付、消息与权限”等数据库课程的典型问题展开。为保持实验聚焦，支付方式采用账户余额扣款，不引入第三方支付；鉴权采用 `token=userId` 的简化方案；数据库建表与数据初始化通过脚本手动执行，突出关系模式与约束设计的可复现性。演示数据主要由 `sql/sample_data.sql` 提供，便于在有限时间内覆盖完整业务链路。

2.3 角色与权限

- 患者（patient）：注册/登录；查询科室、医生与可预约号源；创建挂号、支付/取消、充值；查看挂号与就诊记录；在满足规则下发送站内消息。
- 医生（doctor）：查看某日挂号患者列表；为已挂号记录录入病历；向患者/科室管理员/系统管理员发送消息（受“已支付挂号 +10 天有效期”约束）。
- 科室管理员（dept_manager）：维护本科室医生归属；为医生创建号源（日期、起止时间、容量、费用、备注）；查看本科室号源与挂号记录；按规则向本科室医生/患者发送消息。
- 系统管理员（admin）：创建医生/科室管理员/系统管理员账号；冻结/解冻账号；通过消息机制发送系统通知。

2.4 团队与分工

成员	核心职责	备注
郑仕博	前端美化和后端与核心业务（挂号、支付、取消、权限）	接口联调与验收准备
吴正博	前端页面与交互、消息与工作台体验	交互打磨与展示支持
高俊翔	数据库建模、SQL 脚本、ER 图与文档	演示录屏与流程串联

2.5 典型业务流程

1. 管理员初始化：执行 `sql/schema.sql` 建表，执行 `sql/sample_data.sql` 插入样例数据；使用系统管理员账号登录进行账号维护，使用科室管理员账号登录进行号源维护。

2. 患者挂号：选择科室与日期查询医生号源（含起止时间、余量与费用）→ 选择 `slotId` 创建挂号 → 余额支付或取消（已支付按 90% 退款）。

3. 医生就诊：医生端按日期查看挂号患者列表 → 对某挂号记录录入病历（诊断、治疗方案、医嘱）→ 患者在个人中心查看就诊记录。

4. 消息沟通：基于“已支付挂号关系 + 双方最近消息 10 天内活跃 + 角色限制”校验发送权限；支持会话列表与会话明细查询。

3 系统架构

3.1 总体架构

系统采用前后端分离三层结构：

- 前端：Vue2，实现登录/注册、挂号页、消息页、个人中心与各角色工作台；通过代理将 `/api` 请求转发到后端。
- 后端：Spring Boot + MyBatis，提供 REST 接口（统一前缀 `/api`），业务逻辑集中在 Service 层，DAO 使用 Mapper 访问 PostgreSQL。
- 数据库：PostgreSQL，采用“手动执行脚本建库建表”的方式，保证验收时数据库模式清晰可复现。

3.2 技术选型与原因

- Spring Boot：快速搭建 REST 服务，便于按模块组织 Controller/Service/Mapper。
- MyBatis：SQL 可控、便于体现课程中对查询与事务的设计能力。
- PostgreSQL：支持标准 SQL 与完整约束语义，适合课程要求的关系模型实现。
- Vue2：开发效率高，便于按角色组织页面与交互流程。

3.3 接口规范与前后端协作

- 接口统一前缀为 `/api`，前后端以 JSON 进行数据交互。
- 登录成功后返回 `userId`，`username`，`role`，`token`，其中 `token` 简化为 `userId`。

- 前端将 token 存入本地缓存，并在请求头携带 X-User-Id 供后端识别用户与角色。
- 开发环境下前端通过代理将 /api 请求转发至后端服务，避免跨域问题。

3.4 鉴权与会话策略（实验简化）

为了聚焦数据库与业务一致性，本项目未引入完整 JWT/Spring Security。登录成功后返回的 token 简化为 userId，前端在请求头携带 X-User-Id，后端据此识别当前用户并做角色校验。

3.5 接口调用与数据流

系统数据流按“前端页面 → REST 接口 → Service 业务逻辑 → Mapper 持久化 → PostgreSQL”的路径执行。前端负责参数收集与状态展示，后端负责业务校验与事务一致性，数据库负责约束与持久化。数据库脚本与示例数据独立于应用运行，可在验收时直接复现完整环境。

3.6 关键一致性策略

- 号源容量一致性：创建/取消挂号在同一事务内同时写入 registration 并更新 doctor_time_slot.booked_count，避免出现“挂号成功但容量未扣减”等不一致。
- 事务边界：挂号创建、支付与取消均以事务方式执行，保证“状态更新 + 金额/余量变更”原子提交或回滚。
- 金额统一以“分”存储：patient.money、doctor_time_slot.fee、registration.reg_fee 统一使用整数分，避免浮点误差；前端录入“元”，后端转换为“分”落库。
- 业务约束校验：不可预约过去日期；号源不足禁止挂号；同一患者同一号源仅允许一条未取消记录；号源时间段不可重叠；取消已支付挂号按 90% 退款；消息需满足“付费挂号关系 + 10 天有效期”。

4 数据设计

4.1 ER 图

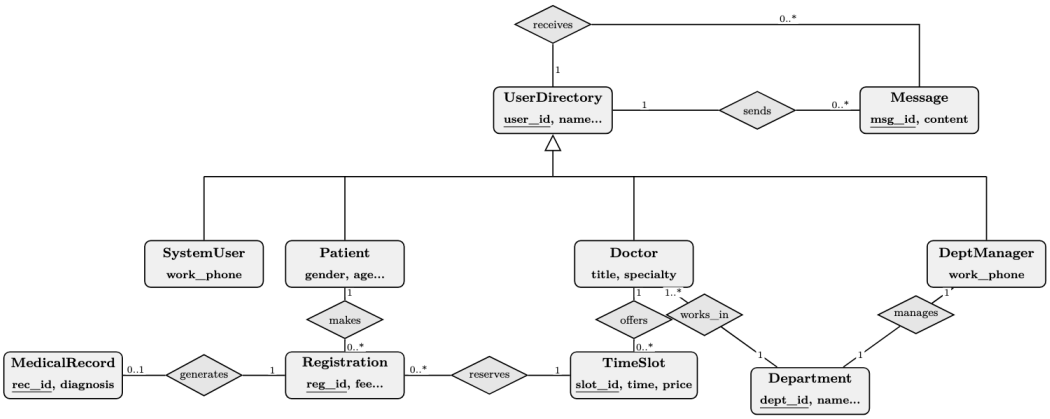

图 1: 数据库核心实体关系图

4.2 关系模式图

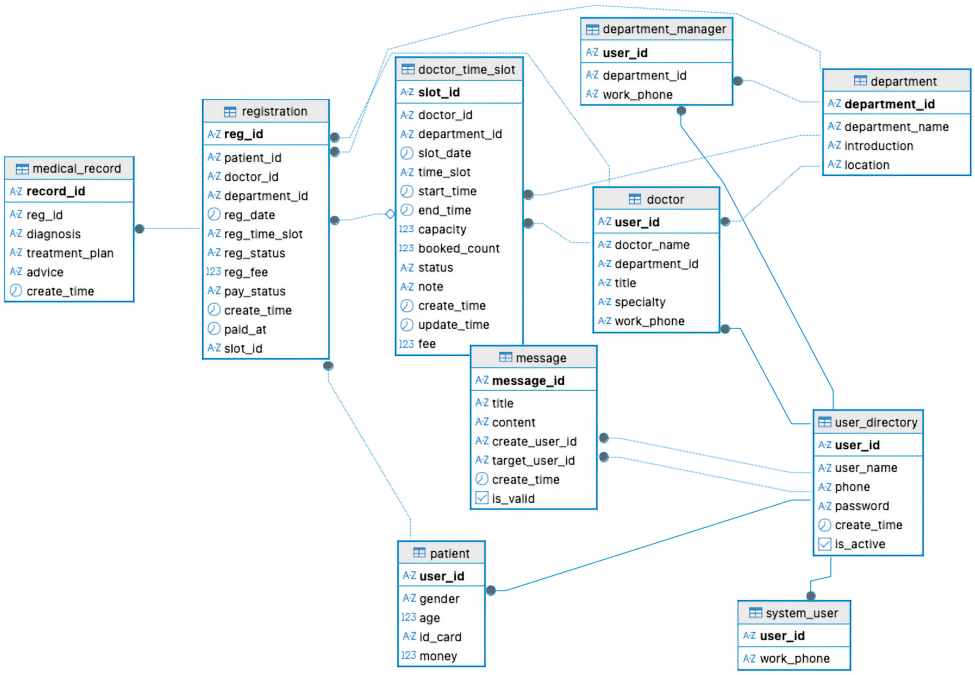

图 2: 数据库关系模式图

4.3 字段设计说明

- 统一用户表 + 角色子表：`user_directory` 负责基础身份信息，`patient/doctor/department_ma`
  存放角色扩展字段。

- 金额字段统一使用“分”:patient.money、doctor_time_slot.fee、registration.reg_fee 避免浮点误差。
- 时间段双存储:start_time/end_time 供约束校验,time_slot 与 reg_time_slot 作为可读展示。
- 状态字段枚举化：reg_status 与 pay_status 用于控制流程与退款逻辑。

4.4 设计要点与规范

- 角色扩展的可维护性：统一用户表避免重复存储基础信息，角色表只保存角色特有字段，便于权限校验与扩展。
- 号源与挂号的分离：号源负责容量与时间段，挂号记录负责状态流转，两者通过 slot_id 关联，利于统计与一致性维护。
- 消息的可追溯性：消息记录保留发送方/接收方与时间戳，便于验收解释沟通路径。
- 命名与单位规范：字段命名保持语义一致，金额使用“分”，时间采用“日期 + 起止时间”的组合表达。

4.5 关系模式与数据字典

表结构与字段含义以 sql/schema.sql 和 requirements/数据表设计.md 为准，核心表如下:

表名	核心字段（节选）	说明
user_directory	user_id, user_name, phone, password, create_time, is_active	统一用户表（四类角色共享）
patient	user_id, gender, age, id_card, money	患者扩展信息，money 单位分
department	department_id, department_name, introduction, location	科室主数据
doctor	user_id, doctor_name, department_id, title, specialty	医生信息，关联科室
department_manager	user_id, department_id, work_phone	科室管理员，关联科室
"system_user"	user_id, work_phone	系统管理员
doctor_time_slot	slot_id, doctor_id, slot_date, start_time, end_time, capacity, booked_count, fee	医生号源（排班）

表名	核心字段（节选）	说明
registration	reg_id, patient_id, doctor_id, reg_date, reg_time_slot, reg_status, reg_fee, pay_status, slot_id	挂号记录与状态流转
medical_record	record_id, reg_id, diagnosis, treatment_plan, advice	病历，一次挂号对应一条记录
message	message_id, title, content, create_user_id, target_user_id, create_time, is_valid	站内消息

4.6 关键约束与规则落地

- 主外键约束：在 sql/schema.sql 中通过 FK 约束保障实体引用完整性（如 registration.patient_id 引用 patient.user_id）。
- 手机号唯一：user_directory.phone 设置唯一约束，避免重复注册。
- 号源时间段不重叠：科室管理员创建号源时，后端在同一医生同一日期内检查 start_time/end_time 区间是否与已有号源重叠。
- 容量控制：创建挂号前校验 capacity > booked_count；成功后在事务内将 booked_count + 1，取消时 booked_count - 1。
- 同号源重复挂号限制：创建挂号前按 (patient_id, slot_id) 查询是否存在未取消记录，避免重复占号。
- 取消退款：已支付挂号取消按 90% 退款并回写余额；未支付直接取消。
- 消息权限与时效：发送消息需满足“已支付挂号关系”且双方最近消息在 10 天内；超期仅可查看历史，不允许继续发送。

4.7 一致性控制示例

1. 挂号创建：在同一事务内写入 registration 并更新号源余量，防止并发下出现“超卖”。
2. 支付与取消：支付记录 paid_at 并更新 pay_status；取消时按规则退款并回写余额。
3. 消息时效：发送消息前校验“已支付挂号关系 + 10 天活跃期”，不满足条件则拒绝创建消息记录。

4.8 状态字段约定

- 挂号状态 registration.reg_status：Booked（已预约）、Canceled（已取消）、Finished（已完成，就诊流程可扩展）。
- 支付状态 registration.pay_status：Unpaid（未支付）、Paid（已支付）、Refunded（已退款/作废）。

4.9 脚本与示例数据

项目不自动建表。验收或演示时，建议使用数据库管理工具连接 PostgreSQL 并执行：sql/schema.sql 创建表结构，sql/sample_data.sql 插入示例数据。示例数据覆盖：

- 10 个科室（D001 ~ D010）
- 10 个患者、10 个医生、10 个科室管理员、10 个系统管理员
- 多个医生排班号源与多条挂号记录（包含已支付、未支付与取消状态）
- 初始病历、站内消息与操作日志等辅助数据

5 组件设计

5.1 账号与用户管理

- 患者注册：POST /api/auth/patient/register，写入 user_directory 与 patient（统一用户 + 角色扩展信息）。
- 登录：POST /api/auth/login，支持使用 userId 或手机号登录；返回 role 与 token=userId。
- 修改密码：POST /api/auth/change-password。
- 账号创建与冻结/解冻：系统管理员通过 /api/users/* 创建医生/科室管理员/系统管理员账号，并通过更新 user_directory.is_active 控制账号可用性。

5.2 科室、医生与号源

- 科室列表：GET /api/departments，提供科室名称与地点，供挂号页筛选。
- 医生列表：GET /api/doctors?departmentId=&date=，按科室与日期返回医生信息与可预约号源（含起止时间、余量与费用）。

- 号源查询：GET /api/registrations/slots 支持按科室/日期检索号源。
- 号源创建：科室管理员通过 POST /api/dept/slots 为本科室医生创建号源；时间段校验 start_time < end_time 且不重叠。

5.2.1 号源创建（科室管理员）实现步骤

1. 权限校验：当前用户必须是科室管理员，且只能为本科室医生创建号源。
2. 参数校验：日期、起止时间、容量、费用必填，且满足 start_time < end_time、capacity > 0。
3. 重叠检查：查询该医生该日期的已有号源，若存在区间重叠（非 end <= existing.start 且非 start >= existing.end）则拒绝创建。
4. 落库：生成 slot_id；将展示字段 time_slot 固化为 “HH:mm-HH:mm”；费用由 “元” 换算为 “分”；初始化 booked_count=0 与 status=OPEN。

5.3 挂号、支付与取消

- 创建挂号：POST /api/registrations，必须选择 slotId；校验日期合法、医生科室匹配、号源余量；写入 registration 并扣减余量。
- 支付：POST /api/registrations/{regId}/pay，从 patient.money 扣减 registration.reg_f 写入支付时间并更新状态。
- 取消：POST /api/registrations/{regId}/cancel，更新挂号状态；已支付按 90% 退款；释放号源余量。
- 查询挂号：患者 GET /api/registrations，医生端 GET /api/doctor/registrations。

5.4 账户余额与充值

- 查询余额：GET /api/patient/me 返回当前账户余额与基本信息。
- 充值：POST /api/patient/recharge 以 “元” 为输入单位，后端转换为 “分” 入账。
- 支付校验：挂号支付时优先校验余额充足，不足则提示先充值。

5.4.1 创建挂号实现步骤

1. 校验 regDate 不能早于当天。
2. 校验医生属于所选科室，防止 “科室/医生不匹配” 造成脏数据。

3. 通过 slotId 定位号源，并校验号源的医生/科室/日期与请求一致。
4. 校验余量：capacity > booked_count 才允许创建。
5. 校验重复挂号：同一患者同一 slotId 若存在未取消记录则拒绝创建。
6. 写入 registration：复制费用到 reg_fee；将 reg_time_slot 统一保存为“HH:mm-HH:mm”；初始化 reg_status=Booked、pay_status=Unpaid。
7. 更新号源：booked_count + 1（与挂号写入处于同一事务内）。

5.4.2 支付与取消实现步骤

1. 支付：校验余额充足后扣减 patient.money，并将 pay_status 更新为 Paid，记录 paid_at。
2. 取消：仅允许取消 Booked 状态；已支付按 90% 退款并更新余额；将挂号状态置为 Canceled/Refunded 并释放号源（booked_count - 1）。

5.5 病历与就诊记录

医生对挂号记录录入病历(诊断、治疗方案、医嘱),患者与医生端分别提供查询接口,保证一次挂号对应一份可追溯的就诊记录。接口包括 POST /api/medical-records(医生录入)、GET /api/medical-records(患者查询)与 GET /api/medical-records/doctor(医生查询)。

5.6 站内消息

消息模块支持收件箱/发件箱、会话列表(最近一条)与会话明细。发送消息时进行角色与关系校验,并结合“10 天活跃期”限制控制继续对话的权限。主要接口包括 POST /api/messages、GET /api/messages、GET /api/messages/conversations 与 GET /api/messages/conversation。

5.6.1 消息权限与 10 天有效期

1. 角色限制：按发送方/接收方角色确定可通信范围（患者仅能向就诊医生或对应科室管理员发送；医生可向就诊患者/本科室管理员/系统管理员发送；科室管理员可向本科室医生与本科室就诊患者发送；系统管理员可向任意用户发送）。
2. 关系校验：对“患者 ↔ 医生/科室管理员”等关系，要求存在近 10 天内的**已支付挂号**记录。
3. 会话活跃校验：若双方最近一条消息时间超过 10 天，禁止继续发送（历史可查看）。

4. 撤回/作废：通过 message.is_valid 进行逻辑失效，不物理删除，便于追溯。

5.7 会话展示与作废逻辑

- 会话列表接口返回每位联系人最近一条消息，按时间倒序展示，便于快速定位活跃对话。
- 会话明细接口返回双方历史消息列表，保留完整时间线，便于复盘就诊沟通。
- 作废消息仅在逻辑层隐藏，不物理删除，满足课程对可追溯性的要求。

6 人机界面设计

前端按角色展示不同导航与工作台：

- 登录/注册：患者注册与四类用户登录入口。
- 挂号页（患者）：选择科室与日期，展示医生与可约号源，完成挂号创建。
- 消息页（全角色）：会话列表 + 聊天明细 + 发送消息。
- 个人中心（全角色）：查看个人信息、修改密码；患者可充值与管理挂号记录；医生可查看挂号与病历。
- 医生端：按日期查看患者挂号并录入病历。
- 科室端：维护医生归属、创建号源、查看本科室挂号记录。
- 系统管理端：创建账号、冻结/解冻、发送系统通知。

6.1 页面与接口映射（节选）

- 登录/注册页：/api/auth/patient/register、/api/auth/login、/api/auth/change-passwor
- 挂号页：/api/departments、/api/doctors、/api/registrations。
- 消息页：/api/messages/conversations、/api/messages/conversation、/api/messages。
- 个人中心：/api/registrations、/api/medical-records、/api/patient/me、/api/patient/recharge。
- 医生端：/api/doctor/registrations、/api/medical-records。
- 科室端：/api/dept/doctors、/api/dept/slots、/api/dept/registrations。
- 系统管理端：/api/users/*（创建账号、冻结/解冻）。

6.2 角色工作台说明

- 患者端：围绕“挂号、支付、取消、消息、就诊记录”组织入口，主路径集中在挂号页与个人中心。
- 医生端：关注“当日挂号列表 + 病历录入 + 消息沟通”，以效率为导向减少跳转。
- 科室管理员端：以“号源创建/查看、医生归属维护、挂号统计”为主，支持快速管理号源。
- 系统管理员端：集中处理账号创建与冻结/解冻，并通过消息机制发布通知。

6.3 交互流程示意

1. 患者端：登录/注册 → 选择科室与日期 → 查看医生与号源 → 创建挂号 → 支付或取消 → 查看就诊记录与消息。
2. 医生端：登录 → 查看当天挂号患者 → 录入病历 → 发出就诊提醒或随访消息。
3. 管理端：登录 → 创建账号/维护号源 → 查看挂号记录 → 发送系统通知。

7 测试与验收

7.1 测试数据准备

1. 执行 `sql/schema.sql` 创建表结构。
2. 执行 `sql/sample_data.sql` 插入示例数据，包含多角色账号与典型挂号记录。
3. 使用系统管理员账号 `AD0001 / admin123` 登录，检查账号创建与冻结功能。

7.2 典型流程验证

1. 患者流程：注册/登录 → 查询号源 → 创建挂号 → 支付/取消 → 查看挂号与就诊记录。
2. 医生流程：查看当日挂号列表 → 录入病历 → 向患者发送就诊提醒。
3. 管理员流程：创建账号与号源 → 查看挂号记录 → 发送系统通知。

7.3 异常与约束场景

- 号源容量不足或日期非法时禁止挂号，提示用户调整时间段。
- 同一患者同一号源重复挂号被拒绝，避免重复占号。
- 余额不足时支付失败，需要先充值再支付。
- 消息超过 10 天未活跃时禁止继续发送，仅可查看历史记录。

8 需求矩阵

需求	实现（主要接口/数据表/页面）
登录/注册与修改密码	/api/auth/*；user_directory、patient；登录注册页、个人中心
科室与医生号源查询	/api/departments、/api/doctors；department、doctor、doctor_time_slot；挂号页
创建挂号、支付与取消	/api/registrations、/api/registrations/{id}/pay、/api/registrations/{id}/cancel；registration、doctor_time_slot、patient；个人中心
号源维护（科室管理员）	/api/dept/slots、/api/dept/doctors；doctor_time_slot；科室端
病历与就诊记录	/api/medical-records；medical_record、registration；医生端、个人中心
站内消息与权限时效	/api/messages*；message、registration；消息页
账号创建与冻结/解冻	/api/users/*；user_directory 与角色子表；系统管理端

表 2: 需求—实现矩阵（节选）

9 总结与展望

本项目以门诊挂号为核心场景，完成了从需求分析、数据库建模、接口实现到前端交互的完整闭环，能够体现课程强调的关系模式设计、约束表达与事务一致性控制。通过统一用户表、号源-挂号分离、消息与权限规则等设计，保证了多角色协同下的数据一致与可追溯性。

后续可扩展方向包括：

- 引入更完整的鉴权与权限体系（JWT/RBAC），提升安全性与可维护性。
- 增加统计报表与运营类查询，丰富数据分析能力。
- 引入容器化部署与自动化测试，提升项目可交付性与稳定性。

10 APPENDICES

10.1 项目目录说明

以项目根目录为基准，主要内容如下：

- `backend/`：后端 Spring Boot + MyBatis 源码与配置。
- `medical-regist/`：前端 Vue2 工程与页面资源。
- `sql/`：数据库建表与示例数据脚本（`schema.sql`、`sample_data.sql`）。
- `requirements/`：需求分析、数据表设计、模块接口等文档。
- `ppt/`：课程汇报 PPT 源文件。
- `期末报告/`：期末报告 LaTeX 源文件与相关资源。
- `README.md`、`SYSTEM_README.md`：项目说明与运行指南。
- `演示视频.mp4`：功能演示录屏（位于项目根目录）。

10.2 环境要求

- JDK 17
- Maven 3.9+
- Node.js 16+ 与 npm/pnpm
- PostgreSQL 16（本地默认 `localhost:5432`）

10.3 配置文件说明

- 后端数据库连接：`backend/backend/src/main/resources/application.properties`
- 前端开发代理：`medical-regist/vue.config.js`
- 前端请求封装：`medical-regist/src/api.js`

10.4 部署与运行

1. 准备 PostgreSQL（本地 `localhost:5432`），连接后执行 `sql/schema.sql`（可选执行 `sql/sample_data.sql`）。
2. 后端配置运行 mvn compile 和 mvn package 进行编译后，用 java -jar 运行生成的 jar 包。
3. 前端安装依赖 npm install , 编译静态文件 npm run build ， 配置服务器能够访问静态文件。
4. 使用 nginx 将前端 /api/ 请求代理到后端使静态文件（即前端）可以向后端发送请求。
5. 可以通过 http://database.zhengshibo.top:2000/访问前端页面。

10.5 示例数据说明

- `sql/sample_data.sql` 提供角色账号、科室、号源与挂号记录，覆盖演示所需的完整流程。
- 若不使用示例数据，患者仍可通过前端注册创建账号，并由管理员补充医生与号源。

10.6 演示账号与建议流程

若执行了 `sql/sample_data.sql`, 可使用系统管理员账号 `AD0001 / admin123` 登录进行演示；患者可通过前端注册创建。建议演示顺序：患者挂号（创建/支付/取消）→ 医生查看挂号并写病历 → 双方消息沟通 → 管理端创建账号与冻结解冻。