

# USTC 数据库学习/考试系统项目文档

## 1. 项目概述

- 背景：面向数据库课程的教学、练习与考试管理，支持分布式授课和多样化题型。
- 目标：提供统一的课程资源管理、在线学习、考试监考、成绩分析与教学反馈平台。
- 系统定位：B/S 架构，前端 Vue 3 单页应用，后端 FastAPI 服务化，PostgreSQL 存储数据，可扩展为多节点部署。

## 2. 核心功能

- 用户管理：注册、登录、找回密码、角色授权（管理员/教师/学生）。
- 课程管理：课程创建、班级分组、资源发布、学员管理。
- 知识学习：课程内容展示、富文本/Markdown 讲义、视频/附件、学习进度追踪。
- 题库管理：选择题、填空题、编程题、SQL 运行题；标签、难度与版本控制。
- 考试中心：考试编排、随机出题策略、时间/权限控制、监考（防刷题、防切屏）。
- 提交与批阅：自动判题（客观题+SQL sandboxes）、教师人工批阅、评语录入。
- 成绩分析：个人成绩单、班级统计、题目难度分析、导出报表。
- 通知与消息：系统公告、考试提醒、成绩推送，可对接邮箱/短信。
- 系统管理：参数配置、审计日志、数据备份、第三方认证。

## 3. 技术栈与架构

层级	方案	说明
前端	Vue 3 + Vite + TypeScript, Pinia, Vue Router, Element Plus	单页应用，组件化开发； Pinia 统一状态，Axios 请求封装。
后端	FastAPI + Python 3.11, SQLAlchemy 2.x, Pydantic v2, Alembic	REST 接口，JWT 鉴权； ORM + 异步数据库；数据 库迁移。
数据库	PostgreSQL 15	支持 JSONB、全文搜索、 CTE，适合考试统计与日

		志。
缓存/ 消息	Redis (会话、验证码、限流); 可选 RabbitMQ/Kafka (异步任务)	提升性能与扩展性。
文件/ 对象存 储	MinIO / 阿里云 OSS / AWS S3	存储附件、富文本资源。
部署	Docker Compose/Helm, Nginx 反 向代理, Caddy/Traefik 可选	支持本地开发与云上部署。

### 3.1 逻辑架构

1. **Client SPA**: 负责页面渲染与交互, 通过 REST 调用后端。
2. **API Gateway/Backend**: FastAPI 提供认证、课程、考试、统计等服务。
3. **Task Workers**: Celery + Redis/RabbitMQ 处理异步批阅、报表、消息。
4. **DB & Storage**: PostgreSQL 存结构化数据; 对象存储保存静态资源。
5. **Observability**: Prometheus + Grafana 监控; Sentry 捕获前端/后端错误。

## 4. 前端设计

### 4.1 目录建议

```
web/
├── src/
│   ├── api/          # Axios 实例、REST 封装
│   ├── assets/
│   ├── components/
│   ├── composables/
│   ├── layouts/
│   ├── router/
│   ├── stores/        # Pinia store
│   ├── views/         # 页面视图
│   ├── utils/
│   └── types/
└── public/
└── vite.config.ts
└── package.json
```

### 4.2 关键模块

- **鉴权流程**: 路由守卫 + Refresh Token + 权限指令。
- **UI 布局**: 基于 Element Plus, 自适应课程/考试工作台与学生端。
- **考试界面**: 题目切换组件、答题卡、计时器、防切屏事件监听。
- **实时功能**: WebSocket 订阅消息 (考试通知、成绩更新)。
- **国际化/多语言**: 可选 Vue I18n。
- **可访问性**: Tab 键导航、ARIA 标签、深浅色模式。

## 5. 后端设计

### 5.1 服务模块

- auth : JWT/OAuth2, 短信/邮箱验证码, 权限模型 (RBAC)。
- users : 个人信息、角色管理、学习档案。
- courses : 课程 CRUD、资源、章节、班级。
- content : 富文本、附件、视频元数据。
- question\_bank : 题目管理、标签、版本。
- exam : 考试计划、策略、监考、反作弊。
- submission : 答卷记录、评分、评语。
- analytics : 成绩统计、题目难度、学习行为。
- notifications : 站内信、邮件任务。
- system : 配置项、审计日志、数据导出。

### 5.2 数据库模型 (示例)

表名	关键字段
users	<code>id, username, password_hash, role, email, status</code>
user_profiles	<code>user_id, real_name, student_id, mobile, avatar_url</code>
courses	<code>id, name, code, teacher_id, description, visibility</code>
course_members	<code>id, course_id, user_id, role, group_tag</code>
chapters	<code>id, course_id, title, order, content</code>
learning_progress	<code>id, course_id, user_id, chapter_id, status, last_viewed_at</code>
questions	<code>id, type, content, options, answer, tags, difficulty, owner_id</code>
exams	<code>id, course_id, title, start_at, end_at, duration, mode, status</code>
exam_papers	<code>id, exam_id, question_id, score, section, order, random_seed</code>
exam_access_rules	<code>id, exam_id, limit_ip, limit_device, anti_cheat</code>

<code>submissions</code>	<code>id, exam_id, user_id, submitted_at, score, status</code>
<code>submission_answers</code>	<code>id, submission_id, question_id, answer, score, feedback</code>
<code>attachments</code>	<code>id, owner_type, owner_id, url, content_type, size</code>
<code>notifications</code>	<code>id, channel, title, payload, status, sent_at</code>

- 索引建议: `questions.tags` 使用 GIN; `submissions.exam_id_user_id` 复合唯一; `learning_progress` 通过 `course_id+user_id` 优化查询。

### 5.3 API 规范

- RESTful + JSON; 使用 `/api/v1`.
- OAuth2 Password + Refresh Token; Header `Authorization: Bearer <token>`.
- 全局响应 `{ code, message, data, trace_id }`.
- 错误码: `1xxx` 认证/权限, `2xxx` 业务, `5xxx` 系统。
- OpenAPI 文档通过 `FastAPI` 自带 `/docs // redoc`。
- 速率限制: `fastapi-limiter` + Redis。

#### 5.3.1 API 示例

模块	方法 & 路径	描述
Auth	<code>POST /api/v1/auth/login</code>	用户登录, 返回 Access/Refresh Token
Auth	<code>POST /api/v1/auth/refresh</code>	刷新 Token
Users	<code>GET /api/v1/users/me</code>	获取当前用户信息
Courses	<code>POST /api/v1/courses</code> (教师/管理员)	创建课程
Courses	<code>GET /api/v1/courses/{id}</code>	课程详情
Question Bank	<code>POST /api/v1/questions</code>	创建题目
Question Bank	<code>GET /api/v1/questions?tags=sql&amp;difficulty=hard</code>	题库查询

Exams	<code>POST /api/v1/exams</code>	创建考试，设定策略
Exams	<code>POST /api/v1/exams/{id}/publish</code>	发布考试
Exams	<code>GET /api/v1/exams/{id}/paper</code> (学生)	拉取试卷（含随机题）
Submission	<code>POST /api/v1/exams/{id}/submit</code>	提交答案
Submission	<code>GET /api/v1/submissions/{id}</code>	查看答卷与评分
Analytics	<code>GET /api/v1/analytics/exams/{id}</code>	考试分析
Notifications	<code>POST /api/v1/notifications</code>	发送系统通知

## 5.4 权限/角色

- **Admin**: 系统配置、全局管理、用户审批。
- **Teacher**: 课程、题库、考试、评分、通知。
- **Student**: 学习、考试、查看成绩。
- RBAC 表结构: `roles`, `permissions`, `role_permissions`, `user_roles`.

## 5.5 业务流程 (示例: 在线考试)

1. 教师创建考试 → 选择题库/策略 → 发布。
2. 学生在考试时间内访问 → 获取 JWT 验证 + 防切屏监控。
3. 前端定时保存答题进度；后端记录 `submission` 状态。
4. 考试结束自动收卷，触发异步评分任务。
5. 分数生成后通知学生，教师可查看统计与错题。

## 6. 部署与环境

### 6.1 开发环境

```
# 后端
python -m venv .venv
source .venv/bin/activate
pip install -r requirements/dev.txt
cp server/.env.example server/.env
alembic upgrade head
uvicorn app.main:app --reload
```

```
# 前端
cd web
pnpm install
pnpm dev
```

## 6.2 测试/生产环境

- 使用 Docker Compose:
  - `frontend` : Vite build → Nginx 静态托管。
  - `backend` : Gunicorn/Uvicorn workers (4 workers, async workers)。
  - `db` : PostgreSQL + 持久卷 + 自动备份。
  - `redis` : 缓存 & Celery broker。
  - `flower` : Celery 监控。
- 生产建议使用 Kubernetes 或云托管 (ACK/EKS/GKE)； Nginx Ingress + Cert-Manager 实现 HTTPS。
- 缓存配置、限流和 WAF 保护登录/考试接口。

## 6.3 配置项

- `.env` 示例：

```
APP_ENV=development
DATABASE_URL=postgresql+asyncpg://user:pass@db:5432/ustc
REDIS_URL=redis://redis:6379/0
JWT_SECRET=...
JWT_EXPIRE=900
REFRESH_EXPIRE=604800
STORAGE_ENDPOINT=https://oss.xxx
```

- 使用 `pydantic-settings` 管理配置，支持环境变量/文件。

## 7. 测试与质量保障

- 前端： Vitest + Vue Testing Library； Cypress 做端到端。
- 后端： Pytest + httpx/pytest-asyncio； 覆盖 auth/考试逻辑； 使用 test DB。
- CI/CD： GitHub Actions/GitLab CI
  - Lint (ESLint、 Stylelint、 mypy/ruff)
  - 单元测试
  - 构建镜像 & 推送
- 静态分析： 前端 TypeScript 严格模式，后端开启 `ruff`、`mypy`，`pre-commit`。
- 安全： SAST/Dependabot 监控依赖； JWT/密码加盐； SQL sandbox 使用隔离容器。

## 8. 运维与监控

- 日志：后端使用 `structlog / loguru` 输出 JSON；ElasticStack 或 Loki 收集。
- 监控：Prometheus + Grafana，暴露 FastAPI Metrics、DB 指标。
- 错误追踪：Sentry (前后端)。

- 备份：PostgreSQL 每日全量 + WAL；对象存储版本化。
- 灾备：只读副本、冷备份；考试期间建议双活部署。
- 审计：记录关键操作（改卷、加分、权限变更），满足课程管理审计需求。

## 9. 路线图

1. **MVP (Sprint 1-2)**：用户认证、课程管理、题库基本 CRUD、简单考试发布/提交。
2. **Sprint 3-4**：自动判题、成绩分析、消息通知、前端 UI 完善。
3. **Sprint 5+**：实时监考、编程题在线判题（Docker sandbox）、多租户支持、移动端适配。
4. **长期**：AI 辅助出题与推荐、学习轨迹个性分析、对接学校统一身份认证。

## 10. 附录

- **编码规范**：前端遵循 ESLint + Prettier；后端遵循 PEP8，建议以 `src/` 模式组织。
- **文档**：使用 `docs/` + VitePress/Docsify；OpenAPI 自动生成接口文档。
- **示例账号**：
  - 管理员：admin / Admin@123
  - 教师：teacher01 / Teacher@123
  - 学生：student01 / Student@123