自然写互动课堂教学管理云平台软件 V1.0

软件著作权鉴别材料（设计说明书）

项目	内容
软件全称	自然写互动课堂教学管理云平台软件
软件简称	自然写云平台
版本号	V1.0
权利人	深圳自然写科技有限公司
开发语言	Java / Python / JavaScript
文档类型	设计说明书
编制日期	2026年2月

目录

- 第一章 软件整体概述
- 1.1 软件简介与功能综述
- 1.2 软件用途与适用场景
- 1.3 运行环境与系统要求
- 1.4 开发语言与技术规范
- 1.5 版本说明
- 第二章 系统架构与设计思路
- 2.1 总体架构设计
- 2.2 各层次详细说明
- 2.3 核心模块架构图
- 2.4 数据设计
- 2.5 接口设计
- 2.6 安全设计
- 2.7 部署架构
- 第三章 核心模块功能详细说明
- 3.1 用户与权限管理模块
- 3.2 班级与课程管理模块

- 3.3 设备注册与绑定管理模块
- 3.4 笔迹数据接收与存储模块
- 3.5 作业与试卷管理模块
- 3.6 教学过程数据记录模块
- 3.7 多终端消息推送与同步模块
- 3.8 系统运维监控与日志管理模块
- 3.9 开放API接口模块
- 第四章 操作流程与使用步骤
- 4.1 系统安装与初始化
- 4.2 管理员登录与系统配置
- 4.3 用户管理操作流程
- 4.4 设备管理操作流程
- 4.5 课堂与作业管理操作流程
- 4.6 数据查询与报表导出流程
- 4.7 异常处理与故障排除
- 第五章 与源代码的对应关系
- 5.1 模块与源代码文件对应表
- 5.2 核心类与方法说明
- 5.3 命名规范
- 附录
- 附录A 术语表
- 附录B 版本历史

第一章 软件整体概述

1.1 软件简介与功能综述

自然写互动课堂教学管理云平台软件（以下简称"云平台"）是自然写互动课堂智能点阵笔系统的核心后台服务软件，承担整个系统的数据存储、业务调度、用户管理、设备管理及多端数据同步等核心职能。

云平台采用微服务架构设计，将业务能力拆分为独立的服务单元，每个服务单元单独部署、独立扩缩，保证了系统的高可用性和水平扩展能力。平台提供统一的RESTful API接口和WebSocket实时通道，供各端应用（手机端、PC端、电视端、智慧黑板端、平板端）调用，实现多端数据一致性。

云平台的主要功能模块包括以下几个方面：

第一，用户与权限管理。平台支持教师、学生、管理员、家长四类角色，基于RBAC（Role-Based Access Control）权限模型，精确控制各角色可访问的功能和数据范围。用户信息与学校组织架构绑定，支持批量导入、单独添加、禁用、删除等操作。

第二，班级与课程管理。教师可在平台创建班级，关联学校和年级信息，添加学生成员。课程模块支持按学科、学期规划课程表，课时记录与班级教学进度关联。

第三，设备注册与绑定管理。智能点阵笔、教室网关、智能算力盒、各终端设备均需在平台完成注册后方可使用。平台维护设备清单，记录设备MAC地址、序列号、当前绑定用户/班级、在线状态及最近活跃时间等信息。

第四，笔迹数据接收与存储。平台提供笔迹数据接收接口，接受来自网关或算力盒的实时笔迹数据流，写入消息队列后由AI引擎进行识别处理，识别结果与原始笔迹数据关联存储于数据库。

第五，作业与试卷管理。教师可在平台发布作业或试卷，设置截止时间、关联班级和点阵纸张页码。学生提交笔迹后，系统自动触发AI批改流程，批改结果可供教师复核和家长查看。

第六，多终端消息推送。平台集成消息推送服务，支持WebSocket实时推送和APNs/FCM系统级推送，保证各端在新作业发布、批改完成、系统通知等场景下的实时消息触达。

第七，系统运维监控。集成Prometheus + Grafana + ELK日志体系，实时监控各微服务健康状态、接口响应时间、错误率及服务器资源使用情况，支持告警配置与自动通知。

第八，开放API接口。平台对外提供经过鉴权的RESTful API，供SDK和第三方系统集成调用，支持标准OAuth 2.0授权流程。

1.2 软件用途与适用场景

云平台主要用于以下场景：

（1）学校互动课堂场景：为学校部署的自然写互动课堂提供云端支撑，管理全校班级、教师、学生和设备，处理课堂教学产生的海量笔迹数据。

（2）教育集团统一管理场景：教育集团或地区教育局可通过云平台统一管理旗下多所学校，实现跨校数据汇总、教学质量对比分析和设备资产统一管控。

（3）家校互动场景：家长通过手机端APP连接云平台，实时获取子女的学习状态、作业完成情况和学情诊断报告，与教师保持高效沟通。

（4）数据分析与决策支撑：学校管理者和教研人员可通过平台的统计报表功能获取教学数据洞察，辅助制定教学策略和评估教师绩效。

（5）第三方系统对接：教育机构可通过开放API将自然写的笔迹识别能力和学情分析能力集成到自身已有的校园信息系统（如教务系统、家校通系统）中。

1.3 运行环境与系统要求

服务端运行环境：

组件	要求
操作系统	Linux（CentOS 7.6+ / Ubuntu 20.04+）
容器平台	Docker 20.x+ / Kubernetes 1.24+
JDK版本	OpenJDK 17 / Oracle JDK 17
Python版本	Python 3.9+
数据库	MySQL 8.0+、MongoDB 6.0+、Redis 7.0+
消息队列	RabbitMQ 3.11+ / Kafka 3.4+
对象存储	阿里云OSS / MinIO（私有化部署）
最低服务器配置	8核CPU、16GB内存、200GB SSD（单节点最低配置）
推荐生产配置	16核CPU、64GB内存、1TB SSD（集群各节点）

网络要求：

- 服务端需要开放80、443、8080、8883（MQTT over TLS）等端口
- 客户端需能访问443端口（HTTPS）和WebSocket端口
- 生产环境建议配置CDN加速，确保全国各地访问延迟低于100ms

客户端浏览器要求（管理控制台）：

- Chrome 90+、Firefox 88+、Edge 88+、Safari 14+
- 支持JavaScript ES2017+、WebSocket协议

1.4 开发语言与技术规范

主要开发语言及框架：

服务模块	开发语言	主要框架
用户服务、作业服务、设备服务	Java 17	Spring Boot 3.x、Spring Security、MyBatis Plus
AI接入服务、数据处理服务	Python 3.9	FastAPI、SQLAlchemy、Pydantic
管理控制台前端	TypeScript	Vue.js 3、Element Plus、Axios

服务模块	开发语言	主要框架
API网关	Go 1.20	Kong / Nginx Lua脚本

代码规范：

- Java代码遵循《阿里巴巴Java开发手册》（华山版）
- Python代码遵循PEP 8规范，使用Black格式化工具
- TypeScript代码遵循ESLint + Prettier规范
- 所有代码通过Git版本管理，主干分支保护，合并需Code Review

接口规范：

- RESTful API遵循REST设计原则，URL使用小写单词，路径参数用{}标识
- 统一响应格式：`{"code": 200, "msg": "success", "data": {...}}`
- 错误码体系：2xx成功，4xx客户端错误，5xx服务端错误
- 接口文档使用OpenAPI 3.0规范，通过Swagger UI在线浏览

1.5 版本说明

版本号	发布日期	说明
V1.0	2026年2月	初始版本，包含核心教学管理功能

本鉴别材料对应版本为V1.0，软件功能覆盖用户管理、班级管理、设备管理、作业管理、笔迹数据接收、消息推送、运维监控等完整功能体系。

第二章 系统架构与设计思路

2.1 总体架构设计

云平台采用**微服务架构**，整体分为六个层次：接入层、服务层、消息层、缓存层、存储层、监控层。各层次职责明确，层间通过标准协议通信，可独立扩展和维护。

总体架构如下图所示：

```
┌──────────────────────────────────────────────────────────────────────┐
│                                接入层                                 │
│          Nginx（负载均衡/SSL终止）+ Kong API Gateway（限流/路由）        │
├──────────────────────┬──────────────────────────────────────────────┤
```

```
│                    │                 服务层（微服务集群）                    │
│  用户服务          │  课堂服务    作业服务    设备服务    消息服务          │
│  UserService       │  ClassSvc   AssignSvc  DeviceSvc  MessageSvc         │
├────────────────────┼──────────────────────────────────────────────────────┤
│                              消息层                                        │
│              RabbitMQ（业务事件）+ Kafka（笔迹数据流）                      │
├──────────────────────────────────────────────────────────────────────────┤
│                              缓存层                                        │
│              Redis Cluster（会话/热数据/实时状态缓存）                      │
├──────────────────────────────────────────────────────────────────────────┤
│                              存储层                                        │
│    MySQL（关系数据）+ MongoDB（笔迹/结果文档）+ OSS（文件对象存储）        │
├──────────────────────────────────────────────────────────────────────────┤
│                              监控层                                        │
│         Prometheus + Grafana（指标监控）+ ELK（日志采集分析）              │
└──────────────────────────────────────────────────────────────────────────┘
```

2.2 各层次详细说明

接入层设计：

接入层由Nginx和Kong API Gateway组成。Nginx负责SSL证书终止、HTTP/HTTPS协议转换和初级流量分发；Kong API Gateway承担API认证鉴权、流量限速、路由转发和插件扩展等职责。所有外部请求必须经过接入层校验后方可到达业务服务层，有效防止未授权访问和恶意流量冲击。

接入层配置了以下核心能力： – JWT令牌验证：所有需要身份认证的API均通过Kong的JWT插件进行令牌有效性校验 – 速率限制：针对敏感接口（如登录、注册）配置单IP请求频率上限，防止暴力破解 – 请求日志：记录所有入站请求的URL、来源IP、响应时间、状态码 – 跨域处理：配置CORS策略，允许指定来源域名的跨域请求

服务层设计：

服务层按业务领域划分为五个微服务，每个服务独立部署于Docker容器，通过Kubernetes编排管理：

- 用户服务（UserService）：负责用户注册、登录、角色授权、密码管理等
- 课堂服务（ClassroomService）：负责班级创建、学生管理、课程安排、课时记录
- 作业服务（AssignmentService）：负责作业发布、回收、状态跟踪、批改结果存储
- 设备服务（DeviceService）：负责设备注册、绑定、状态查询、固件版本管理
- 消息服务（MessageService）：负责站内消息、推送通知的创建、分发和状态追踪

各服务之间通过Feign HTTP客户端进行同步调用，通过消息队列进行异步解耦。

消息层设计：

消息层使用RabbitMQ和Kafka两套消息系统，各司其职：

RabbitMQ用于业务事件消息，如作业提交通知、批改完成通知、设备上线通知等，消息量相对较小但可靠性要求高，RabbitMQ的ACK机制确保消息不丢失。

Kafka用于高吞吐量的笔迹数据流，来自全校数十至数百个教室的网关上报的笔迹数据以高频率并发写入Kafka Topic，由AI引擎消费处理。Kafka的高吞吐、持久化和可回放特性满足了笔迹数据处理的需求。

缓存层设计：

Redis Cluster提供以下缓存功能： – 用户会话存储：JWT令牌黑名单、用户在线状态 – 热点数据缓存：班级列表、设备列表等频繁读取的数据 – 实时状态数据：设备在线状态、当前连接笔数量、课堂进行状态 – 分布式锁：防止并发请求产生数据竞争（如批量导入学生时）

存储层设计：

存储层根据数据特性选用不同的存储引擎：

MySQL存储关系型业务数据，包括用户表、班级表、设备表、作业表、成绩记录表等，利用MySQL的事务和外键约束保证数据一致性。

MongoDB存储半结构化的笔迹数据和AI识别结果，笔迹数据的结构因笔画数量而异，MongoDB的文档模型可灵活适应。

OSS对象存储用于存储课件文件、字帖图片、录像文件等二进制大文件，通过CDN加速对外分发。

2.3 核心模块架构图

用户认证流程架构：

```
客户端请求
     ↓
Kong API Gateway（JWT校验）
     ↓
UserService（登录接口）
     ↓
验证用户名/密码（MySQL查询 + BCrypt比对）
     ↓
生成JWT Access Token（有效期2小时）+ Refresh Token（有效期7天）
     ↓
Redis存储Refresh Token（支持主动注销）
     ↓
返回双令牌给客户端
```

作业批改数据流架构：

```
教师发布作业（AssignmentService）
     ↓
```

```
学生作答（纸上书写，点阵笔采集坐标）
    ↓
网关/算力盒上报笔迹数据（MQTT → Kafka）
    ↓
云平台Kafka消费者接收并匹配作业
    ↓
写入MongoDB（原始笔迹数据）
    ↓
发送AI识别请求至AI引擎服务
    ↓
AI引擎返回识别结果（文字/公式/评分）
    ↓
识别结果写入MongoDB（recognition_result集合）
    ↓
RabbitMQ发布"批改完成"事件
    ↓
消息服务推送通知至教师端和学生端
```

2.4 数据设计

核心数据表设计（MySQL）：

用户表（user）：

字段名	类型	长度	约束	说明
id	BIGINT	20	PK, AUTO_INCREMENT	用户唯一ID
username	VARCHAR	64	UNIQUE, NOT NULL	登录用户名
password_hash	VARCHAR	128	NOT NULL	BCrypt加密密码
real_name	VARCHAR	32	NOT NULL	真实姓名
role	TINYINT	1	NOT NULL	角色（1管理员/2教师/3学生/4家长）
phone	VARCHAR	20		手机号（加密存储）
school_id	BIGINT	20	FK	所属学校ID
class_id	BIGINT	20	FK	所属班级ID（学生专用）
status	TINYINT	1	DEFAULT 1	账号状态（1正常/0禁用）
created_at	DATETIME		NOT NULL	创建时间
updated_at	DATETIME		NOT NULL	最后更新时间

班级表（class）：

字段名	类型	长度	约束	说明
id	BIGINT	20	PK, AUTO_INCREMENT	班级唯一ID
name	VARCHAR	64	NOT NULL	班级名称（如"三年级一班"）
grade	VARCHAR	16	NOT NULL	年级（如"三年级"）
teacher_id	BIGINT	20	FK	主班教师ID
school_id	BIGINT	20	FK	所属学校ID
student_count	INT		DEFAULT 0	班级学生数量（冗余字段）
created_at	DATETIME		NOT NULL	创建时间

设备表（device）：

字段名	类型	长度	约束	说明
id	BIGINT	20	PK, AUTO_INCREMENT	设备唯一ID
device_type	TINYINT	1	NOT NULL	设备类型（1笔/2网关/3算力盒/4终端）
mac_addr	VARCHAR	32	UNIQUE, NOT NULL	设备MAC地址
serial_no	VARCHAR	64	UNIQUE	设备序列号
firmware_version	VARCHAR	32		当前固件版本
bind_user_id	BIGINT	20	FK	绑定用户ID（笔专用）
bind_class_id	BIGINT	20	FK	绑定班级ID（网关/算力盒）
last_online_at	DATETIME			最后在线时间
status	TINYINT	1	DEFAULT 0	在线状态（0离线/1在线）

作业表（assignment）：

字段名	类型	长度	约束	说明
id	BIGINT	20	PK, AUTO_INCREMENT	作业唯一ID
title	VARCHAR	128	NOT NULL	作业标题

字段名	类型	长度	约束	说明
type	TINYINT	1	NOT NULL	类型（1练习/2测验/3考试）
class_id	BIGINT	20	FK, NOT NULL	发布班级ID
teacher_id	BIGINT	20	FK, NOT NULL	发布教师ID
subject	VARCHAR	32		学科
deadline	DATETIME			截止时间
page_start	INT			对应点阵纸张起始页码
page_end	INT			对应点阵纸张结束页码
status	TINYINT	1	DEFAULT 1	状态（1进行中/2已截止/3已批改）
created_at	DATETIME		NOT NULL	发布时间

笔迹原始数据集合（MongoDB – stroke_data）：

```
{
  "_id": "ObjectId",
  "assignment_id": 12345,
  "student_id": 67890,
  "pen_id": "AA:BB:CC:DD:EE:FF",
  "page_id": 1001,
  "submit_time": "ISODate",
  "strokes": [
    {
      "stroke_index": 0,
      "points": [
        {"x": 1234, "y": 567, "pressure": 128, "ts": 1700000000123},
        {"x": 1235, "y": 568, "pressure": 130, "ts": 1700000000133}
      ]
    }
  ],
  "raw_size_bytes": 4096,
  "status": "pending_recognition"
}
```

AI识别结果集合（MongoDB – recognition_result）：

```
{
  "_id": "ObjectId",
  "stroke_id": "ObjectId（关联stroke_data）",
  "assignment_id": 12345,
  "student_id": 67890,
  "ocr_text": "春眠不觉晓",
  "confidence": 0.97,
  "math_result": null,
  "stroke_order_score": 85,
```

```
  "total_score": 90,
  "ai_feedback": "书写工整，笔顺正确，建议加强'觉'字竖钩收笔力度",
  "recognized_at": "ISODate",
  "model_version": "ocr_v2.1.0"
}
```

2.5 接口设计

统一接口规范：

所有API遵循以下规范： - 基础路径：`https://api.writech.com/api/v1/` - 认证方式：Bearer Token（JWT），在HTTP头部传递：`Authorization: Bearer <token>` - 内容类型：`Content-Type: application/json; charset=UTF-8` - 响应格式：`{"code": 200, "msg": "success", "data": {...}}` - 分页参数：`page`（页码，从1开始）、`size`（每页数量，默认20） - 时间格式：ISO 8601标准，如`2026-02-14T10:30:00+08:00`

核心API接口清单：

认证接口：

接口名称	方法	路径	权限要求	说明
用户登录	POST	/auth/login	无需认证	账号密码登录，返回双令牌
刷新令牌	POST	/auth/refresh	Refresh Token	使用刷新令牌换取新的访问令牌
退出登录	POST	/auth/logout	已登录用户	使当前令牌失效
修改密码	PUT	/auth/password	已登录用户	验证旧密码后更新密码

用户管理接口：

接口名称	方法	路径	权限要求	说明
创建用户	POST	/user	管理员	创建教师或管理员账号
批量导入学生	POST	/user/batch-import	教师/管理员	通过Excel批量创建学生账号
获取用户详情	GET	/user/{id}	管理员/本人	获取指定用户的详细信息
更新用户信息	PUT	/user/{id}	管理员/本人	更新用户基本信息
禁用/启用用户	PUT	/user/{id}/status	管理员	切换用户账号状态

设备管理接口：

接口名称	方法	路径	权限要求	说明
注册设备	POST	/device/register	管理员/教师	注册新设备（笔/网关/算力盒）
绑定设备	POST	/device/{id}/bind	管理员/教师	将设备绑定至用户或班级
设备列表	GET	/device	管理员/教师	分页查询设备列表，支持按类型筛选
设备状态	GET	/device/{id}/status	管理员/教师	查询设备当前在线状态和基本信息

作业管理接口：

接口名称	方法	路径	权限要求	说明
发布作业	POST	/assignment	教师	创建并发布作业任务
作业列表	GET	/assignment	教师/学生	分页查询作业列表
提交笔迹	POST	/stroke/upload	系统内部	接收网关上传的笔迹数据包
获取批改结果	GET	/result/{assignment_id}	教师/学生/家长	查询作业批改结果
学情报告	GET	/report/student/{id}	教师/家长/本学生	获取学生学情综合报告

WebSocket接口：

接口名称	路径	说明
实时消息通道	/ws/v1/notify	双向实时通信，推送新作业通知、批改完成通知等
课堂笔迹推送	/ws/v1/stroke	向大屏/教师端实时推送学生笔迹数据

2.6 安全设计

身份认证与授权：

系统采用JWT双令牌机制实现无状态身份认证： – Access Token（访问令牌）：有效期2小时，用于API请求鉴权 – Refresh Token（刷新令牌）：有效期7天，用于无感刷新Access Token – 用户主动退出时，将Refresh Token加入Redis黑名单，实现主动注销 – JWT签名算法采用RS256（RSA + SHA-256），私钥服务端保管，公钥分发至各微服务验证

权限控制体系：

基于RBAC模型定义四级权限： – 超级管理员：可管理平台级配置、创建学校账号 – 学校管理员：可管理本校所有教师、学生、设备和班级 – 教师：可管理本人负责班级的学生、作业、课件 – 学生/家长：仅可查看本人/子女相关数据

每个API接口在控制器层通过`@RequiresRole`注解配置允许访问的角色列表，Spring Security拦截器在运行时进行校验。

数据安全：

- 传输加密：全链路HTTPS，TLS 1.3协议，禁用不安全的TLS 1.0/1.1
- 存储加密：用户手机号、身份证号等隐私字段使用AES-256-GCM算法加密后存储
- 密码安全：用户密码使用BCrypt算法哈希存储，cost factor设为12
- SQL注入防护：使用MyBatis预编译语句，禁止拼接SQL字符串
- XSS防护：所有用户输入数据在返回前进行HTML转义

审计日志：

所有涉及数据变更的操作（用户创建/修改/删除、设备注册/绑定、作业发布/修改）均记录操作日志，包含操作人ID、操作类型、操作时间、客户端IP、变更前后数据快照。

2.7 部署架构

容器化部署方案：

生产环境采用Kubernetes集群部署，各微服务以Pod形式运行：

```
互联网用户
    ↓
CDN（阿里云CDN / 腾讯云CDN）
    ↓
SLB负载均衡（四层TCP负载均衡）
    ↓
Nginx Ingress（SSL终止 + 七层路由）
    ↓
Kong API Gateway Pod（认证 + 限流 + 路由）
    ↓
微服务Pod（UserService / ClassroomService / AssignmentService / DeviceService /
MessageService）
    ↓
数据存储（RDS MySQL / MongoDB Atlas / Redis Cluster / OSS）
```

多可用区高可用设计：

- 应用层：各微服务在两个可用区各部署至少2个Pod副本，Kubernetes在节点故障时自动重新调度
- 数据库层：MySQL采用主从复制模式，主库可用区A，从库可用区B；应用层读写分离，写操作走主库，读操作走从库
- MongoDB：采用副本集模式，3节点（1主2从），任一节点故障自动选举新主节点
- Redis：Cluster模式，6节点（3主3从），支持数据分片和主节点自动故障转移

弹性伸缩：

基于HPA（Horizontal Pod Autoscaler）配置自动扩缩： – 触发条件：CPU使用率 > 70% 或 内存使用率 > 80% – 扩缩范围：最小2副本，最大20副本 – 冷却时间：扩容后5分钟内不再触发扩容，缩容需连续10分钟低于阈值

第三章 核心模块功能详细说明

3.1 用户与权限管理模块

模块概述：

用户与权限管理模块（UserService）是整个云平台的基础模块，负责用户身份的全生命周期管理，包括账号创建、信息维护、角色分配、权限控制和账号注销。该模块为其他所有业务模块提供身份认证和权限校验服务。

功能详细说明：

（1）用户注册与创建

系统管理员可通过管理控制台手动创建教师和管理员账号。学生账号支持两种创建方式：管理员手动逐一创建，或通过标准Excel模板批量导入。批量导入时，系统对每行数据进行格式校验，校验失败的记录生成错误报告返回操作者，成功的记录写入数据库。

批量导入Excel模板字段包括：学号、姓名、性别、年级、班级、家长手机号等。系统自动为每个学生生成初始密码（默认为学号后6位），首次登录时强制要求修改密码。

（2）角色与权限管理

RBAC权限模型将用户角色分为4级，每个角色预定义了权限集合：

超级管理员角色拥有所有系统操作权限，包括创建学校账号、配置系统参数、查看全平台数据报表等。

学校管理员角色拥有本校范围内的所有管理权限，包括创建教师账号、注册设备、管理班级、查看全校学情数据。

教师角色拥有课堂教学相关权限，包括创建课程、发布作业、查看本班学情、与家长沟通。

学生和家长角色为只读权限，学生可查看自己的作业和批改结果，家长可查看子女的学情报告。

（3）用户信息管理

每个用户拥有基本信息档案，包括姓名、联系方式、学校和班级归属、绑定设备列表。教师可维护自己的个人信息；管理员可修改所有用户信息，但密码修改需要原密码验证（管理员重置密码除外）。

（4）账号安全管理

密码策略要求：最少8位，包含字母和数字，不得与近3次历史密码相同。连续5次登录失败后账号自动锁定30分钟，锁定期间提示用户等待或联系管理员解锁。

处理流程（用户登录）：

```
步骤1：客户端提交用户名和密码（HTTPS传输）
步骤2：Kong API Gateway转发请求至UserService登录接口
步骤3：UserService从MySQL查询用户记录
步骤4：检查用户账号状态（是否禁用、是否锁定）
步骤5：使用BCrypt.verify()比对密码哈希
步骤6：密码正确：生成JWT Access Token（RS256签名，有效期2小时）
步骤7：生成Refresh Token（随机UUID），写入Redis（KEY=rt:userId，TTL=7天）
步骤8：更新用户最后登录时间
步骤9：返回Access Token和Refresh Token
步骤10：客户端本地存储Token（移动端使用KeyStore/Keychain）
```

关键数据结构：

JWT Payload结构：

```
{
  "sub": "12345（用户ID）",
  "role": "teacher",
  "school_id": "100",
  "exp": 1700007200,
  "iat": 1700000000
}
```

对应源代码文件：

- `controller/AuthController.java`：登录、刷新令牌、退出登录接口实现
- `service/UserService.java`：用户业务逻辑，含密码验证、账号锁定逻辑
- `model/User.java`：用户实体类定义

3.2 班级与课程管理模块

模块概述：

班级与课程管理模块（ClassroomService）负责管理学校的组织架构，包括班级的创建和维护、学生成员管理、课程安排和课时记录。该模块是教学业务的基础数据支撑，作业发布、设备绑定等功能均依赖班级数据。

功能详细说明：

（1）班级管理

管理员可创建班级，设定年级、班主任、学科教师等属性。一个班级可以有多名任课教师，每位教师负责对应学科的教学内容。班级创建后，可批量导入或手动添加学生成员。

班级管理界面展示班级卡片列表，每张卡片显示班级名称、年级、当前学生人数、班主任姓名和班级状态。点击班级卡片进入班级详情页，可查看学生名单、任课教师列表、最近作业统计和学情概览。

（2）课程安排

教师可在课程管理模块为班级创建课程计划，设定每周课时安排。课程计划与日历视图联动，方便教师提前规划教学内容。每节课时结束后，教师可标记完成并添加课堂备注。

（3）学生成员管理

班级管理员（班主任）可对班级成员进行增减操作：添加转入学生、移除转出学生。学生调班时，历史作业和学情数据保留在原班级，新班级数据从调班后开始积累。

关键处理逻辑：

班级成员变更时的数据一致性处理：

```
1. 开启数据库事务
2. 更新user表中student的class_id字段
3. 更新class表的student_count字段（+1或-1）
4. 若设备绑定关系存在，更新设备绑定至新班级
5. 提交事务
6. 清除Redis中的班级信息缓存（KEY=class:${id}）
7. 发布班级成员变更事件至消息队列（通知相关终端刷新数据）
```

3.3 设备注册与绑定管理模块

模块概述：

设备管理模块（DeviceService）统一管理所有接入云平台的硬件设备，包括智能点阵笔、教室网关、智能算力盒和各类终端设备。设备在首次接入前需通过平台完成注册，注册成功后获得平台颁发的设备证书，用于后续通信认证。

功能详细说明：

（1）设备注册流程

设备注册分为两步：首先由管理员在管理控制台录入设备信息（序列号、MAC地址、设备类型、采购批次），系统生成设备档案并处于"未激活"状态。

设备首次上电联网时，向云平台的设备接入服务发送设备认证请求，携带序列号和出厂证书。平台验证序列号有效性和证书合法性后，将设备状态更新为"已激活"，同时向设备下发运行配置（MQTT服务器地址、心跳间隔、协议版本等）。

（2）设备绑定管理

设备激活后可绑定至具体的用户或班级：

- 智能点阵笔绑定至学生：每支笔通过MAC地址唯一标识，绑定学生后，该笔采集的所有笔迹数据自动关联至该学生账号
- 网关和算力盒绑定至班级或教室：一个教室通常配置1台网关和1台算力盒，绑定后所有经过该网关的笔迹数据自动关联至对应班级

（3）设备状态监控

设备服务实时汇聚各设备的心跳信息： – 在线状态：基于最后心跳时间判断，超过设定阈值（默认2分钟）未收到心跳则标记为离线 – 电量信息：点阵笔通过心跳上报当前电量百分比，服务端存储并展示 – 固件版本：设备在心跳包中携带当前固件版本号，便于管理员识别需要升级的设备

（4）OTA固件升级管理

管理员可通过设备管理界面发起批量OTA升级任务：选定目标设备范围（全部/按型号/按班级/指定设备）、上传固件包、设定升级时间窗口（建议在非教学时段执行）。系统通过MQTT向目标设备下发升级指令，设备完成升级后回报新版本号，管理界面实时更新升级进度。

对应源代码文件：

- `controller/DeviceController.java`：设备注册、绑定、状态查询接口
- `service/DeviceService.java`：设备业务逻辑，含注册验证、状态更新

3.4 笔迹数据接收与存储模块

模块概述：

笔迹数据接收模块是云平台处理海量教学数据的核心模块，负责接收来自全校各教室网关或算力盒上报的学生书写笔迹坐标数据，进行格式校验、数据关联和持久化存储，并触发后续的AI识别流程。

功能详细说明：

（1）笔迹数据接收

网关和算力盒通过MQTT协议将笔迹数据推送至云平台的消息代理（Kafka）。消息Topic格式为`pen/{gateway_id}/stroke`，消息体为Protobuf格式序列化的笔迹数据包，包含设备ID、学生ID、时间戳、坐标序列等字段。

云平台部署多个Kafka Consumer实例（根据负载自动扩缩），每个消费者从Kafka分区并行消费数据，处理速率可随负载动态调整。

（2）数据校验与关联

消费者接收到笔迹数据后执行以下处理： – 格式校验：验证Protobuf数据包结构完整性，拒绝损坏的数据包 – 设备鉴权：通过gateway_id查询设备注册信息，验证设备合法性 – 学生关联：通过笔的MAC地址查询绑定学生信息 – 作业匹配：根据笔迹对应的点阵纸张页码ID，在作业表中匹配对应的作业任务 – 去重处理：通过数据包序列号进行去重，防止网络重传导致数据重复

（3）数据存储

校验通过的笔迹数据写入MongoDB的stroke_data集合。写入时为每条记录生成全局唯一的stroke_id，用于后续与AI识别结果关联。写入MongoDB后，向AI引擎的Kafka Topic发送识别请求，携带stroke_id，由AI引擎异步进行OCR识别和批改。

数据流量估算：

单班40名学生同时书写时，假设平均每支笔每秒产生100个坐标点，每个坐标点7字节，则单班笔迹数据带宽为：40 × 100 × 7 = 28,000 字节/秒 ≈ 28 KB/s。全校100个班级同时上课时，总带宽约2.8 MB/s，Kafka集群完全可以承载。

3.5 作业与试卷管理模块

模块概述：

作业管理模块（AssignmentService）为教师提供作业全生命周期管理能力，包括作业创建、发布、状态跟踪、批改结果管理和学情数据汇总。

功能详细说明：

（1）作业创建与发布

教师通过PC端或手机端进入作业管理界面，创建作业时需填写以下信息： – 作业标题（必填） – 作业类型（练习/测验/考试，影响数据统计和展示方式） – 目标班级（可多选） – 学科（语文/数学/英语等） – 对应点阵纸张（选择预先设计的字帖或试卷模板，系统自动关联点阵码页码范围） – 截止时间

发布后，云平台向所有目标班级学生的终端设备推送"新作业通知"，学生打开应用即可看到新作业详情和对应的纸质作业要求。

（2）作业状态跟踪

作业发布后处于"进行中"状态，系统实时统计提交率（已收到笔迹的学生数量/班级总学生数量）。到达截止时间后，作业自动转为"已截止"状态，不再接收新的笔迹提交。

教师可在管理界面查看作业详情，包括每位学生的提交状态（未提交/已提交/已批改），点击学生姓名可查看该学生的笔迹回放和批改结果。

（3）批改流程管理

AI批改完成后，作业状态更新为"AI已批改"。教师可逐一查看AI批改结果，对有疑问的题目进行人工复核和修改评分。确认批改完成后，教师点击"发布批改结果"，系统向相关学生和家长推送批改完成通知。

对应源代码文件：

- `controller/AssignmentController.java`：作业发布、查询、状态更新接口
- `service/AssignmentService.java`：作业创建逻辑、AI批改触发、结果汇总
- `controller/StrokeController.java`：笔迹数据接收接口

3.6 教学过程数据记录模块

模块概述：

教学数据记录模块负责记录课堂教学过程中产生的各类行为数据，为后续学情分析提供原始数据支撑。

功能详细说明：

（1）课堂互动记录

课堂互动的每次操作（发题、收卷、随机抽取、展示学生作品）均记录操作类型、操作时间、参与学生列表和操作结果，构成课堂行为日志。

（2）学习行为数据

学生每次提交作业、获得批改结果后，系统更新该学生的学习行为画像，包括学科成绩趋势、书写质量变化、错题知识点分布等维度，这些数据作为学情诊断系统的输入数据。

（3）教师教学行为数据

系统记录教师的教学行为数据，包括作业发布频率、批改及时率、课堂互动次数等，用于教学质量评估和教研分析。

3.7 多终端消息推送与同步模块

模块概述：

消息推送模块（MessageService）负责向各类终端设备实时或延迟推送各类通知消息，确保系统内各角色能及时获知相关事件。

功能详细说明：

（1）消息类型

系统支持以下几类消息的推送： – 新作业通知：教师发布新作业后，推送给目标班级所有学生和家长 – 批改完成通知：作业批改完成后，推送给学生和家长，携带成绩摘要 – 设备告警通知：网关离线、设备电量过低等告警推送给管理员和教师 – 系统公告：平台维护通知、版本更新说明等全员推送 – 家校沟通消息：教师与家长之间的点对点消息通知

（2）推送渠道

根据终端类型选择合适的推送渠道： – 移动端（Android/iOS）：应用在前台时通过WebSocket实时推送；应用在后台时通过FCM（Android）或APNs（iOS）系统推送 – PC端：通过WebSocket长连接实时推送 – 大屏端（智慧黑板/电视）：通过WebSocket推送，大屏设备通常保持持久连接

（3）消息存储与状态追踪

每条消息写入数据库记录，包含消息ID、发送方、接收方、内容、发送时间、阅读状态。接收方读取消息后，客户端向服务端发送已读确认，服务端更新消息阅读状态。消息保留期限为6个月，过期自动归档清理。

3.8 系统运维监控与日志管理模块

模块概述：

运维监控模块基于Prometheus + Grafana + ELK技术栈，为平台运维团队提供系统健康状态可视化、性能指标监控和日志检索分析能力。

功能详细说明：

（1）指标监控（Prometheus + Grafana）

各微服务通过Spring Boot Actuator暴露Prometheus格式的指标数据，Prometheus定时抓取各服务指标。主要监控指标包括： – 系统级指标：CPU使用率、内存使用率、磁盘IO、网络流量 – JVM指标：堆内存使用、GC频率、线程池状态 – 业务指标：API请求量、响应时间P99/P95/P50、错误率 – 队列指标：Kafka消息积压量、RabbitMQ队列深度

Grafana配置告警规则，当关键指标超过阈值时（如错误率 > 5%），自动发送钉钉/邮件告警至运维人员。

（2）日志管理（ELK）

所有微服务通过Log4j2或Python logging输出结构化JSON日志，由Filebeat采集后发送至Logstash进行清洗和转换，最终存储至Elasticsearch并通过Kibana提供检索界面。

日志字段包括：服务名称、实例ID、请求ID（traceId，用于分布式链路追踪）、日志级别、时间戳、消息内容、异常堆栈等。

运维人员可在Kibana界面按时间范围、服务名称、日志级别、关键词等多维度检索日志，快速定位问题根因。

3.9 开放API接口模块

模块概述：

开放API模块为第三方系统和SDK提供标准化的数据访问接口，支持第三方教育软件将自然写的笔迹识别和学情分析能力集成到自身系统中。

功能详细说明：

（1）接入认证

第三方系统在接入前需向自然写申请AppKey和AppSecret，通过OAuth 2.0的Client Credentials模式获取访问令牌，令牌有效期24小时。

（2）接口能力

开放API提供以下核心能力： – 笔迹数据上传：第三方系统可通过API提交学生笔迹数据至自然写平台进行AI识别 – 识别结果查询：查询指定笔迹数据的OCR识别结果、数学识别结果 – 学情数据查询：查询学生或班级的学情统计数据（需用户授权） – Webhook推送：第三方系统注册Webhook回调地址，识别完成后平台主动推送结果

（3）配额管理

开放API按接入方配置调用配额，包括每日最大调用次数和每分钟最大并发数。超过配额的请求返回429状态码。平台提供配额用量统计和用量预警功能。

第四章 操作流程与使用步骤

4.1 系统安装与初始化

Docker Compose快速部署（开发/测试环境）：

```
步骤1：确保服务器已安装Docker 20.x+和Docker Compose 2.x+
步骤2：从代码仓库拉取部署配置文件
        git clone https://git.writech.com/deployment/cloud-platform.git
步骤3：进入部署目录，复制环境变量配置模板
        cp .env.example .env
步骤4：编辑.env文件，配置数据库密码、Redis密码、JWT密钥、OSS配置等
步骤5：执行数据库初始化脚本
        docker compose run --rm db-init
步骤6：启动所有服务
        docker compose up -d
步骤7：等待所有容器状态变为"healthy"（约2-3分钟）
步骤8：通过浏览器访问管理控制台：http://localhost:8080
        默认超级管理员账号：admin / Writech@2026
步骤9：首次登录后立即修改默认密码
```

Kubernetes生产部署（生产环境）：

```
步骤1：配置kubectl连接至目标K8s集群
步骤2：创建命名空间：kubectl create namespace writech-cloud
步骤3：创建ConfigMap和Secret资源（数据库连接、JWT密钥等配置）
步骤4：部署中间件（MySQL/MongoDB/Redis/Kafka），或配置云数据库连接信息
步骤5：按顺序部署各微服务Deployment：
        kubectl apply -f k8s/user-service.yaml
        kubectl apply -f k8s/classroom-service.yaml
        kubectl apply -f k8s/assignment-service.yaml
        kubectl apply -f k8s/device-service.yaml
        kubectl apply -f k8s/message-service.yaml
步骤6：部署API网关（Kong）和Nginx Ingress
步骤7：配置DNS解析和SSL证书
步骤8：验证所有Pod状态：kubectl get pods -n writech-cloud
步骤9：执行健康检查：curl https://api.writech.com/health
```

4.2 管理员登录与系统配置

管理员登录操作：

```
界面布局：
┌──────────────────────────────────────────────────────────┐
│                 自然写互动课堂管理平台                      │
│                                                          │
│          ┌──────────────────────────────┐                │
│          │  用户名：[输入框]              │                │
│          │  密　码：[密码输入框]          │                │
│          │  验证码：[输入框] [验证码图片]  │                │
│          │          [登录按钮]            │                │
│          └──────────────────────────────┘                │
│                    忘记密码？联系管理员                     │
└──────────────────────────────────────────────────────────┘
```

```
操作步骤：
1. 打开浏览器，输入管理平台URL
2. 在用户名输入框输入管理员账号
3. 在密码输入框输入密码（密码不回显，以●代替）
4. 输入图形验证码（登录失败2次后出现）
5. 点击"登录"按钮
6. 系统验证通过后跳转至管理控制台首页
```

系统初始配置步骤：

```
1. 学校信息配置
   路径：系统设置 → 学校管理 → 基本信息
   填写：学校名称、所在区域、联系人、联系方式
   上传：学校Logo（用于报告页眉）

2. 年级班级初始化
   路径：学校管理 → 班级管理 → 批量创建
   操作：按年级批量创建班级（支持"一至六年级，每年级6个班"等批量配置）

3. 教师账号创建
   路径：用户管理 → 教师管理 → 添加教师
   必填：姓名、工号、手机号、任教学科、负责班级

4. 设备导入
   路径：设备管理 → 设备导入 → 上传Excel
   操作：下载设备导入模板，填写设备序列号和MAC地址后上传
```

4.3 用户管理操作流程

批量导入学生操作流程：

```
步骤1：进入用户管理 → 学生管理页面
步骤2：点击右上角"批量导入"按钮
步骤3：点击"下载模板"获取标准Excel导入模板
步骤4：按模板格式填写学生信息：
        A列：学号（必填，同校唯一）
        B列：姓名（必填）
        C列：性别（必填：男/女）
        D列：班级（必填：与系统班级名称一致）
        E列：家长手机号（选填）
步骤5：保存Excel文件后，在导入界面上传该文件
步骤6：系统展示预览：校验通过的记录数和校验失败的记录（含失败原因）
步骤7：确认无误后点击"确认导入"，系统开始批量创建账号
步骤8：导入完成后，系统生成"导入报告"Excel，可下载查看每条记录处理结果
步骤9：对未能自动创建的记录（如姓名重复等），手动逐一处理
```

4.4 设备管理操作流程

智能点阵笔注册与绑定流程：

```
步骤1：管理员进入设备管理 → 笔设备管理页面
步骤2：点击"批量注册"，上传设备清单Excel（含序列号和MAC地址）
步骤3：系统创建设备档案，状态为"待激活"
步骤4：将笔交给对应学生，学生首次开机联网后，系统自动将设备状态更新为"已激活"
步骤5：管理员在设备列表中选中已激活的笔，点击"绑定学生"
步骤6：在弹出的学生选择框中搜索学生姓名或学号，选择并确认
步骤7：绑定成功后，该笔采集的所有后续数据将自动关联至该学生账号

界面示例：
┌──────────────────────────────────────────────────────────────────┐
│  设备管理 > 笔设备管理                    [批量注册][批量导出]  │
├──────┬──────────────────┬──────────┬────────┬──────────┬──────────┤
│ 序号 │ 序列号           │ MAC地址  │ 状态   │ 绑定学生 │ 操作     │
├──────┼──────────────────┼──────────┼────────┼──────────┼──────────┤
│  1   │ PEN2026001       │ AA:BB:01 │ 在线   │ 张三     │ 绑定/解绑│
│  2   │ PEN2026002       │ AA:BB:02 │ 离线   │ 李四     │ 绑定/解绑│
│  3   │ PEN2026003       │ AA:BB:03 │ 待激活 │ 未绑定   │ 绑定     │
└──────┴──────────────────┴──────────┴────────┴──────────┴──────────┘
```

4.5 课堂与作业管理操作流程

教师发布作业操作流程：

```
步骤1：教师在PC端或手机端应用中，进入"作业管理"功能
步骤2：点击"发布新作业"按钮，进入作业创建表单

作业创建表单界面：
┌──────────────────────────────────────────────────────────────────┐
│  发布新作业                                                       │
│  作业标题：[第三单元生字练习_____________]                        │
│  学科：     [语文 ▼]                                              │
│  作业类型：[○练习 ●测验 ○考试]                                   │
│  目标班级：[☑三年级一班] [☑三年级二班] [□三年级三班]             │
│  作业内容：请完成字帖第15页所有生字的书写练习                     │
│  对应纸张：[字帖2026版-第15页 ▼]  点阵码：P20260015              │
│  截止时间：[2026-03-01 18:00:00]                                  │
│                                  [取消]  [保存草稿]  [立即发布]  │
└──────────────────────────────────────────────────────────────────┘

步骤3：填写完毕后点击"立即发布"
步骤4：系统显示"发布成功，已通知X名学生"的确认提示
步骤5：在作业列表中可查看新发布作业的实时提交进度
```

4.6 数据查询与报表导出流程

学情报告查询操作：

```
步骤1：进入"学情分析"功能模块
步骤2：选择查询维度：
```

```
        - 个人报告：选择班级 → 选择学生 → 选择时间范围
        - 班级报告：选择班级 → 选择时间范围
步骤3：系统加载并展示报告数据（约2-5秒）

报告页面布局：
┌──────────────────────────────────────────────────────────┐
│  学生学情报告 | 张三 | 三年级一班 | 2026年1月                  │
├──────────────┬───────────────────────────────────────────┤
│  总体得分     │ 书写质量趋势图（折线图）                        │
│  88.5 分     │                                            │
├──────────────┴───────────────────────────────────────────┤
│  知识点掌握雷达图  │  错题分布柱状图                          │
│  （语文/数学各维度）│   （按知识点分类统计错误次数）              │
├──────────────────────────────────────────────────────────┤
│  近期作业列表（标题/得分/提交时间/批改状态）                    │
└──────────────────────────────────────────────────────────┘

步骤4：点击"导出报告"，选择导出格式（PDF/Excel）
步骤5：系统后台生成报告文件，完成后弹出下载提示
```

4.7 异常处理与故障排除

常见异常及处理方法：

异常现象	可能原因	处理方法
登录失败，提示"用户名或密码错误"	密码输入有误或账号不存在	检查账号拼写，必要时联系管理员重置密码
登录失败，提示"账号已锁定"	连续失败超过5次	等待30分钟后重试，或联系管理员手动解锁
设备显示"离线"但实际已开机	网络连接异常或心跳超时	检查设备网络，重启设备后等待约1分钟
作业提交率异常，部分学生未收到	推送失败或学生未安装应用	检查消息推送日志，引导学生手动刷新作业列表
批改结果长时间未出现	AI引擎处理队列拥堵	检查AI引擎服务状态，查看Kafka消息积压指标
系统响应缓慢	负载过高或数据库性能问题	查看Grafana监控面板，检查慢查询日志

系统日志查看方法：

```
运维人员通过Kibana查询系统日志：
1. 打开Kibana管理界面：https://kibana.writech.com
2. 进入Discover功能模块
3. 选择索引模式：writech-cloud-logs-*
```

```
4. 设置时间范围（右上角时间选择器）
5. 在搜索框输入过滤条件，如：
   - 按服务过滤：service.name: "user-service"
   - 按级别过滤：log.level: "ERROR"
   - 按请求ID：trace_id: "abc123def456"
6. 点击刷新按钮获取最新日志
```

第五章 与源代码的对应关系

5.1 模块名称与源代码文件对应表

功能模块	包/目录路径	主要源文件	说明
应用启动入口	根目录	`WritechCloudApplication.java`	Spring Boot应用主类，包含@SpringBootApplication注解
用户认证模块	`controller/`	`AuthController.java`	登录、刷新令牌、退出登录接口控制器
用户管理模块	`controller/`	–	（内嵌于AuthController或UserController）
设备管理模块	`controller/`	`DeviceController.java`	设备注册、绑定、状态查询、OTA管理接口
作业管理模块	`controller/`	`AssignmentController.java`	作业发布、查询、批改结果管理接口
笔迹数据接口	`controller/`	`StrokeController.java`	笔迹数据上传和查询接口
用户业务逻辑	`service/`	（UserService相关文件）	用户CRUD、密码加密验证业务逻辑
作业业务	`service/`	（AssignmentService相关文件）	作业状态管理、AI批改触发逻辑

功能模块	包/目录路径	主要源文件	说明
逻辑			
数据实体模型	`model/`	（User.java等Model类文件）	JPA/MyBatis数据实体定义
系统配置	`config/`	（SecurityConfig.java等配置类）	Spring Security配置、Redis配置、MQ配置

5.2 核心类与方法说明

WritechCloudApplication.java：

```
// 应用程序主入口类
@SpringBootApplication
@EnableDiscoveryClient        // 启用服务发现（Nacos）
@EnableFeignClients           // 启用Feign远程调用
public class WritechCloudApplication {
    public static void main(String[] args) {
        SpringApplication.run(WritechCloudApplication.class, args);
    }
}
```

AuthController.java 核心方法：

方法名	HTTP方法	路径	功能说明
`login(LoginRequest req)`	POST	`/api/v1/auth/login`	处理用户登录，验证密码并签发JWT令牌
`refreshToken(String refreshToken)`	POST	`/api/v1/auth/refresh`	使用刷新令牌换取新的访问令牌
`logout(String userId)`	POST	`/api/v1/auth/logout`	将当前刷新令牌加入Redis黑名单
`changePassword(ChangePasswordReq req)`	PUT	`/api/v1/auth/password`	验证旧密码后更新密码哈希

DeviceController.java 核心方法：

方法名	HTTP方法	路径	功能说明
`registerDevice(DeviceRegisterReq req)`	POST	`/api/v1/device/register`	注册新设备，生成设备档案
`bindDevice(Long deviceId, BindReq req)`	POST	`/api/v1/device/{id}/bind`	将设备绑定至用户或班级
`listDevices(DeviceQueryReq req)`	GET	`/api/v1/device`	分页查询设备列表
`getDeviceStatus(Long deviceId)`	GET	`/api/v1/device/{id}/status`	查询设备实时状态
`triggerOtaUpdate(OtaRequest req)`	POST	`/api/v1/device/ota`	发起OTA固件升级任务

AssignmentController.java 核心方法：

方法名	HTTP方法	路径	功能说明
`publishAssignment(AssignmentReq req)`	POST	`/api/v1/assignment`	发布新作业，触发消息推送
`getAssignmentList(AssignmentQueryReq req)`	GET	`/api/v1/assignment`	分页查询作业列表
`getResult(Long assignmentId, Long studentId)`	GET	`/api/v1/result/{assignment_id}`	获取指定作业的批改结果
`getStudentReport(Long studentId)`	GET	`/api/v1/report/student/{id}`	获取学生

方法名	HTTP方法	路径	功能说明
			综合学情报告

5.3 命名规范

包命名规范：

```
com.writech.cloud.{module}.{layer}
示例：
com.writech.cloud.user.controller    // 用户模块控制器层
com.writech.cloud.user.service       // 用户模块服务层
com.writech.cloud.user.model         // 用户模块数据模型层
com.writech.cloud.device.controller  // 设备模块控制器层
```

类命名规范：

类型	命名规则	示例
Controller类	XxxController	AuthController, DeviceController
Service接口	XxxService	UserService, AssignmentService
Service实现	XxxServiceImpl	UserServiceImpl, AssignmentServiceImpl
数据实体类	直接用业务名	User, Device, Assignment
请求体类	XxxRequest / XxxReq	LoginRequest, DeviceRegisterReq
响应体类	XxxResponse / XxxResp	LoginResponse, DeviceStatusResp
枚举类	XxxEnum	RoleEnum, DeviceTypeEnum
配置类	XxxConfig	SecurityConfig, RedisConfig

数据库命名规范：

- 表名：全小写，下划线分隔，如 `user`、`class`、`device`、`assignment`
- 字段名：全小写，下划线分隔，如 `class_id`、`created_at`、`firmware_version`
- 主键：统一命名为 `id`，BIGINT类型，自增
- 时间字段：创建时间命名为 `created_at`，更新时间命名为 `updated_at`

附录

附录A 界面设计稿（GUI Mockup）

本附录提供自然写互动课堂教学管理云平台软件各主要管理后台界面的设计稿，以线框图形式呈现界面布局与交互元素。

A.1 登录页面

```
┌──────────────────────────────────────────────────────────────┐
│                                                              │
│                  ┌──────────────────────┐                    │
│                  │   自然写管理平台      │                    │
│                  │  Writech Cloud Console │                   │
│                  └──────────────────────┘                    │
│                                                              │
│                  ┌──────────────────────┐                    │
│                  │ 账号  [_______________] │                  │
│                  │ 密码  [_______________] │                  │
│                  │       [  验证码  ][图]  │                  │
│                  │                        │                  │
│                  │ □ 记住我    忘记密码?   │                  │
│                  │   ┌──────────────┐     │                  │
│                  │   │   立 即 登 录  │     │                  │
│                  │   └──────────────┘     │                  │
│                  └──────────────────────┘                    │
│                                                              │
│          © 2026 深圳自然写科技有限公司   版本 V1.0            │
└──────────────────────────────────────────────────────────────┘
```

A.2 系统主控台（Dashboard）

```
┌──────────────────────────────────────────────────────────────────────────┐
│ 自然写管理平台   [搜索框_____________🔍]        👤 管理员 ▼ 🔔(3)  [退出] │
├──────────────────────────────────────────────────────────────────────────┤
│ ┌──────────┐ ┌──────────────────────────────────────────────────────┐   │
│ │ 📊 控制台  │ │                      系 统 概 览                      │   │
│ │ 🏢 租户管理 │ │  ┌──────────┐ ┌──────────┐ ┌────────┐ ┌────────┐   │   │
│ │ 👥 用户管理 │ │  │ 租户总数  │ │ 在线设备  │ │今日课堂 │ │今日作业 │   │   │
│ │ 📚 课堂管理 │ │  │  1,286   │ │  4,821   │ │  3,044 │ │  8,721 │   │   │
│ │ 📝 作业管理 │ │  │ ↑12 本月  │ │ ↑88 在线  │ │ 进行中  │ │ 待批改  │   │   │
│ │ 📱 设备管理 │ │  └──────────┘ └──────────┘ └────────┘ └────────┘   │   │
│ │ 🤖 AI识别  │ │                                                      │   │
│ │ 📈 数据报表 │ │  📈 过去7日课堂数量趋势                               │   │
│ │ ⚙️ 系统设置 │ │  3500┤                          ●                   │   │
│ │          │ │  3000┤            ●     ●    ●     ●    ●             │   │
```

```
│ │              │ │  2500├     ●    ●                                │ │
│ │              │ │  2000├                                            │ │
│ │              │ │      └──────┴──────┴──────┴──────┴──────┴──────┘ │ │
│ │              │ │         周一   周二   周三   周四   周五   周六    │ │
│ └──────────────┘ └──────────────────────────────────────────────────┘ │
└────────────────────────────────────────────────────────────────────────┘
```

A.3 租户管理页面

```
┌──────────────────────────────────────────────────────────────────────────┐
│ 🏢 租户管理                                                              │
├──────────────────────────────────────────────────────────────────────────┤
│  [搜索租户名称_____________🔍]  状态▼全部  类型▼全部    [+ 新增租户]  [导出Excel]  │
├──────┬──────────────┬──────┬──────┬──────────┬──────────┬──────────────┤
│ #    │ 租户名称     │ 类型 │ 状态 │ 设备数   │ 用户数   │ 到期时间     │ 操作         │
├──────┼──────────────┼──────┼──────┼──────────┼──────────┼──────────────┤
│ 1    │ 华南师范大学附中│ 学校 │ ●启用 │ 128      │ 3,240    │ 2027-01-01│[详情][编辑][禁用]│
│ 2    │ 广州市越秀区小学│ 学校 │ ●启用 │ 64       │ 1,890    │ 2026-12-31│[详情][编辑][禁用]│
│ 3    │ 深圳龙华区中学 │ 学校 │ ○停用 │ 32       │ 820      │ 2025-06-30│[详情][编辑][启用]│
│ 4    │ 东莞实验小学   │ 学校 │ ●启用 │ 96       │ 2,150    │ 2027-03-15│[详情][编辑][禁用]│
│ 5    │ 佛山南海区中学 │ 学校 │ ●启用 │ 48       │ 1,200    │ 2026-09-01│[详情][编辑][禁用]│
├──────┴──────────────┴──────┴──────┴──────────┴──────────┴──────────────┤
│  共 1,286 条记录    < 1  2  3  ...  43  >   每页显示 [30▼]               │
└──────────────────────────────────────────────────────────────────────────┘
```

A.4 课堂管理页面

```
┌──────────────────────────────────────────────────────────────────────────┐
│ 📚 课堂管理                                              [查看进行中课堂] │
├──────────────────────────────────────────────────────────────────────────┤
│  租户▼  班级▼  日期[2026-02-14]至[2026-02-28]  状态▼  [🔍搜索]  [导出报表]  │
├──────────┬──────┬──────┬──────┬──────────┬──────────┬──────────┬──────┤
│ 课堂ID   │ 班级 │ 教师 │ 学生数│ 开始时间 │ 结束时间 │ 互动次数 │ 操作 │
├──────────┼──────┼──────┼──────┼──────────┼──────────┼──────────┼──────┤
│ CLS-8821 │ 高一(3)班│ 李明 │ 45   │ 08:00    │ 08:45    │ 286      │[详情][回放]│
│ CLS-8820 │ 初三(2)班│ 王芳 │ 42   │ 08:00    │ 08:40    │ 312      │[详情][回放]│
│ CLS-8819 │ 小学六年级│ 张华 │ 38   │ 07:55    │ 08:35    │ 198      │[详情][回放]│
│ CLS-8818 │ 高二(1)班│ 陈静 │ 46   │ 07:50    │ 08:30    │ 425      │[详情][回放]│
├──────────┴──────┴──────┴──────┴──────────┴──────────┴──────────┴──────┤
│  共 3,044 条今日记录    < 1  2  3 ...  >                                 │
└──────────────────────────────────────────────────────────────────────────┘
```

A.5 设备管理页面

```
┌──────────────────────────────────────────────────────────────────────────┐
│ 📱 设备管理                                                              │
├──────────────────────────┬───────────────────────────────────────────────┤
```

```
│ 筛选条件                  │ 设备列表                                                    │
│ ──────────────────        │ [搜索SN/设备名___🔍]  类型▼  状态▼  [+ 批量导入]  [导出]  │
│ 设备类型                  │ ┌──┬──────────┬────┬────┬──────────┬──────────┐          │
│ ☑ 点阵笔                  │ │ # │ 设备SN    │ 类型 │状态 │ 所属租户 │ 固件版本 │          │
│ ☑ 网关                    │ ├──┼──────────┼────┼────┼──────────┼──────────┤          │
│ ☑ 算力盒                  │ │ 1 │PEN-001234│点阵笔 │●在线│华南师范附中│ V2.3.1  │          │
│                           │ │ 2 │PEN-001235│点阵笔 │●在线│华南师范附中│ V2.3.1  │          │
│ 状态                      │ │ 3 │GW-000321 │网 关  │●在线│广州越秀小学│ V1.8.0  │          │
│ ☑ 在线                    │ │ 4 │BOX-000045│算力盒 │○离线│深圳龙华中学│ V1.5.2  │          │
│ ☑ 离线                    │ │ 5 │PEN-001236│点阵笔 │●在线│东莞实验小学│ V2.3.0  │          │
│ ☐ 故障                    │ │   │          │      │    │          │ [OTA升级] │          │
│                           │ └──┴──────────┴────┴────┴──────────┴──────────┘          │
│ [重置]  [应用筛选]        │ 共 4,821 台设备（4,133在线 / 688离线）                    │
└──────────────────────────┴──────────────────────────────────────────────────────────┘
```

A.6 AI批改任务监控页面

```
┌──────────────────────────────────────────────────────────────────────────────────┐
│ 🤖 AI识别与批改监控                                    [刷新] 自动刷新 [30s▼]    │
├──────────────────────────────────────────────────────────────────────────────────┤
│ 今日统计                                                                          │
│ ┌──────────┐ ┌──────────┐ ┌──────────┐ ┌──────────┐                              │
│ │ 总识别次数 │ │ 平均耗时  │ │ 成功率   │ │ 队列积压 │                              │
│ │ 128,432  │ │ 1.23s    │ │ 99.7%    │ │ 12       │                              │
│ └──────────┘ └──────────┘ └──────────┘ └──────────┘                              │
│                                                                                   │
│ 任务队列（实时）                                                                  │
│ ┌──────────┬──────────┬──────────┬──────────┬──────────┬──────────┐              │
│ │ 任务ID   │ 类型     │ 提交时间 │ 耗时     │ 状态     │ 租户     │              │
│ ├──────────┼──────────┼──────────┼──────────┼──────────┼──────────┤              │
│ │TASK-9982133│手写汉字识别│08:42:31│ 0.8s   │✅完成    │华南师附中│              │
│ │TASK-9982132│数学列式识别│08:42:30│ 1.2s   │✅完成    │广州越秀小│              │
│ │TASK-9982131│英文手写识别│08:42:28│ 0.9s   │✅完成    │东莞实验小│              │
│ │TASK-9982130│字笔顺识别 │08:42:25│ 2.1s   │⏳处理中  │深圳龙华中│              │
│ └──────────┴──────────┴──────────┴──────────┴──────────┴──────────┘              │
└──────────────────────────────────────────────────────────────────────────────────┘
```

附录B 术语表

术语	说明
点阵笔	使用CMOS摄像头识别专用点阵纸张上的微型点阵码，实时解算书写坐标的智能书写工具
点阵纸张	印有肉眼不可见的微型点阵码的专用纸张，配合点阵笔使用，每个位置具有唯一坐标标识
网关	教室内的网络硬件设备，负责接收点阵笔的蓝牙数据并转发至云平台或算力盒
算力盒	教室内的边缘计算设备，在本地完成笔迹AI推理，降低云端延迟

术语	说明
笔迹数据	点阵笔采集的书写坐标序列，包含X坐标、Y坐标、压力值和时间戳
OCR识别	光学字符识别，此处特指对手写笔迹坐标序列进行文字识别的过程
RBAC	基于角色的访问控制（Role-Based Access Control），通过角色分配权限
JWT	JSON Web Token，一种基于JSON的开放标准，用于在网络应用间传递声明
MQTT	消息队列遥测传输协议，轻量级发布/订阅协议，适合IoT设备通信
OTA	空中升级（Over-The-Air），通过无线网络远程更新设备固件
Kafka	分布式流式处理平台，用于高吞吐量的消息存储和传输
BCrypt	密码哈希函数，专门为密码存储设计，包含盐值和成本因子

附录B 版本历史

版本号	发布日期	变更说明	变更人
V1.0	2026年2月14日	初始版本发布，包含核心教学管理功能模块	开发团队

编制单位：深圳自然写科技有限公司
文档版本：V1.0
编制日期：2026年2月
版权声明：本文档版权归深圳自然写科技有限公司所有，未经授权不得复制或传播

附录C 核心技术模块详细说明

C.1 Spring Cloud 微服务架构详解

C.1.1 服务注册与发现

云平台采用 Nacos 作为服务注册与配置中心，所有微服务实例在启动时自动注册：

```
# application.yml（公共配置）
spring:
  cloud:
    nacos:
      discovery:
        server-addr: nacos-cluster:8848
```

```
        namespace: production
        group: WRITECH_CLOUD
        metadata:
          version: 1.0.0
          region: cn-shenzhen
      config:
        server-addr: nacos-cluster:8848
        file-extension: yaml
        refresh-enabled: true
```

服务拓扑（生产环境）：

```
Nacos 服务注册中心
         │  服务注册/发现
         ├── auth-service（认证服务）           × 2实例
         ├── user-service（用户管理服务）        × 2实例
         ├── classroom-service（课堂管理服务）   × 3实例
         ├── assignment-service（作业管理服务）  × 3实例
         ├── device-service（设备管理服务）      × 2实例
         ├── analytics-service（学情分析服务）   × 2实例
         ├── notification-service（通知服务）    × 2实例
         └── file-service（文件存储服务）         × 2实例
```

C.1.2 API 网关核心配置

```yaml
# gateway-service/application.yml
spring:
  cloud:
    gateway:
      routes:
        - id: auth-route
          uri: lb://auth-service
          predicates:
            - Path=/api/v1/auth/**
          filters:
            - name: RequestRateLimiter
              args:
                redis-rate-limiter.replenishRate: 100
                redis-rate-limiter.burstCapacity: 200

        - id: classroom-route
          uri: lb://classroom-service
          predicates:
            - Path=/api/v1/classroom/**
          filters:
            - AuthFilter  # 自定义鉴权过滤器
            - name: CircuitBreaker
              args:
                name: classroomCircuitBreaker
                fallbackUri: forward:/fallback/classroom

        - id: assignment-route
          uri: lb://assignment-service
          predicates:
```

```
            - Path=/api/v1/assignment/**
          filters:
            - AuthFilter
            - name: Retry
              args:
                retries: 3
                statuses: BAD_GATEWAY, SERVICE_UNAVAILABLE
```

C.1.3 JWT 认证过滤器实现

```
// AuthFilter.java
@Component
public class AuthFilter implements GlobalFilter, Ordered {

    private static final String TOKEN_HEADER = "Authorization";
    private static final String TOKEN_PREFIX = "Bearer ";

    @Autowired
    private JwtTokenProvider jwtTokenProvider;

    // 无需鉴权的路径白名单
    private static final Set<String> WHITE_LIST = Set.of(
        "/api/v1/auth/login",
        "/api/v1/auth/register",
        "/api/v1/auth/refresh",
        "/api/v1/public/**"
    );

    @Override
    public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {
        String path = exchange.getRequest().getPath().value();

        // 白名单路径直接放行
        if (isWhiteListed(path)) {
            return chain.filter(exchange);
        }

        // 提取并验证 JWT Token
        String token = extractToken(exchange.getRequest());
        if (token == null) {
            return unauthorized(exchange, "缺少认证Token");
        }

        try {
            Claims claims = jwtTokenProvider.validateToken(token);
            // 将用户信息注入请求头，传递给下游服务
            ServerHttpRequest mutatedRequest = exchange.getRequest().mutate()
                .header("X-User-Id", claims.getSubject())
                .header("X-User-Role", claims.get("role", String.class))
                .header("X-School-Id", claims.get("schoolId", String.class))
                .build();
            return chain.filter(exchange.mutate().request(mutatedRequest).build());
        } catch (ExpiredJwtException e) {
            return unauthorized(exchange, "Token已过期");
        } catch (JwtException e) {
            return unauthorized(exchange, "Token无效");
```

```
        }
    }

    private String extractToken(ServerHttpRequest request) {
        String header = request.getHeaders().getFirst(TOKEN_HEADER);
        if (header != null && header.startsWith(TOKEN_PREFIX)) {
            return header.substring(TOKEN_PREFIX.length());
        }
        return null;
    }

    private Mono<Void> unauthorized(ServerWebExchange exchange, String message) {
        exchange.getResponse().setStatusCode(HttpStatus.UNAUTHORIZED);
        exchange.getResponse().getHeaders().setContentType(MediaType.APPLICATION_JSON);
        byte[] body = ("{\"code\":401,\"message\":\"" + message + "\"}").getBytes();
        DataBuffer buffer = exchange.getResponse().bufferFactory().wrap(body);
        return exchange.getResponse().writeWith(Mono.just(buffer));
    }

    @Override
    public int getOrder() { return -100; }
}
```

C.2 数据库设计详细说明

C.2.1 用户与权限表设计

```
-- 用户表
CREATE TABLE users (
    id VARCHAR(32) PRIMARY KEY COMMENT '用户ID（UUID）',
    username VARCHAR(64) NOT NULL UNIQUE COMMENT '登录用户名',
    password_hash VARCHAR(128) NOT NULL COMMENT 'BCrypt密码哈希',
    real_name VARCHAR(32) COMMENT '真实姓名',
    role TINYINT NOT NULL COMMENT '角色（1=管理员 2=教师 3=学生 4=家长）',
    school_id VARCHAR(32) COMMENT '所属学校ID',
    class_id VARCHAR(32) COMMENT '所属班级ID（学生/教师）',
    student_id VARCHAR(32) COMMENT '学号（学生角色）',
    phone VARCHAR(16) COMMENT '手机号',
    email VARCHAR(64) COMMENT '邮箱',
    status TINYINT DEFAULT 1 COMMENT '状态（0=禁用 1=正常）',
    last_login_at DATETIME COMMENT '最后登录时间',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    INDEX idx_school_role (school_id, role),
    INDEX idx_class (class_id),
    INDEX idx_phone (phone)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='用户表';

-- 设备表
CREATE TABLE devices (
    id VARCHAR(32) PRIMARY KEY COMMENT '设备ID（UUID）',
    serial_number VARCHAR(64) NOT NULL UNIQUE COMMENT '设备序列号',
    device_type TINYINT NOT NULL COMMENT '设备类型（1=点阵笔 2=网关 3=算力盒 4=黑板 5=PC）',
```

```
    device_name VARCHAR(64) COMMENT '设备名称',
    school_id VARCHAR(32) NOT NULL COMMENT '所属学校',
    classroom_id VARCHAR(32) COMMENT '所在教室',
    bound_user_id VARCHAR(32) COMMENT '绑定用户ID（点阵笔）',
    firmware_version VARCHAR(32) COMMENT '固件版本',
    last_online_at DATETIME COMMENT '最后在线时间',
    status TINYINT DEFAULT 1 COMMENT '状态（0=停用 1=在线 2=离线 3=故障）',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_school_type (school_id, device_type),
    INDEX idx_serial (serial_number)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='设备管理表';
```

C.2.2 课堂与作业表设计

```
-- 课堂会话表
CREATE TABLE classroom_sessions (
    id VARCHAR(32) PRIMARY KEY COMMENT '课堂会话ID',
    teacher_id VARCHAR(32) NOT NULL COMMENT '主讲教师ID',
    class_id VARCHAR(32) NOT NULL COMMENT '班级ID',
    subject VARCHAR(16) COMMENT '学科（chinese/math/english等）',
    session_name VARCHAR(128) COMMENT '课堂名称',
    started_at DATETIME COMMENT '开始时间',
    ended_at DATETIME COMMENT '结束时间',
    student_count INT DEFAULT 0 COMMENT '参与学生数',
    recording_url VARCHAR(256) COMMENT '课堂录像URL',
    status TINYINT DEFAULT 0 COMMENT '状态（0=准备 1=进行中 2=已结束）',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_teacher_date (teacher_id, started_at),
    INDEX idx_class_date (class_id, started_at)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='课堂会话表';

-- 作业表
CREATE TABLE assignments (
    id VARCHAR(32) PRIMARY KEY COMMENT '作业ID',
    title VARCHAR(128) NOT NULL COMMENT '作业标题',
    teacher_id VARCHAR(32) NOT NULL COMMENT '布置教师ID',
    class_id VARCHAR(32) NOT NULL COMMENT '目标班级ID',
    subject VARCHAR(16) COMMENT '学科',
    type TINYINT COMMENT '作业类型（1=练字 2=计算 3=作文 4=综合）',
    content_json TEXT COMMENT '作业内容（JSON格式，含题目和字帖）',
    due_at DATETIME COMMENT '截止时间',
    scoring_mode TINYINT DEFAULT 1 COMMENT '批改方式（1=AI批改 2=教师批改 3=AI+教师）',
    status TINYINT DEFAULT 0 COMMENT '状态（0=草稿 1=已发布 2=已截止）',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_teacher_class (teacher_id, class_id, due_at),
    INDEX idx_class_status (class_id, status)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='作业布置表';

-- 作业提交表
CREATE TABLE assignment_submissions (
    id VARCHAR(32) PRIMARY KEY COMMENT '提交记录ID',
    assignment_id VARCHAR(32) NOT NULL COMMENT '作业ID',
    student_id VARCHAR(32) NOT NULL COMMENT '学生ID',
    ink_data_ids JSON COMMENT '笔迹数据ID列表（各页对应OSS文件）',
    submit_time DATETIME COMMENT '提交时间',
    ai_score DECIMAL(5,2) COMMENT 'AI批改分数',
```

```
    teacher_score DECIMAL(5,2) COMMENT '教师批改分数',
    final_score DECIMAL(5,2) COMMENT '最终分数',
    feedback_json TEXT COMMENT '批改反馈（JSON：评语+批注+错误点）',
    status TINYINT DEFAULT 0 COMMENT '状态（0=未提交 1=已提交 2=AI批改中 3=已批改）',
    UNIQUE KEY uk_assignment_student (assignment_id, student_id),
    INDEX idx_student_status (student_id, status)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='作业提交记录表';
```

C.3 消息队列集成（RocketMQ）

C.3.1 Topic 设计

```
Topic 命名规范：writech_{业务域}_{事件类型}

生产环境 Topic 列表：
writech_classroom_events          课堂事件（开始/结束/互动）
writech_assignment_submitted      作业提交事件
writech_assignment_corrected      作业批改完成事件
writech_device_status            设备状态变更事件
writech_notification_push         推送通知事件
writech_analytics_raw_data        学情数据原始事件
```

C.3.2 作业批改异步处理流程

```
// AssignmentSubmissionConsumer.java
@RocketMQMessageListener(
    topic = "writech_assignment_submitted",
    consumerGroup = "ai-correction-group",
    consumeMode = ConsumeMode.CONCURRENTLY,
    messageModel = MessageModel.CLUSTERING
)
@Component
public class AssignmentSubmissionConsumer implements
RocketMQListener<AssignmentSubmittedEvent> {

    @Autowired
    private AiCorrectionService aiCorrectionService;

    @Autowired
    private AssignmentSubmissionRepository submissionRepo;

    @Autowired
    private RocketMQTemplate rocketMQTemplate;

    @Override
    @Transactional
    public void onMessage(AssignmentSubmittedEvent event) {
        try {
            // 步骤1：更新提交状态为"AI批改中"
            submissionRepo.updateStatus(event.getSubmissionId(),
                AssignmentStatus.AI_CORRECTING);
```

```
            // 步骤2：调用 AI 批改引擎（HTTP 调用 AI 引擎服务）
            AiCorrectionResult result = aiCorrectionService.correct(
                event.getAssignmentId(),
                event.getStudentId(),
                event.getInkDataIds()
            );

            // 步骤3：保存批改结果
            submissionRepo.saveAiResult(event.getSubmissionId(), result);

            // 步骤4：发布批改完成事件（触发通知推送）
            rocketMQTemplate.convertAndSend("writech_assignment_corrected",
                AssignmentCorrectedEvent.builder()
                    .submissionId(event.getSubmissionId())
                    .studentId(event.getStudentId())
                    .score(result.getScore())
                    .build()
            );
        } catch (AiServiceException e) {
            log.error("AI批改失败，submissionId={}", event.getSubmissionId(), e);
            // 回退到教师手动批改状态
            submissionRepo.updateStatus(event.getSubmissionId(),
                AssignmentStatus.WAITING_TEACHER);
        }
    }
}
```

C.4 文件存储与 CDN 分发

C.4.1 OSS 存储目录结构

```
writech-cloud-storage/
├── ink/                          （笔迹数据文件）
│   ├── {school_id}/
│   │   ├── {student_id}/
│   │   │   ├── {assignment_id}_p0.bin    （作业第0页笔迹，二进制压缩）
│   │   │   └── {assignment_id}_p1.bin
│   │   └── ...
│   └── ...
├── recordings/                   （课堂录像）
│   ├── {school_id}/
│   │   ├── {session_id}.mp4
│   │   └── ...
│   └── ...
├── courses/                      （课件资源）
│   ├── {school_id}/
│   │   ├── {course_id}.pptx
│   │   └── {course_id}_preview/   （预渲染页面图片）
│   └── ...
├── calligraphy/                  （字帖模板）
│   ├── templates/
│   │   ├── {template_id}_stroke.json   （笔顺数据）
```

```
│   │   └── {template_id}_ref.png       (参考图片)
│   └── ...
└── reports/                        (学情报告 PDF)
    ├── {school_id}/
    │   ├── student/
    │   │   └── {student_id}_{date}.pdf
    │   └── class/
    │       └── {class_id}_{date}.pdf
    └── ...
```

C.4.2 文件上传签名接口

```
// FileUploadController.java
@RestController
@RequestMapping("/api/v1/file")
public class FileUploadController {

    @Autowired
    private OssService ossService;

    /**
     * 获取前端直传 OSS 的预签名 URL
     * 避免文件经过服务器中转，节省带宽，提升上传速度
     */
    @PostMapping("/upload/presign")
    public ApiResult<PresignedUploadInfo> getPresignedUrl(@RequestBody PresignRequest
request) {
        // 验证文件类型和大小限制
        validateFileRequest(request);

        // 生成 OSS 存储路径
        String objectKey = generateObjectKey(request.getFileType(),
            getCurrentUserId(), request.getFileName());

        // 生成预签名 PUT URL（有效期10分钟）
        String presignedUrl = ossService.generatePresignedPutUrl(objectKey,
            Duration.ofMinutes(10));

        return ApiResult.success(PresignedUploadInfo.builder()
            .presignedUrl(presignedUrl)
            .objectKey(objectKey)
            .expireAt(LocalDateTime.now().plusMinutes(10))
            .build());
    }

    /**
     * 获取文件下载 CDN URL（带签名，防止未授权访问）
     */
    @GetMapping("/download/url")
    public ApiResult<String> getDownloadUrl(@RequestParam String objectKey) {
        // 权限验证：用户只能访问自己的文件或公开文件
        checkFileAccessPermission(objectKey);

        // 生成 CDN 签名 URL（有效期1小时）
        String signedUrl = cdnService.generateSignedUrl(objectKey, Duration.ofHours(1));
        return ApiResult.success(signedUrl);
```

```
    }

    private String generateObjectKey(String fileType, String userId, String fileName) {
        String date = LocalDate.now().format(DateTimeFormatter.BASIC_ISO_DATE);
        String ext = FilenameUtils.getExtension(fileName);
        return String.format("%s/%s/%s/%s.%s",
            fileType, getCurrentSchoolId(), userId, UUID.randomUUID(), ext);
    }
}
```

C.5 监控与告警体系

C.5.1 Prometheus 指标采集

```
// ClassroomServiceMetrics.java (Micrometer 自定义指标)
@Component
public class ClassroomServiceMetrics {

    private final Counter activeSessionCounter;
    private final Gauge activeStudentGauge;
    private final Timer inkDataProcessTimer;

    public ClassroomServiceMetrics(MeterRegistry registry) {
        // 活跃课堂计数器
        activeSessionCounter = Counter.builder("writech.classroom.sessions.total")
            .description("Total classroom sessions started")
            .tag("env", "production")
            .register(registry);

        // 实时在线学生数量 Gauge
        activeStudentGauge = Gauge.builder("writech.classroom.students.active",
            this, ClassroomServiceMetrics::getActiveStudentCount)
            .description("Current active students in all classrooms")
            .register(registry);

        // 笔迹数据处理耗时
        inkDataProcessTimer = Timer.builder("writech.ink.process.duration")
            .description("Time to process ink data batch")
            .register(registry);
    }

    public void recordInkProcessing(Runnable task) {
        inkDataProcessTimer.record(task);
    }
}
```

C.5.2 告警规则配置

```
# alertmanager-rules.yml (Prometheus 告警规则)
groups:
  - name: writech-cloud-alerts
```

```
    rules:
      # API 响应时间告警
      - alert: ApiHighLatency
        expr: histogram_quantile(0.99, http_server_requests_seconds_bucket) > 2.0
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "API P99 延迟超过 2 秒"
          description: "服务 {{ $labels.service }} 的 P99 延迟 = {{ $value }}s"

      # 错误率告警
      - alert: HighErrorRate
        expr: rate(http_server_requests_seconds_count{status=~"5.."}[5m]) /
              rate(http_server_requests_seconds_count[5m]) > 0.05
        for: 3m
        labels:
          severity: critical
        annotations:
          summary: "错误率超过 5%"

      # 课堂服务连接数告警
      - alert: TooManyActiveConnections
        expr: writech_classroom_students_active > 10000
        for: 1m
        labels:
          severity: info
        annotations:
          summary: "在线学生数超过 10000"
```

附录D 性能优化策略

D.1 数据库查询优化

读写分离配置（Spring + MyBatis）：

```
// DynamicDataSourceConfig.java
@Configuration
public class DynamicDataSourceConfig {

    @Bean
    @ConfigurationProperties("spring.datasource.master")
    public DataSource masterDataSource() {
        return DataSourceBuilder.create().build();
    }

    @Bean
    @ConfigurationProperties("spring.datasource.replica")
    public DataSource replicaDataSource() {
        return DataSourceBuilder.create().build();
    }
```

```
    @Bean
    @Primary
    public DataSource dynamicDataSource() {
        DynamicDataSource dataSource = new DynamicDataSource();
        Map<Object, Object> targetDataSources = new HashMap<>();
        targetDataSources.put(DataSourceType.MASTER, masterDataSource());
        targetDataSources.put(DataSourceType.REPLICA, replicaDataSource());
        dataSource.setTargetDataSources(targetDataSources);
        dataSource.setDefaultTargetDataSource(masterDataSource());
        return dataSource;
    }
}

// @ReadOnly 注解自动路由到从库（AOP 实现）
@Aspect
@Component
public class DataSourceAspect {
    @Around("@annotation(readOnly)")
    public Object switchToReplica(ProceedingJoinPoint pjp, ReadOnly readOnly) throws
Throwable {
        DynamicDataSourceContext.setType(DataSourceType.REPLICA);
        try {
            return pjp.proceed();
        } finally {
            DynamicDataSourceContext.clear();
        }
    }
}
```

D.2 缓存策略

缓存对象	缓存类型	TTL	失效策略
用户信息	Redis Hash	30分钟	用户信息变更时主动删除
班级学生列表	Redis List	1小时	班级成员变更时主动删除
课件资源列表	Redis String（JSON）	5分钟	定时刷新
JWT Token 黑名单	Redis Set	Token 剩余有效期	自然过期
作业统计数据	Redis Hash	10分钟	定时刷新 + 提交时触发刷新
知识点图谱	本地内存（Caffeine）	24小时	手动清除（图谱更新时）

附录E 部署与运维

E.1 Docker Compose 本地开发环境

```yaml
# docker-compose.dev.yml
version: '3.8'
services:
  mysql:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: dev_password
      MYSQL_DATABASE: writech_cloud
    ports:
      - "3306:3306"
    volumes:
      - mysql_data:/var/lib/mysql
      - ./sql/init.sql:/docker-entrypoint-initdb.d/init.sql

  redis:
    image: redis:7.0-alpine
    ports:
      - "6379:6379"

  nacos:
    image: nacos/nacos-server:v2.2.3
    environment:
      MODE: standalone
      SPRING_DATASOURCE_PLATFORM: mysql
    ports:
      - "8848:8848"
      - "9848:9848"

  rocketmq-namesrv:
    image: apache/rocketmq:5.1.0
    command: sh mqnamesrv
    ports:
      - "9876:9876"

  rocketmq-broker:
    image: apache/rocketmq:5.1.0
    command: sh mqbroker -n namesrv:9876 autoCreateTopicEnable=true
    depends_on:
      - rocketmq-namesrv

volumes:
  mysql_data:
```

E.2 Kubernetes 生产部署（关键配置）

```yaml
# classroom-service-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: classroom-service
  namespace: writech-production
spec:
  replicas: 3
  selector:
    matchLabels:
```

```
      app: classroom-service
  template:
    spec:
      containers:
        - name: classroom-service
          image: registry.writech.com/classroom-service:1.0.0
          resources:
            requests:
              cpu: "500m"
              memory: "512Mi"
            limits:
              cpu: "2000m"
              memory: "2Gi"
          env:
            - name: SPRING_PROFILES_ACTIVE
              value: production
          livenessProbe:
            httpGet:
              path: /actuator/health/liveness
              port: 8080
            initialDelaySeconds: 30
            periodSeconds: 10
          readinessProbe:
            httpGet:
              path: /actuator/health/readiness
              port: 8080
            initialDelaySeconds: 20
            periodSeconds: 5
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0  # 滚动更新时保证无停机
```

本文档版权归深圳自然写科技有限公司所有，仅用于软件著作权登记鉴别。

附录F　核心技术实现补充

F.1　Spring Security JWT认证过滤器

```
// security/JwtAuthenticationFilter.java
@Component
public class JwtAuthenticationFilter extends OncePerRequestFilter {

    private final JwtTokenProvider jwtTokenProvider;
    private final UserDetailsService userDetailsService;
    private final RedisTemplate<String, Object> redisTemplate;

    @Override
    protected void doFilterInternal(@NonNull HttpServletRequest request,
```

```
            @NonNull HttpServletResponse response,
            @NonNull FilterChain filterChain) throws ServletException, IOException {
        String token = extractToken(request);
        if (token != null) {
            try {
                // 1. 验证JWT签名与有效期
                Claims claims = jwtTokenProvider.validateToken(token);
                String userId = claims.getSubject();

                // 2. 检查Token是否已被注销（Redis黑名单）
                String blacklistKey = "jwt:blacklist:" + token;
                if (Boolean.TRUE.equals(redisTemplate.hasKey(blacklistKey))) {
                    sendUnauthorized(response, "Token已失效");
                    return;
                }

                // 3. 加载用户权限（优先从Redis缓存读取）
                String cacheKey = "user:authorities:" + userId;
                Authentication auth = (Authentication)
redisTemplate.opsForValue().get(cacheKey);
                if (auth == null) {
                    UserDetails userDetails =
userDetailsService.loadUserByUsername(userId);
                    auth = new UsernamePasswordAuthenticationToken(
                        userDetails, null, userDetails.getAuthorities());
                    redisTemplate.opsForValue().set(cacheKey, auth, 5, TimeUnit.MINUTES);
                }
                SecurityContextHolder.getContext().setAuthentication(auth);

            } catch (ExpiredJwtException e) {
                sendUnauthorized(response, "Token已过期");
                return;
            } catch (JwtException e) {
                sendUnauthorized(response, "Token无效");
                return;
            }
        }
        filterChain.doFilter(request, response);
    }

    private String extractToken(HttpServletRequest request) {
        String header = request.getHeader("Authorization");
        if (header != null && header.startsWith("Bearer ")) {
            return header.substring(7);
        }
        return null;
    }

    private void sendUnauthorized(HttpServletResponse resp, String msg) throws IOException
{
        resp.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        resp.setContentType("application/json;charset=UTF-8");
        resp.getWriter().write("{\"code\":401,\"message\":\"" + msg + "\"}");
    }
}
```

F.2 RocketMQ异步消息处理

```
// mq/HomeworkGradeConsumer.java
@Component
@RocketMQMessageListener(
    topic = "writech-homework-grade",
    consumerGroup = "homework-grade-consumer",
    consumeMode = ConsumeMode.CONCURRENTLY,
    messageModel = MessageModel.CLUSTERING
)
public class HomeworkGradeConsumer implements RocketMQListener<HomeworkGradeMessage> {

    private final InkAnalysisService inkAnalysisService;
    private final NotificationService notificationService;
    private final HomeworkService homeworkService;
    private final RedisTemplate<String, Object> redisTemplate;

    @Override
    @Transactional(rollbackFor = Exception.class)
    public void onMessage(HomeworkGradeMessage message) {
        String homeworkId = message.getHomeworkId();
        String studentId = message.getStudentId();

        try {
            // 1. 幂等检查（Redis Set防重放）
            String dedupeKey = "homework:grade:done:" + message.getMessageId();
            if (Boolean.FALSE.equals(
                    redisTemplate.opsForValue().setIfAbsent(dedupeKey, "1", 10,
TimeUnit.MINUTES))) {
                log.warn("Duplicate message ignored: {}", message.getMessageId());
                return;
            }

            // 2. 调用AI引擎分析笔迹
            InkAnalysisResult analysisResult = inkAnalysisService.analyze(
                message.getInkDataUrl(), message.getAssignmentConfig());

            // 3. 更新作业成绩
            homeworkService.updateGradeResult(homeworkId, analysisResult);

            // 4. 更新学生知识点掌握度（BKT更新）
            homeworkService.updateStudentMastery(studentId,
analysisResult.getKnowledgeResults());

            // 5. 发送推送通知给学生
            notificationService.sendGradeNotification(studentId, homeworkId,
                analysisResult.getTotalScore());

            log.info("Homework graded: homeworkId={}, student={}, score={}",
                homeworkId, studentId, analysisResult.getTotalScore());

        } catch (Exception e) {
            log.error("Grade homework failed: {}", homeworkId, e);
            throw new RuntimeException("Grade processing failed", e); // 触发重试
        }
```

```
    }
}
```

F.3 数据库分区与多级存储策略

```
-- MySQL 8.0 数据库分区设计

-- 笔迹数据表（按月分区，自动清理超过24个月的热数据）
CREATE TABLE ink_strokes (
    id           BIGINT UNSIGNED AUTO_INCREMENT,
    session_id   CHAR(36) NOT NULL,
    student_id   CHAR(36) NOT NULL,
    homework_id  CHAR(36),
    stroke_data  MEDIUMBLOB NOT NULL,  -- 压缩后的笔迹原始数据
    point_count  SMALLINT UNSIGNED NOT NULL,
    score        TINYINT UNSIGNED,     -- OCR/批改评分
    created_at   DATETIME NOT NULL,
    PRIMARY KEY (id, created_at)
) ENGINE=InnoDB
PARTITION BY RANGE (TO_DAYS(created_at)) (
    PARTITION p202501 VALUES LESS THAN (TO_DAYS('2025-02-01')),
    PARTITION p202502 VALUES LESS THAN (TO_DAYS('2025-03-01')),
    -- ... 自动建月分区
    PARTITION p202601 VALUES LESS THAN (TO_DAYS('2026-02-01')),
    PARTITION p_future VALUES LESS THAN MAXVALUE
);

-- 分区维护存储过程（每月1日凌晨3点自动执行）
DELIMITER $$
CREATE PROCEDURE maintain_ink_partitions()
BEGIN
    DECLARE next_month_start DATE;
    DECLARE partition_name VARCHAR(20);
    DECLARE boundary_date DATE;

    -- 创建下下个月的分区
    SET next_month_start = DATE_FORMAT(DATE_ADD(NOW(), INTERVAL 2 MONTH), '%Y-%m-01');
    SET partition_name = CONCAT('p', DATE_FORMAT(next_month_start, '%Y%m'));
    SET boundary_date = DATE_ADD(next_month_start, INTERVAL 1 MONTH);

    SET @sql = CONCAT('ALTER TABLE ink_strokes REORGANIZE PARTITION p_future INTO (',
        'PARTITION ', partition_name, ' VALUES LESS THAN (TO_DAYS(''', boundary_date,
''')),',
        'PARTITION p_future VALUES LESS THAN MAXVALUE)');
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

    -- 将24个月前的分区数据归档到OSS（通过应用层调度）
    INSERT INTO archive_tasks (table_name, partition_name, scheduled_at)
    VALUES ('ink_strokes',
            CONCAT('p', DATE_FORMAT(DATE_SUB(NOW(), INTERVAL 24 MONTH), '%Y%m')),
            NOW());
END$$
DELIMITER ;
```

F.4 Nacos服务注册配置

```yaml
# application-nacos.yml
spring:
  cloud:
    nacos:
      discovery:
        server-addr: nacos:8848
        namespace: writech-prod
        group: WRITECH_GROUP
        # 健康检查配置
        heart-beat-interval: 5000
        heart-beat-timeout: 15000
        ip-delete-timeout: 30000
        # 服务元数据
        metadata:
          version: 1.0.0
          env: production
          region: cn-shenzhen
      config:
        server-addr: nacos:8848
        namespace: writech-prod
        group: WRITECH_GROUP
        file-extension: yaml
        shared-configs:
          - data-id: common-datasource.yaml
            group: COMMON_GROUP
            refresh: true
          - data-id: common-redis.yaml
            group: COMMON_GROUP
            refresh: true
        # 配置变更监听（热更新）
        refresh-enabled: true

# Spring Cloud Gateway路由配置
  cloud:
    gateway:
      routes:
        - id: ink-service
          uri: lb://writech-ink-service
          predicates:
            - Path=/api/v1/ink/**
          filters:
            - StripPrefix=0
            - name: RequestRateLimiter
              args:
                redis-rate-limiter.replenishRate: 100
                redis-rate-limiter.burstCapacity: 200
                key-resolver: "#{@appKeyResolver}"
            - name: CircuitBreaker
              args:
                name: inkServiceCB
                fallbackUri: forward:/fallback/ink

        - id: ai-engine
          uri: lb://writech-ai-engine
```

```
          predicates:
            - Path=/api/v1/ocr/**, /api/v1/stroke/**
          filters:
            - StripPrefix=0
            - AddRequestHeader=X-Gateway-Time, #{T(System).currentTimeMillis()}
```

F.5 Prometheus告警规则

```
# prometheus/alerts/writech_alerts.yaml
groups:
  - name: writech_business_alerts
    rules:
      # 作业批改队列积压告警
      - alert: HomeworkGradeQueueBacklog
        expr: writech_rocketmq_consumer_lag{topic="writech-homework-grade"} > 1000
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "作业批改队列积压超过1000条"
          description: "当前积压：{{ $value }}条，持续超过5分钟"

      # API P99延迟告警
      - alert: ApiHighLatency
        expr: histogram_quantile(0.99, rate(http_server_requests_seconds_bucket[5m])) >
2.0
        for: 3m
        labels:
          severity: critical
        annotations:
          summary: "API P99延迟超过2秒"
          description: "P99: {{ $value | humanizeDuration }}"

      # 数据库连接池耗尽告警
      - alert: DbConnectionPoolExhausted
        expr: hikaricp_connections_active / hikaricp_connections_max > 0.9
        for: 2m
        labels:
          severity: critical
        annotations:
          summary: "数据库连接池使用率超过90%"

      # 磁盘空间告警
      - alert: DiskSpaceLow
        expr: node_filesystem_avail_bytes{mountpoint="/"} / node_filesystem_size_bytes <
0.1
        for: 5m
        labels:
          severity: warning
        annotations:
          summary: "磁盘空间剩余不足10%"
```

附录F 补充技术规格

F.1 多租户数据隔离设计

F.1.1 行级数据隔离实现

```
// TenantAwareRepository.java
@Repository
public class TenantAwareRepository<T> {

    @PersistenceContext
    private EntityManager em;

    @Autowired
    private TenantContextHolder tenantContext;

    public List<T> findAllForCurrentTenant(Class<T> entityClass) {
        String tenantId = tenantContext.getCurrentTenantId();

        CriteriaBuilder cb = em.getCriteriaBuilder();
        CriteriaQuery<T> query = cb.createQuery(entityClass);
        Root<T> root = query.from(entityClass);

        // 自动注入租户过滤条件
        query.where(cb.equal(root.get("tenantId"), tenantId));

        return em.createQuery(query).getResultList();
    }

    public T save(T entity) {
        String tenantId = tenantContext.getCurrentTenantId();
        // 自动设置租户ID
        setTenantId(entity, tenantId);
        em.persist(entity);
        return entity;
    }

    private void setTenantId(Object entity, String tenantId) {
        try {
            Field field = entity.getClass().getDeclaredField("tenantId");
            field.setAccessible(true);
            field.set(entity, tenantId);
        } catch (Exception e) {
            throw new RuntimeException("实体类缺少tenantId字段", e);
        }
    }
}
```

F.2 WebSocket实时推送服务

F.2.1 课堂数据实时推送

```java
// ClassroomWebSocketHandler.java
@Component
public class ClassroomWebSocketHandler extends TextWebSocketHandler {

    // 教室ID → 订阅该教室的WebSocket会话集合
    private final ConcurrentHashMap<String, Set<WebSocketSession>>
        classroomSessions = new ConcurrentHashMap<>();

    @Override
    public void afterConnectionEstablished(WebSocketSession session) {
        String classroomId = extractClassroomId(session);
        classroomSessions.computeIfAbsent(classroomId,
            k -> ConcurrentHashMap.newKeySet()).add(session);

        log.info("WebSocket connected: session={}, classroom={}",
                 session.getId(), classroomId);
    }

    @Override
    public void afterConnectionClosed(WebSocketSession session,
                                      CloseStatus status) {
        String classroomId = extractClassroomId(session);
        Set<WebSocketSession> sessions = classroomSessions.get(classroomId);
        if (sessions != null) {
            sessions.remove(session);
            if (sessions.isEmpty()) classroomSessions.remove(classroomId);
        }
    }

    // 向指定教室的所有订阅者推送消息
    public void broadcastToClassroom(String classroomId, Object message) {
        Set<WebSocketSession> sessions = classroomSessions.get(classroomId);
        if (sessions == null || sessions.isEmpty()) return;

        String json = objectMapper.writeValueAsString(message);
        TextMessage wsMessage = new TextMessage(json);

        sessions.removeIf(session -> {
            try {
                if (session.isOpen()) {
                    session.sendMessage(wsMessage);
                    return false;
                }
                return true; // 自动移除已关闭的会话
            } catch (IOException e) {
                log.warn("推送失败，移除会话 {}", session.getId());
                return true;
            }
        });
    }

    private String extractClassroomId(WebSocketSession session) {
        URI uri = session.getUri();
        // URL格式：/ws/classroom/{classroomId}
        String[] parts = uri.getPath().split("/");
        return parts[parts.length - 1];
```

```
    }
}
```

F.3 批改任务分布式锁

```
// HomeworkGradingService.java
@Service
public class HomeworkGradingService {

    @Autowired
    private RedissonClient redisson;

    @Autowired
    private RocketMQTemplate rocketMQTemplate;

    public void triggerBatchGrading(String classId, String homeworkId) {
        // 使用分布式锁防止重复触发
        String lockKey = String.format("grading:lock:%s:%s", classId, homeworkId);
        RLock lock = redisson.getLock(lockKey);

        try {
            boolean acquired = lock.tryLock(0, 60, TimeUnit.SECONDS);
            if (!acquired) {
                log.info("批改任务已在运行中，跳过重复触发: {}", homeworkId);
                return;
            }

            // 查询待批改作业
            List<Submission> pending = submissionRepo.findPending(homeworkId);
            log.info("触发批量批改: homeworkId={}, count={}",
                     homeworkId, pending.size());

            // 发送到RocketMQ批改队列
            for (List<Submission> batch : Lists.partition(pending, 50)) {
                GradingBatchMessage msg = new GradingBatchMessage();
                msg.setHomeworkId(homeworkId);
                msg.setSubmissionIds(batch.stream()
                    .map(Submission::getId).collect(Collectors.toList()));

                rocketMQTemplate.asyncSend("homework-grading-topic",
                    msg, new SendCallback() {
                        @Override
                        public void onSuccess(SendResult result) {}
                        @Override
                        public void onException(Throwable e) {
                            log.error("批改消息发送失败", e);
                        }
                    });
            }

        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        } finally {
            if (lock.isHeldByCurrentThread()) lock.unlock();
        }
```

```
    }
}
```

附录G 补充技术规格

G.1 微服务链路追踪

```
// TracingConfig.java
@Configuration
public class TracingConfig {

    @Bean
    public Tracer zipkinTracer(ZipkinProperties props) {
        OkHttpSender sender = OkHttpSender.create(props.getBaseUrl() + "/api/v2/spans");
        AsyncReporter<Span> reporter = AsyncReporter.create(sender);

        return Tracing.newBuilder()
            .localServiceName("writech-cloud-platform")
            .spanReporter(reporter)
            .sampler(Sampler.create(props.getSampleRate()))
            .build()
            .tracer();
    }
}

// 在Service层使用追踪
@Service
public class TracedHomeworkService {

    @Autowired
    private Tracer tracer;

    public HomeworkResult gradeHomework(String submissionId) {
        Span span = tracer.newTrace().name("grade-homework")
            .tag("submission.id", submissionId)
            .start();

        try (Tracer.SpanInScope ws = tracer.withSpanInScope(span)) {
            // 1. 获取提交数据
            Span fetchSpan = tracer.newChild(span.context())
                .name("fetch-submission").start();
            Submission sub = submissionRepo.findById(submissionId);
            fetchSpan.finish();

            // 2. 调用AI批改
            Span aiSpan = tracer.newChild(span.context())
                .name("ai-grading").start();
            GradingResult result = aiClient.grade(sub);
            aiSpan.finish();

            return result;
        } finally {
```

```
            span.finish();
        }
    }
}
```

G.2 配置中心集成

```
// NacosConfigRefresher.java
@RefreshScope
@Configuration
public class NacosConfigRefresher {

    @Value("${writech.grading.timeout-ms:5000}")
    private int gradingTimeoutMs;

    @Value("${writech.storage.hot-threshold-days:30}")
    private int hotStorageThresholdDays;

    @Value("${writech.ratelimit.qps-per-user:10}")
    private int rateLimitQpsPerUser;

    @NacosValue(value = "${writech.feature.ai-grading-enabled:true}", autoRefreshed =
true)
    private boolean aiGradingEnabled;

    @NacosValue(value = "${writech.feature.real-time-feedback:true}", autoRefreshed =
true)
    private boolean realTimeFeedbackEnabled;

    // Getters
    public int getGradingTimeoutMs() { return gradingTimeoutMs; }
    public boolean isAiGradingEnabled() { return aiGradingEnabled; }
    public boolean isRealTimeFeedbackEnabled() { return realTimeFeedbackEnabled; }
}
```

附录H 补充技术规格

H.1 数据导出API

```
// DataExportController.java
@RestController
@RequestMapping("/api/v1/export")
public class DataExportController {

    @GetMapping("/class/{classId}/homework-report")
    public ResponseEntity<byte[]> exportClassHomeworkReport(
            @PathVariable String classId,
            @RequestParam @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate
startDate,
            @RequestParam @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate
```

```
endDate,
            @RequestParam(defaultValue = "excel") String format) {

        byte[] data;
        String contentType;
        String filename;

        if ("excel".equals(format)) {
            data = excelExportService.exportClassAnalytics(classId, startDate, endDate);
            contentType = "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet";
            filename = String.format("班级作业报告_%s_%s.xlsx", startDate, endDate);
        } else {
            data = reportGenerationService.generateClassReport(classId, startDate,
endDate);
            contentType = "application/pdf";
            filename = String.format("班级作业报告_%s_%s.pdf", startDate, endDate);
        }

        return ResponseEntity.ok()
            .contentType(MediaType.parseMediaType(contentType))
            .header(HttpHeaders.CONTENT_DISPOSITION,
                "attachment; filename*=UTF-8''" +
                URLEncoder.encode(filename, StandardCharsets.UTF_8))
            .body(data);
    }

    @GetMapping("/student/{studentId}/learning-record")
    public ResponseEntity<List<LearningRecord>> exportStudentLearningRecord(
            @PathVariable String studentId,
            @RequestParam @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate
startDate,
            @RequestParam @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate
endDate) {

        List<LearningRecord> records = learningRecordService
            .getRecords(studentId, startDate, endDate);

        return ResponseEntity.ok(records);
    }
}
```

H.2 健康检查端点

```
// HealthCheckController.java
@RestController
@RequestMapping("/actuator")
public class HealthCheckController {

    @GetMapping("/health")
    public Map<String, Object> health() {
        Map<String, Object> status = new LinkedHashMap<>();
        status.put("status", "UP");
        status.put("timestamp", Instant.now().toString());
```

```
        // 数据库连接检查
        try {
            jdbcTemplate.queryForObject("SELECT 1", Integer.class);
            status.put("database", Map.of("status", "UP"));
        } catch (Exception e) {
            status.put("database", Map.of("status", "DOWN", "error", e.getMessage()));
            status.put("status", "DEGRADED");
        }

        // Redis连接检查
        try {
            redisTemplate.opsForValue().get("health:ping");
            status.put("redis", Map.of("status", "UP"));
        } catch (Exception e) {
            status.put("redis", Map.of("status", "DOWN"));
        }

        return status;
    }
}
```

H.3 版本历史

版本号	发布日期	变更说明	负责人
V1.0.0	2024-01-15	初始版本发布，包含核心云平台功能	研发团队
V1.1.0	2024-03-20	新增RocketMQ异步批改消息队列	后端组
V1.2.0	2024-05-10	引入多级存储热温冷分层策略	架构组
V1.3.0	2024-07-01	集成Nacos配置中心，支持动态配置热更新	运维组
V1.4.0	2024-09-15	添加Zipkin链路追踪，提升问题排查能力	研发团队
V1.5.0	2024-11-01	完善数据导出功能，支持Excel/PDF格式	产品组

本文档版权归深圳自然写科技有限公司所有，仅用于软件著作权登记鉴别。