**详细设计规约**

**目录**

[修订历史记录 3](#_Toc16962)

[1. 引言 5](#_Toc15524)

[1.1 编写目的 5](#_Toc10377)

[1.2 背景与依据 5](#_Toc22315)

[1.3 参考资料 6](#_Toc10752)

[2. 系统软件体系结构 6](#_Toc13377)

[3. 子系统设计 6](#_Toc28785)

[3.1 子系统描述 6](#_Toc16761)

[3.1.1 账号管理子系统 6](#_Toc14922)

[3.1.2 训练分析子系统 6](#_Toc22025)

[3.1.3 俱乐部子系统 7](#_Toc21093)

[3.1.4 赛艇社区子系统 7](#_Toc2041)

[3.2 功能（包含哪些微服务或类） 7](#_Toc23282)

[3.2.1 账号管理子系统 7](#_Toc27611)

[3.2.2 训练分析子系统 7](#_Toc11479)

[3.2.3 俱乐部子系统 8](#_Toc6954)

[3.2.4 赛艇社区子系统 8](#_Toc22284)

[3.3 和哪些子系统有交互（接口） 8](#_Toc32453)

[3.3.1 账号管理子系统 8](#_Toc26031)

[3.3.2 训练分析子系统 8](#_Toc1569)

[3.3.3 俱乐部子系统 8](#_Toc5866)

[3.3.4 赛艇社区子系统 8](#_Toc31012)

[4. 微服务设计 9](#_Toc5091)

[4.1 微服务功能 9](#_Toc16846)

[4.1.1 业务服务 9](#_Toc13899)

[4.1.2 基础设施服务 10](#_Toc13325)

[4.2 和哪些微服务有交互（接口） 11](#_Toc26440)

[4.2.1 用户服务 / 个人信息服务： 11](#_Toc19840)

[4.2.2 训练数据服务 / 大语言模型分析服务 / 姿态识别服务： 11](#_Toc19151)

[4.2.3 俱乐部管理服务 / 活动服务： 11](#_Toc29676)

[4.2.4 帖子服务 / 互动服务 / 搜索服务： 11](#_Toc17064)

[4.2.5 网关服务： 12](#_Toc12459)

[4.2.6 OSS 对象存储服务： 12](#_Toc28334)

[4.2.7 分布式 ID 生成服务： 12](#_Toc6251)

[5. 类设计 12](#_Toc15380)

[5.1 功能 12](#_Toc18922)

[5.2 性能 13](#_Toc25514)

[5.3 输入项（输入接口） 13](#_Toc918)

[1. UserController 13](#_Toc30587)

[2. TrainingController 13](#_Toc3904)

[5.4 输出项（输出接口） 14](#_Toc25154)

[1.用户相关反映 14](#_Toc31804)

[2. 训练相关相应 14](#_Toc3964)

[5.5 属性详细描述 15](#_Toc9123)

[1. User 类属性 15](#_Toc26921)

[2. TrainingSession 类属性 15](#_Toc3992)

[5.6 方法详细描述及实现流程 15](#_Toc121)

[1. 用户注册流程 15](#_Toc26043)

[2. 训练创建流程 16](#_Toc8438)

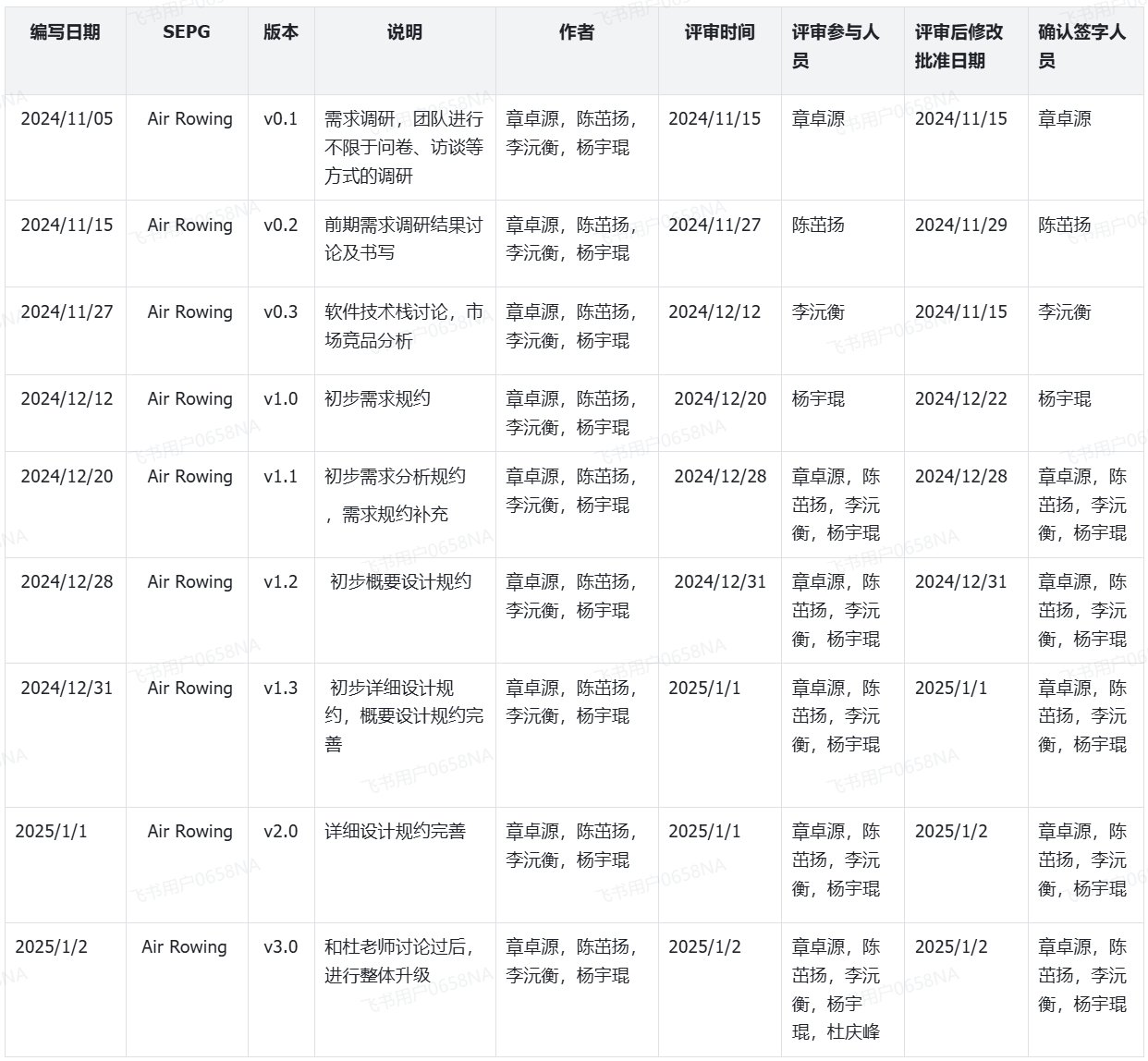
[5.7 类内、类外方法调用状态图 17](#_Toc13872)

[5.8 限制条件及出错处理 18](#_Toc18925)

[1. 输入验证 18](#_Toc23527)

[2. 异常处理 19](#_Toc10269)

**修订历史记录**

****

1. **引言**

**1.1 编写目的**

本详细设计文档旨在为 **Air Rowing 系统** 的开发团队提供清晰、全面的设计指导。通过描述系统各子系统、微服务及其内部类的设计考虑，确保系统的高效实现、可维护性和可扩展性。该文档作为概要设计的延伸，进一步细化功能模块的内部结构和接口定义，以及类设计，为开发和后续维护提供依据。

**1.2 背景与依据**

详细设计阶段的开展基于以下几个方面的背景与依据：

* **概要设计基本原则**

概要设计阶段遵循模块化、可扩展性、高内聚低耦合等基本原则，确保系统架构清晰、逻辑严谨，为详细设计奠定坚实基础。

* **详细设计基本原则**

详细设计阶段强调接口清晰、设计模式的合理应用、代码可维护性和可测试性。通过采用标准化的设计方法，确保各组件功能明确，系统整体性能优越。

* **Air Rowing 需求规约**

本详细设计依据 **Air Rowing 系统** 的需求规约文档，全面理解系统需实现的功能和性能要求，确保设计方案符合用户需求和业务目标。

* **Air Rowing 需求分析规约**

需求分析规约提供了系统的功能需求和非功能需求的详细描述，指导详细设计阶段针对性地解决各类需求，确保系统设计的全面性和准确性。

* **Air Rowing 概要设计规约**

概要设计规约描述了系统的整体架构、主要子系统及其交互方式。详细设计在此基础上，进一步细化各子系统和微服务的内部结构、接口设计及数据流动，确保系统设计的连贯性和一致性。

**1.3 参考资料**

* **设计模式**

参考经典设计模式（如《Design Patterns》），应用于系统设计中，以提高系统的可复用性、可扩展性和维护性。设计模式的合理使用有助于解决常见的设计问题，优化系统结构。

* **软件工程--实践者的研究方法（第8版）**

基于《软件工程--实践者的研究方法（第8版）》中的理论和方法论，指导详细设计的实施。书中的最佳实践和方法论为系统设计提供了科学、系统的支持，确保设计过程的规范性和高效性。

2. **系统软件体系结构**

**概要设计的2.2节中的软件体系结构**

3. **子系统设计**

**3.1 子系统描述**

AirRowing 系统采用微服务架构，通过功能拆分将整个系统划分为多个子系统，包括：

**3.1.1 账号管理子系统**

* **主要功能**：注册登录、密码修改、个人信息查看与编辑；确保用户数据的安全与隐私。
* **作用范围**：提供统一的身份认证和权限管理，为其他子系统提供用户信息和权限验证接口。

**3.1.2 训练分析子系统**

* **主要功能**：
* 用户可以添加和管理训练记录，支持多种训练数据（如距离、时间、心率等）的输入与存储。
* 通过大语言模型分析训练数据，提供训练效果评估，帮助用户了解自身训练进展与不足。
* 通过大语言模型分析用户上传的赛艇图片或视频，提供精准的姿态评估与纠正建议。
* 利用人工智能 RAG 技术，为用户提供赛艇训练的智能指导。
* 用户可将训练成果分享到社区，与其他用户交流经验。
* **作用范围**：对用户训练数据进行全面处理与分析，在增强用户体验的同时，提高训练效率和质量。

**3.1.3 俱乐部子系统**

* **主要功能**：俱乐部信息维护、训练和比赛报名管理、成员数据统计与计划制定。
* **作用范围**：集中管理俱乐部的日常活动、成员信息和比赛/训练统计，为俱乐部管理员提供科学的训练计划制定依据。

**3.1.4 赛艇社区子系统**

* **主要功能**：社区帖子发布与管理、评论和点赞交互、个人主页查看与搜索功能。
* **作用范围**：为用户提供线上交流平台，分享训练心得、赛艇资讯，实现社区互动与资源共享。

**3.2 功能（包含哪些微服务或类）**

**3.2.1 账号管理子系统**

* **用户服务（User Service）**：注册、登录、权限验证、密码修改
* **个人信息服务（Profile Service）**：查看、编辑个人信息，隐私设置

**3.2.2 训练分析子系统**

* **训练数据服务（Training Data Service）**：训练记录的创建、更新、查询及存储
* **大语言模型分析服务（LLM Analysis Service）**：通过大语言模型对训练数据进行分析，输出训练效果评估
* **姿态识别服务（Pose Recognition Service）**：通过大语言模型识别用户上传的图片或视频，评估划船姿态并给出纠正建议

**3.2.3 俱乐部子系统**

* **俱乐部管理服务（Club Management Service）**：俱乐部信息增删改查、俱乐部动态更新
* **活动服务（Activity Service）**：训练或比赛活动的报名管理

**3.2.4 赛艇社区子系统**

* **帖子服务（Post Service）**：发帖、删帖、编辑、查看帖子详情
* **互动服务（Interaction Service）**：评论发布、点赞、收藏、关注等互动操作
* **搜索服务（Search Service）**：按关键词或用户进行检索，支持高级筛选和排序

**3.3 和哪些子系统有交互（接口）**

具体接口描述在**“概要设计”**中已经列出，这里只从系统层面介绍交互。

**3.3.1 账号管理子系统**

* 向所有其他子系统提供用户身份验证与权限校验接口。
* 为「训练分析子系统」校验训练数据提交、姿态识别等操作的合法性。
* 为「俱乐部子系统」验证俱乐部管理员身份、活动报名者身份。
* 为「赛艇社区子系统」验证发帖评论者的合法身份。

**3.3.2 训练分析子系统**

* 调用「账号管理子系统」验证用户身份。
* 与「赛艇社区子系统」联动，支持用户分享训练数据和分析结果到社区帖子。
* 与「俱乐部子系统」交互，为俱乐部管理员提供成员训练数据和姿态分析报表。

**3.3.3 俱乐部子系统**

* 调用「账号管理子系统」进行俱乐部管理员权限校验。
* 从「训练分析子系统」获取俱乐部成员训练和姿态数据，用于制定训练计划或查看比赛成绩。
* 与「赛艇社区子系统」交互，在社区发布俱乐部动态，展示成员优秀成绩或训练通知。

**3.3.4 赛艇社区子系统**

* 调用「账号管理子系统」进行发帖、评论、点赞等操作的身份校验。
* 可与「训练分析子系统」对接，分享用户的训练数据、姿态分析结果。
* 与「俱乐部子系统」共享部分活动信息，使用户在社区也能了解俱乐部最新动态。

4. **微服务设计**

**4.1 微服务功能**

**4.1.1 业务服务**

**4.1.1.1 用户服务（User Service）**

* 用户注册、登录、身份校验
* 密码找回与修改
* 维护用户启用/禁用状态

**4.1.1.2 个人信息服务（Profile Service）**

* 管理用户的个人资料（头像、昵称、简介等）
* 提供用户隐私设置和对外信息查询接口

**4.1.1.3 训练数据服务（Training Data Service）**

* 用户训练数据（距离、时长、心率、功率等）的记录与管理
* 数据查询和统计（周期性或按需求）

**4.1.1.4 大语言模型分析服务（LLM Analysis Service）**

* 使用大语言模型（LLM）对训练数据进行语义理解与评估
* 输出训练效果分析结果，帮助用户了解自身训练进展与不足
* 支持根据自然语言描述，解读用户输入的训练问题与需求

**4.1.1.5 姿态识别服务（Pose Recognition Service）**

* 接收用户上传的赛艇图片/视频，利用大语言模型和图像/视频处理技术进行姿态分析
* 返回具体的姿态评估与纠正建议，含关节角度、动作节奏等信息

**4.1.1.6 俱乐部管理服务（Club Management Service）**

* 支持俱乐部信息的增删改查、俱乐部动态发布、管理者权限校验
* 生成俱乐部专属数据报表（如赛事成绩、集体训练进展）

**4.1.1.7 活动服务（Activity Service）**

* 处理各类训练/比赛的报名流程，跟踪成员信息

**4.1.1.8 帖子服务（Post Service）**

* 发帖、删帖、编辑、获取帖子详情
* 与搜索服务结合，实现按关键词或标签检索帖子

**4.1.1.9 互动服务（Interaction Service）**

* 支持评论、点赞、收藏、关注等互动功能，更新相关计数或用户活跃度
* 记录并统计用户在社区的贡献度

**4.1.1.10 搜索服务（Search Service）**

* 对帖子、俱乐部信息或用户进行搜索
* 提供模糊搜索、标签搜索等高级检索能力

**4.1.2 基础设施服务**

**4.1.2.1 网关服务**

* **功能**：对外统一入口，处理请求路由及负载均衡，进行初步的权限与参数校验。
* **说明**：外部请求先经过网关服务，再根据路径或规则转发至后端对应的业务微服务。

**4.1.2.2 认证服务**

* **功能**：生成与验证用户 Token（或 Session），支持多种认证（如 JWT、OAuth2 等）；管理用户授权信息。
* **说明**：与用户服务对接，用于在用户登录态和资源访问权限上进行严格把控。

**4.1.2.3 分布式id生成服务**

* **功能**：采用分布式算法（如 Snowflake）生成全局唯一 ID；保证系统内各业务主键的一致性与高效性。
* **说明**：在高并发场景下有效避免重复 ID 和数据库冲突，可应用于帖子、训练记录、俱乐部、活动等多领域主键生成。

**4.1.2.4 OSS对象存储服务**

* **功能**：管理用户上传的图片、视频等大文件；支持高并发读写和断点续传。
* **说明**：姿态识别服务可从该服务获取多媒体文件进行分析；个人信息服务可将头像等资料上传至 OSS，引用对应的访问链接。

**4.2 和哪些微服务有交互（接口）**

具体接口描述在**“概要设计”**中已经列出，这里只从系统层面介绍交互。

**4.2.1 用户服务 / 个人信息服务：**

* 被大多数业务微服务调用，用于用户身份验证与个人信息获取。
* 与认证服务交互，对登录态进行验证与权限管理。

**4.2.2 训练数据服务 / 大语言模型分析服务 / 姿态识别服务：**

* 四者之间互相调用，形成训练数据采集、智能分析和个性化指导的闭环。
* 与 OSS 对象存储服务对接，实现视频/图片的上传与获取。
* 与分布式 ID 生成服务结合，为训练记录生成唯一标识。

**4.2.3 俱乐部管理服务 / 活动服务：**

* 通过用户服务和认证服务实现权限校验；
* 可从训练分析子系统获取成员训练数据，用于俱乐部级别的分析与赛事组织；
* 与搜索服务交互，支持俱乐部或活动信息的检索与展示。

**4.2.4 帖子服务 / 互动服务 / 搜索服务：**

* 共同构成社区的主要功能板块；
* 用户服务和认证服务确保发帖评论者的合法身份；
* 分布式 ID 生成服务为帖子、评论生成唯一标识；
* 互动服务提供社区活跃度统计；搜索服务可对社区内容进行多维度检索；
* 可以结合训练数据或俱乐部信息进行帖子扩展（如分享训练成果、活动通知等）。

**4.2.5 网关服务：**

* 对外统一网关入口，将外部请求转发至对应业务微服务；
* 与认证服务配合完成最初的用户 Token 校验与权限检查。

**4.2.6 OSS 对象存储服务：**

* 与姿态识别服务联动，提供图片/视频的存储与读取；
* 与个人信息服务对接，保存用户头像等资源；
* 与帖子服务结合，可能用于帖子中的图片或视频附件。

**4.2.7 分布式 ID 生成服务：**

* 与各业务服务对接，为需要唯一主键的业务对象（帖子、训练记录、俱乐部、活动等）提供 ID 生成接口。

5. **类设计**

**5.1 功能**

1. 用户管理类 (UserService)

用户注册

用户登录

用户信息管理

权限验证

1. 训练管理类 (TrainingService)

训练计划创建和管理

训练会话记录

实时数据采集

训练数据存储

1. 数据分析类 (AnalysisService)

训练数据分析

性能评估

进度追踪

报告生成

**5.2 性能**

1. 响应时间

用户操作响应时间 < 1秒

数据分析处理时间 < 3秒

实时数据传输延迟 < 100ms

1. 并发处理

支持同时100个用户在线训练

数据库连接池最大连接数200

**5.3 输入项（输入接口）**

**1. UserController**

|  |
| --- |
| Java public interface UserController { // 用户注册  @POST("/api/auth/register")  ResponseEntity<User> register(@RequestBody UserDTO userDTO);  // 用户登录  @POST("/api/auth/login")  ResponseEntity<AuthToken> login(@RequestBody LoginDTO loginDTO);  // 更新用户信息  @PUT("/api/users/{userId}")  ResponseEntity<User> updateUser(@PathVariable Long userId, @RequestBody UserDTO userDTO); } |

**2. TrainingController**

|  |
| --- |
| Java public interface TrainingController {  // 创建训练计划  @POST("/api/training/plan")  ResponseEntity<TrainingPlan> createPlan(@RequestBody TrainingPlanDTO planDTO);    // 开始训练会话  @POST("/api/training/session")  ResponseEntity<TrainingSession> startSession(@RequestBody SessionDTO sessionDTO);    // 提交训练数据  @POST("/api/training/data")  ResponseEntity<DataAnalysis> submitData(@RequestBody TrainingDataDTO dataDTO); } |

**5.4 输出项（输出接口）**

**1.用户相关反映**

|  |
| --- |
| Java public class UserResponse {  private Long userId;  private String username;  private String email;  private UserProfile profile;  private List<Role> roles; }  public class AuthResponse {  private String token;  private Long expiresIn;  private String tokenType; } |

**2. 训练相关相应**

|  |
| --- |
| Java public class TrainingResponse {  private Long sessionId;  private TrainingStats stats;  private List<TrainingMetrics> metrics;  private AnalysisResult analysis; } |

**5.5 属性详细描述**

**1. User 类属性**

|  |
| --- |
| Java public class User {  private Long id; // 用户ID  private String username; // 用户名  private String email; // 邮箱  private String password; // 加密密码  private UserProfile profile; // 用户档案  private List<Role> roles; // 用户角色  private Date createTime; // 创建时间  private Date updateTime; // 更新时间 } |

**2. TrainingSession 类属性**

|  |
| --- |
| Java public class TrainingSession {  private Long id; // 会话ID  private Long userId; // 用户ID  private Date startTime; // 开始时间  private Date endTime; // 结束时间  private TrainingType type;// 训练类型  private TrainingStats stats; // 训练统计  private List<TrainingData> dataPoints; // 训练数据点 } |

**5.6 方法详细描述及实现流程**

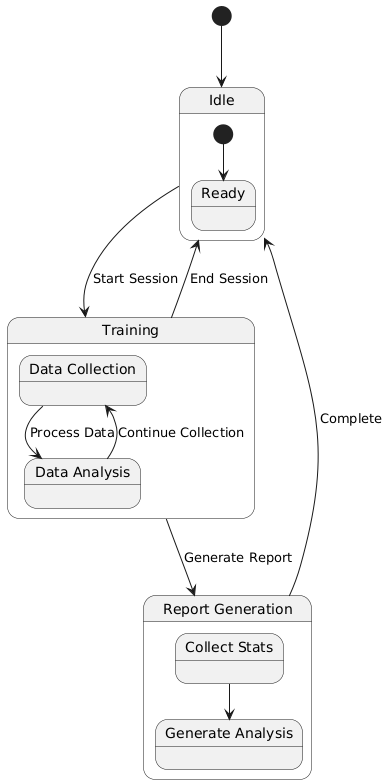
**1. 用户注册流程**

|  |
| --- |
| Java public User registerUser(UserDTO userDTO) {  // 1. 验证输入数据  validateUserInput(userDTO);    // 2. 检查用户是否存在  if (userRepository.existsByEmail(userDTO.getEmail())) {  throw new UserExistsException();  }    // 3. 创建用户  User user = new User();  user.setUsername(userDTO.getUsername());  user.setEmail(userDTO.getEmail());  user.setPassword(passwordEncoder.encode(userDTO.getPassword()));    // 4. 分配默认角色  user.setRoles(Collections.singletonList(Role.USER));    // 5. 保存用户  user = userRepository.save(user);    // 6. 创建用户档案  createUserProfile(user);    return user; } |

**2. 训练创建流程**

|  |
| --- |
| Java public TrainingSession createTrainingSession(SessionDTO sessionDTO) {  // 1. 验证用户权限  validateUserPermission(sessionDTO.getUserId());    // 2. 创建会话  TrainingSession session = new TrainingSession();  session.setUserId(sessionDTO.getUserId());  session.setStartTime(new Date());  session.setType(sessionDTO.getType());    // 3. 初始化训练统计  session.setStats(new TrainingStats());    // 4. 保存会话  session = sessionRepository.save(session);    // 5. 启动数据采集  startDataCollection(session);    return session; } |

**5.7 类内、类外方法调用状态图**



**5.8 限制条件及出错处理**

**1. 输入验证**

|  |
| --- |
| Java public class InputValidator {  // 用户名验证  public void validateUsername(String username) {  if (username == null || username.length() < 3) {  throw new InvalidInputException("Username must be at least 3 characters");  }  }    // 邮箱验证  public void validateEmail(String email) {  if (!email.matches("^[A-Za-z0-9+\_.-]+@(.+)$")) {  throw new InvalidInputException("Invalid email format");  }  } } |

**2. 异常处理**

|  |
| --- |
| Java @ControllerAdvice public class GlobalExceptionHandler {  @ExceptionHandler(UserExistsException.class)  public ResponseEntity<ErrorResponse> handleUserExists(UserExistsException ex) {  return new ResponseEntity<>(  new ErrorResponse("User already exists"),  HttpStatus.CONFLICT  );  }    @ExceptionHandler(InvalidInputException.class)  public ResponseEntity<ErrorResponse> handleInvalidInput(InvalidInputException ex) {  return new ResponseEntity<>(  new ErrorResponse(ex.getMessage()),  HttpStatus.BAD\_REQUEST  );  } } |