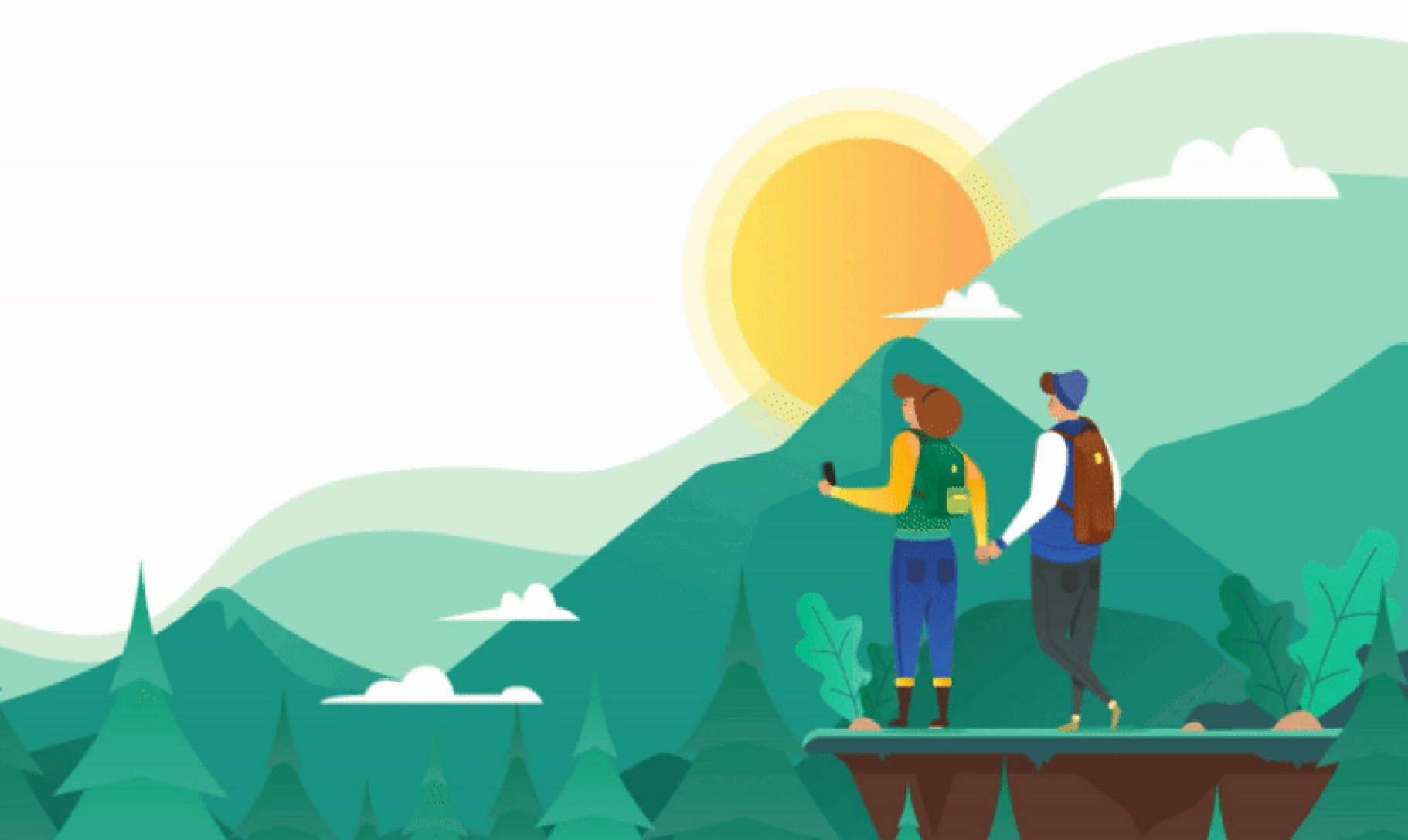
**Software Design Model**

**WildLand一站式营地预约平台**



**团队成员：**

**2252092 毕雅琳**

**2251926 罗诗雨**

**2251654 付宝莹**

**2252954 文彧婕**

目录

[1.系统设计进展概述 3](#_Toc2566)

[1.1逻辑架构优化 4](#_Toc13295)

[1.2技术与物理架构设计 4](#_Toc8004)

[1.3特定接口实现 4](#_Toc1032)

[1.4分析机制 4](#_Toc5228)

[1.5架构机制 5](#_Toc14046)

[1.6用例分析 5](#_Toc1607)

[2. 架构优化 5](#_Toc15612)

[2.1. 平台依赖架构 5](#_Toc3713)

[2.2. 子系统和接口 10](#_Toc1727)

[2.3. 外部接口规范 12](#_Toc28747)

[2.4. 论坛服务接口详述 18](#_Toc13811)

[3. 设计机制 25](#_Toc8989)

[3.1. 数据持久化机制 27](#_Toc32703)

[3.2. 用户访问控制机制 28](#_Toc17387)

[4. 用例实现 34](#_Toc32402)

[4.1. 用户登录 34](#_Toc15100)

[4.2. 营地预约 36](#_Toc11171)

[5. 架构风格与设计决策 38](#_Toc19374)

[5.1. 架构风格 38](#_Toc10875)

[5.2. 关键设计决策 40](#_Toc31869)

[6. 已解决的非功能需求 40](#_Toc10426)

[6.1. 可扩展性 40](#_Toc24243)

[6.2. 安全性 41](#_Toc27372)

[6.3. 高可用性和性能 41](#_Toc23428)

[6.4. 用户体验优化 41](#_Toc10977)

[7. 原型进展 41](#_Toc30345)

[8. 设计模型中的开放问题 42](#_Toc20165)

[8.1. 接口与第三方服务集成的兼容性 42](#_Toc6956)

[8.2. 数据准确性与一致性 42](#_Toc25239)

[8.3. 安全性强化 42](#_Toc17057)

[8.4. 用户体验中的多样化需求 42](#_Toc2832)

[9. 项目自我反思 42](#_Toc5583)

[10. 成员分工 43](#_Toc12379)

## 1.系统设计进展概述

在上一次的报告中，我们已经设计了WildLand一站式露营平台的分析模型，提供了具体用户界面、架构和类图。 基于之前的工作，这次我们确定了项目的微服务架构，将提供具体的技术架构和逻辑架构模型，并为我们的项目设计具体的接口，详细描述部分第三方API、子系统接口、分析与架构机制、用例实现和系统原型设计。我们将重点关注项目的具体设计、实现项目所需的主要技术、以及确定项目将如何实现。以下六个部分分别概括了我们的进展与优化点。

### 1.1逻辑架构优化

##### 我们重新优化了逻辑架构，融合了功能性、安全性等多方面的分析机制，使系统设计更高效、更具模块化特点。新的逻辑架构不仅提升了平台的扩展性与服务分离能力，还进一步保障了数据流通的清晰性和可靠性。

### 1.2技术与物理架构设计

（1）**技术架构：**明确了各服务模块的技术栈选择，包括前端基于 Vue.js 开发，后端采用 .NET 微服务框架，数据库使用 MySQL，并通过 Redis 实现缓存优化。

（2）**物理架构：**设计了分布式部署方案，涵盖容器化服务运行（如 Docker）、负载均衡（如 Nginx），以及云端部署（如 AWS 或阿里云）。

### 1.3特定接口实现

### 为各服务模块制定了明确的 API 接口规范，详细定义了参数类型、返回值类型以及调用方式。同时，还编写了与第三方服务（如短信服务 API、支付网关 API）的集成文档，全面阐述了集成方式和注意事项，确保数据交互的高效性与可靠性。

### 1.4分析机制

（1）**核心功能分析**：深入分析了平台的核心功能，如营地预订流程和用户评价系统，并明确了各个服务模块间的协作关系。通过梳理功能逻辑和模块之间的协作关系，确保系统的功能设计与需求完全对接。

（2）**设计模式选择：**选择了单例模式、策略模式和中介者模式等设计模式，能够为不同的功能模块提供高效的实现方式，并通过优化结构降低系统耦合，提升系统可维护性。

（3）**验证与优化：**通过用例建模、性能测试来验证系统的逻辑一致性、性能瓶颈和数据流通的高效性。确保了系统在各种运行条件下能够平稳高效地工作。

### 1.5架构机制

（1）**持久性机制：**数据存储方面，采用了关系型数据库（如 MySQL）并结合分布式存储方案，确保数据一致性和可用性。

（2）**安全性机制**：设计了多种安全机制，如数据加密、身份验证等，确保系统不受外部攻击和内部威胁的影响。

（3）**用户访问控制：**实现了基于角色的权限管理（RBAC），通过多因素认证、细粒度权限控制等手段，确保了系统对不同用户的访问管理。提高了系统的安全性，防止了潜在的未授权访问和数据泄露。

### 1.6用例分析

结合实际场景，深入分析了用户登录、预定营位等关键用例，确保系统流程与设计目标的一致性。针对每个用例，制定了详细的流程图和交互细节，为后续的开发和测试提供了充分的支持。

## 2. 架构优化

### **2.1. 平台依赖架构**

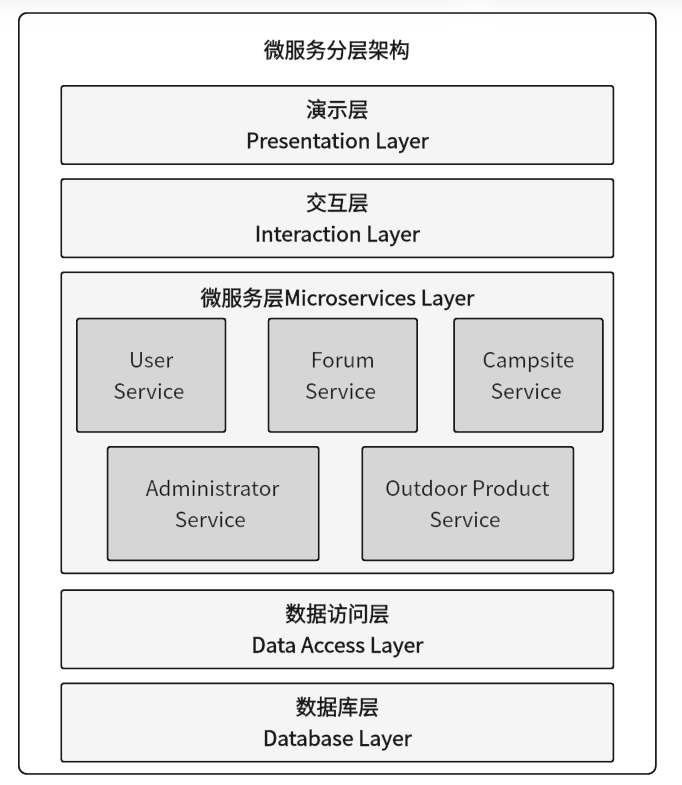
由于在之前的分析中缺乏对某些技术和架构方面的详细考虑，因此将对系统的架构进行了进一步的优化。这样做的目的是改进系统的后续部署，减少耦合，增强可维护性、可伸缩性和灵活性。

本系统之前采用的是通用的三层架构（表现层、中间层、持久层），经过对实际业务需求和技术实现的深入分析，我们决定采用微服务架构来优化和升级系统的架构。

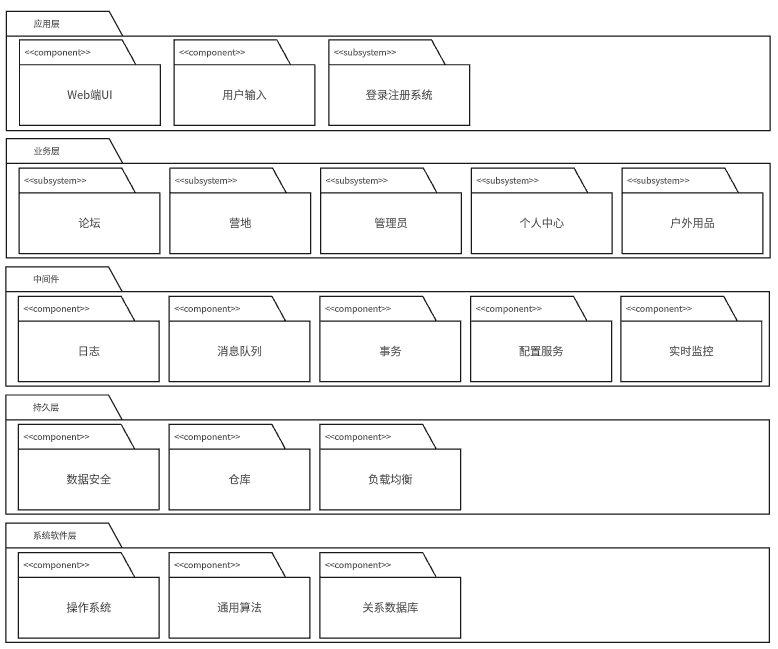
从分层架构过渡到微服务架构的决策基于多方面的考量。首先，Wildland营地交流平台的服务具有相对完整且独立的特性，这使得我们能够将不同的功能模块划分为独立的微服务集群，并供其他系统复用。这种模块化设计不仅提高了服务的可复用性和互操作性，也增强了平台对外部系统的兼容能力。其次，微服务架构具备灵活性和快速迭代的优势。通过独立开发和部署，各微服务可以快速更新和扩展，显著提高了开发的敏捷性，使得我们能够快速响应业务需求的变化。

微服务架构的另一个显著优势在于其高内聚与松耦合的特性。每个微服务专注于特定的业务能力，可以独立设计、部署和扩展。这种松耦合特性确保了对某一微服务的修改不会对整个系统产生影响，从而简化了系统的维护和扩展工作。同时，这种设计方法进一步提升了系统的可靠性和灵活性。

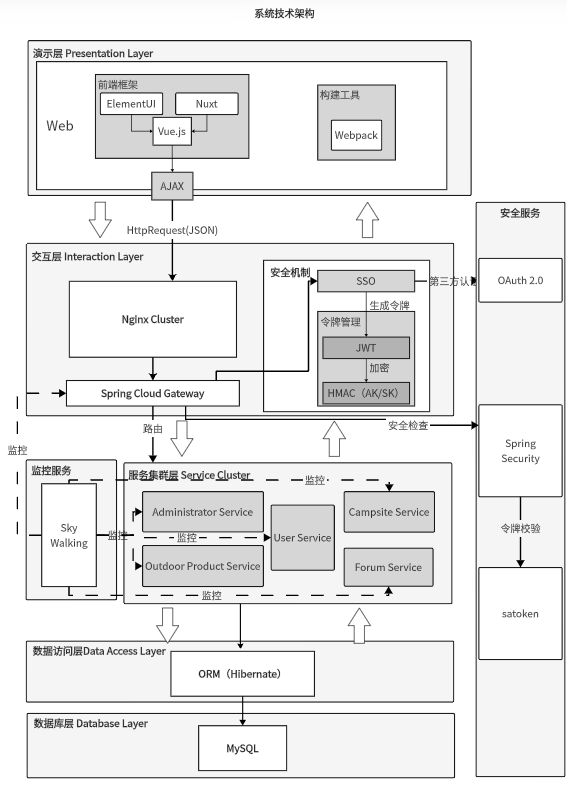
综合微服务架构的优势和传统分层架构的特性，我们对系统架构进行了提炼和优化，形成了以下分层架构：



最终设计的逻辑架构在保持系统高内聚特性和安全性的同时，确保了轻量化和模块化的实现，为平台的未来扩展提供了坚实基础。逻辑架构的示意图如下：



基于上述的逻辑架构，我们采用前后端分离的开发方法，来适应项目规模的微服务架构。前端开发使用Vue框架，后端开发使用Springboot，前后端通过REST API调用和JSON数据文件进行通信。在我们的业务流程中，每一层都会执行业务验证与过滤，以确保账户信息安全、支付营地信息等基本信息的正常状态。结合系统的物理实现，最终实现的技术架构如下：



当前架构的设计实现了以下特性：

**（1）微服务架构特性：**

* 系统的业务逻辑被拆分为多个独立的服务（如 **User Service**、**Forum Service** **、Camptsite Service、Administrator Service、Outdoor Product Service**等），每个服务专注于单一业务功能，遵循高内聚低耦合的原则。
* 微服务架构使系统易于扩展，各服务可以独立部署和扩展，适应高并发和动态业务需求。
* 服务之间通过 **Spring Cloud Gateway** 和 REST 接口协作，路由和调用逻辑清晰。

**（2）强大的安全机制：**

* 系统的业务逻辑被拆分为多个独立的服务（如 User Service、Forum Service 等），每个服务专注于单一业务功能，遵循高内聚低耦合的原则。
* 微服务架构使系统易于扩展，各服务可以独立部署和扩展，适应高并发和动态业务需求。
* 服务间通过 Spring Cloud Gateway 和REST接口协作，路由和调用逻辑清晰。

**（3）高性能和高可用性：**

* 使用 Nginx 集群 进行负载均衡，分担请求流量，提升系统的性能和可靠性。
* Spring Cloud Gateway 进一步增强了网关的路由能力，并支持动态路由、流量控制等功能。
* 微服务层和网关层均支持监控（通过 Sky Walking），帮助及时发现问题和进行优化。

**（4）可观测性与运维能力：**

* 集成了 Sky Walking，对服务性能、调用链路进行全局监控，帮助运维人员快速定位问题。
* 提供了细粒度的监控视图，包括服务调用、响应时间等，适合分布式系统的运维需求。

**（5）开发和部署效率：**

* 使用现代化前端工具（如 Vue.js 和 Webpack），提升了前端开发效率。
* 后端使用 Spring Cloud 提供微服务框架支持，简化了服务的开发和管理。
* 通过分层设计和标准化工具链，前后端分工明确，便于团队协作。

由此，我们综合设计优化了架构，使之具备模块化与可扩展性、高性能和高可用性、安全性、监控和运维能力等优势。

**2.2. 子系统和接口**

在前方的架构设计中，我们将业务逻辑层提供的多种服务打包，划分形成了五个微服务，并设计了相适应的接口，接下来我们就介绍这些子系统与相关接口：

**2.2.1. 用户微服务**

|  |  |  |
| --- | --- | --- |
| **子系统名称** | **接口功能** | **API** |
| **用户管理子系统** | 用户登录 | POST /api/user/login |
| 用户注册 | POST /api/user/register |
| 重置密码 | POST /api/user/resetPassword |
| 编辑个人信息 | POST /api/user/update |
| 获取用户信息 | GET /api/user/getUserInfo |
| 获取用户列表 | GET /api/user/getUser |
| **个人钱包子系统** | 获取钱包余额 | GET /api/wallet/getBalance |
| 钱包充值 | POST /api/wallet/recharge |
| 钱包消费 | POST /api/wallet/consume |
| 退款到账 | POST /api/wallet/refund |
| **个人空间子系统** | 申请商户 | POST /api/user/applyForMerchant |
| 获取用户收藏夹 | GET /api/user/getStarItems |
| 取消用户收藏项 | POST /api/user/cancelStarItem |
| 获取用户订单表 | GET /api/user/getOrders |
| 获取用户消息表 | GET /api/user/getMessages |
| **商户子系统** | 获取户外用品列表 | GET /api/user/getProducts |
| 获取营地列表 | GET /api/user/getCampsites |
| 增加新的营地 | POST /api/user/newCampsite |
| 删除某营地 | DELETE /api/user/deleteCampsite |
| 修改营地信息 | POST /api/user/editCampsite |
| 完成营地退款 | POST /api/user/refundCampsite |
| 增加新的户外用品 | POST /api/user/newProduct |
| 删除某户外用品 | DELETE /api/user/deleteProduct |
| 修改户外用品信息 | POST /api/user/editProduct |
| 完成户外用品退款 | POST /api/user/refundProduct |

**2.2.2. 营地微服务**

|  |  |  |
| --- | --- | --- |
| **子系统名称** | **接口功能** | **API** |
| **营地查看子系统** | 获取营地列表 | GET /api/camp/getCampList |
| 获取营地信息 | GET /api/camp/getCampInfo |
| **营地预约子系统** | 预定营地 | POST /api/camp/reserveCampOrder |
| 支付订单 | POST /api/camp/payForOrder |
| 取消营地预订 | POST /api/camp/cancelCampOrder |

**2.2.3. 论坛微服务**

|  |  |  |
| --- | --- | --- |
| **子系统名称** | **接口功能** | **API** |
| **论坛查看子系统** | 获取帖子预览 | GET /api/post/getPostOverview |
| 获取帖子详情 | GET /api/post/getPostDetail |
| **帖子互动子系统** | 点赞帖子 | POST /api/post/likePost |
| 取消点赞帖子 | POST /api/post/dislikePost |
| 收藏帖子 | POST /api/post/starPost |
| 取消收藏帖子 | POST /api/post/cancelStarPost |
| 举报帖子 | POST /api/post/reportPost |
| **帖子管理子系统** | 发布帖子 | POST /api/post/publishPost |
| 删除帖子 | DELETE /api/post/deletePost |
| 修改帖子可见状态 | POST /api/post/modifyStatus |
| 举报帖子 | DELETE /api/post/reportPost |
| **评论管理子系统** | 评论帖子 | POST /api/post/commentOnPost |
| 回复评论 | POST /api/post/commentOnComment |
| 删除评论 | DELETE /api/post/deleteComment |
| 点赞评论 | POST /api/post/likeComment |
| 取消点赞评论 | POST /api/post/dislikeComment |
| 举报评论 | POST /api/post/reportComment |

**2.2.4. 户外用品微服务**

|  |  |  |
| --- | --- | --- |
| **子系统名称** | **接口功能** | **API** |
| **户外用品查看子系统** | 获取户外用品列表 | GET /api/product/getProductcts |
| 搜索户外用品 | GET /api/product/searchProduct |
| **户外用品操作子系统** | 下单户外用品 | POST /api/product/createProductOrder |
| 取消户外用品订单 | DELETE /api/product/deleteProductOrder |
| 收藏户外用品 | POST /api/product/starProduct |
| 取消收藏户外用品 | POST /api/product/cancelStarProduct |

**2.2.5. 管理员微服务**

|  |  |  |
| --- | --- | --- |
| **子系统名称** | **接口功能** | **API** |
| **管理员登录子系统** | 用户登录 | POST /api/admin/login |
| 重置密码 | POST /api/admin/resetPassword |
| **管理员论坛审核子系统** | 获取待审核帖子 | GET /api/admin/PostsToReview |
| 获取待审核举报 | GET /api/admin/ReportsToReview |
| 获取待审核达人 | GET /api/admin/ExpertsToReview |
| 审核用户发布帖子 | POST /api/admin/reviewPublish |
| 审核用户举报信息 | POST /api/admin/reviewReport |
| 审核户外达人申请 | POST /api/admin/reviewExpert |
| **管理员管理商户子系统** | 获取商户列表 | GET /api/admin/getMerchants |
| 审核商户申请 | POST /api/admin/reviewMerchant |
| 删除商户 | DELETE/api/admin/deleteMerchant |

**2.3. 外部接口规范**

**2.3.1. 支付接口**

**（1）接入背景：**

我们的系统设计要求用户通过钱包余额完成后续交易（如预订营位、租赁户外用品等），钱包支持充值和提现，这些操作需要通过与第三方支付平台的集成来实现。为了保证安全性和规范性，使用 OAuth 2.0 进行用户认证和授权，确保用户的支付操作得到充分保护。

1. **功能要求：**

**充值（支付）功能：**

* 用户通过第三方支付平台（如支付宝、微信支付等）充值钱包余额。
* 支持多种支付方式（银行卡、微信、支付宝等）。
* 在充值完成后，更新钱包余额并通知用户。

**提现功能：**

* 用户将钱包余额提现到绑定的银行卡或第三方支付账户。
* 支持验证用户身份，确保提现账户合法性。
* 记录提现请求及状态（如处理中、成功、失败）。

**安全性：**

* 使用 OAuth 2.0 进行认证，确保交易操作仅限授权用户。
* 支持签名验证和回调接口，防止交易数据被篡改。

1. **接口规范：**

**1、接口地址：**

|  |  |
| --- | --- |
| **充值接口** | POST /api/payment/recharge |
| **提现接口** | POST /api/payment/withdraw |
| **回调接口** | POST /api/payment/callback |

1. **鉴权方式：**

OAuth 2.0 Bearer Token：每次调用接口需在Authorization头中携带用户的Bearer Token。

1. **请求和响应规范：**

**·充值接口（Recharge API）**

|  |  |
| --- | --- |
| URL | /api/payment/recharge |
| 方法 | POST |
| 请求参数 | {  "amount": 100.00, // 充值金额（单位：元）  "payment\_method": "alipay", // 支付方式（alipay、wechat、card）  "currency": "CNY" // 币种  } |
| 响应参数 | {  "transaction\_id": "1234567890", // 充值交易ID  "redirect\_url": "https://payment-gateway.com/pay/1234567890"  // 支付跳转URL  } |

**·提现接口（Withdraw API）**

|  |  |
| --- | --- |
| URL | /api/payment/withdraw |
| 方法 | POST |
| 请求参数 | {  "amount": 50.00, // 提现金额（单位：元）  "account": "621234567890", // 提现账户（银行卡号或第三方账号）  "account\_type": "bank\_card", // 账户类型（bank\_card、alipay、wechat）  "currency": "CNY" // 币种  } |
| 响应参数 | {  "withdraw\_id": "9876543210", // 提现交易ID  "status": "processing" // 提现状态（processing、success、failed）  } |

**·回调接口（Callback API）**

|  |  |
| --- | --- |
| URL | /api/payment/callback |
| 方法 | POST |
| 请求参数 | {  "transaction\_id": "1234567890", // 第三方支付平台交易ID  "status": "success", // 交易状态（success、failed）  "amount": 100.00, // 交易金额  "timestamp": "2024-12-29T12:00:00Z", // 交易完成时间  "signature": "abcdef123456" // 签名  } |
| 响应参数 | {  "status": "ok" // 确认回调成功  } |

1. **签名机制：**

在回调接口中使用 HMAC-SHA256 签名，确保数据完整性，签名生成规则如下：

*signature = HMAC-SHA256(secret\_key, transaction\_id + status + amount + timestamp)*

1. **示例代码：**

|  |  |
| --- | --- |
| **调用充值接口** | **截屏2024-12-29 04.16.01** |
| **调用提现接口** | **截屏2024-12-29 04.16.11** |
| **调用回调接口** | **截屏2024-12-29 04.21.13** |

**2.3.2. 短信服务接口**

**（1）接入背景：**

我们的系统在用户注册或重置密码时，需要调用第三方短信服务接口，向用户的手机号码发送验证码进行身份验证。这是一个典型的短信验证功能，通过集成第三方短信服务实现。为确保安全性和规范性，使用 OAuth 2.0 进行身份认证和授权，保证短信服务调用的合法性。

**（2）功能要求：**

**发送短信验证码：**

* 向用户提供的手机号发送包含验证码的短信，用于注册或重置密码验证。
* 验证码具有时效性（如 5 分钟），超时无效。

**校验短信验证码：**

* 后端校验用户输入的验证码是否与系统发送的验证码一致，并检查验证码是否在有效期内。。

**安全性：**

* 接口调用通过 OAuth 2.0 鉴权，确保只有授权的客户端可调用短信服务。
* 短信内容动态生成，避免敏感信息泄露。

**（3）接口规范：**

**1、接口地址：**

|  |  |
| --- | --- |
| **发送验证码接口** | POST /api/sms/send |
| **验证验证码接口** | POST /api/sms/verify |

**2、鉴权方式：**

OAuth 2.0 Bearer Token：每次调用接口需在请求头中携带Authorization格式为：Authorization: Bearer <access\_token>

**3、请求和响应规范：**

**·发送验证码接口（send API）**

|  |  |
| --- | --- |
| URL | /api/sms/send |
| 方法 | POST |
| 请求参数 | {  "phone\_number": "13800138000", // 用户手机号码  "template\_id": "SMS\_123456", // 短信模板ID（第三方平台配置）  "operation": "register" // 操作类型（如注册：register，重置密码：reset\_password）  } |
| 响应参数 | {  "message\_id": "1234567890", // 短信服务平台的消息ID  "expires\_in": 300 // 验证码有效期（单位：秒）  } |

**·验证验证码接口（verify API）**

|  |  |
| --- | --- |
| URL | /api/sms/verify |
| 方法 | POST |
| 请求参数 | {  "phone\_number": "13800138000", // 用户手机号码  "code": "123456", // 用户输入的验证码  "operation": "register" // 操作类型（如注册：register，重置密码：reset\_password）  } |
| 响应参数 | {  "verified": true, // 验证结果（true：成功，false：失败）  "message": "Verification succeeded" // 返回信息  } |

**（4）示例代码：**

|  |  |
| --- | --- |
| **调用发送**  **验证码接口** | **截屏2024-12-29 05.53.45** |
| **调用验证**  **验证码接口** | **截屏2024-12-29 05.55.01** |

**2.4. 论坛服务接口详述**

**2.4.1 论坛查看子系统**

**（1）获取帖子预览：**

|  |  |
| --- | --- |
| **方法** | **GET** |
| **URL** | **/api/post/getPostOverview** |
| **功能** | **获取帖子列表的预览信息，用于展示论坛中的帖子概要。** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **postkind** | **string** | **是** | **选择预览的帖子类别** | | **userID** | **String** | **是** | **当前登录用户的ID，用于获取与每条帖子的互动状态** | |
| **响应参数** | **{**  **"postOverviews": [**  **{**  **"postId": 123,**  **"title": "Post Title",**  **"summary": "Post Summary",**  **"authorName": "Author Name",**  **"authorAvatar": "Author AvatarUrl",**  **"createdAt": "2024-12-29T12:00:00Z",**  **"likes": 120,**  **"stars": 50,**  **"isLiked": false,**  **"isStarred": true,**  **}**  **],**  **"total": 100, // 总帖子数**  **}** |

**（2）获取帖子详情：**

|  |  |
| --- | --- |
| **方法** | **GET** |
| **URL** | **/api/post/getPostDetail** |
| **功能** | **获取某个帖子的详细信息。** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **postId** | **string** | **是** | **帖子ID** | | **userID** | **String** | **是** | **当前登录用户的ID，用于获取与当前帖子的互动状态** | |
| **响应参数** | **{**  **"postDetail": [**  **{**  **"postId": 123,**  **"title": "Post Title",**  **"content": "Post detailed content",**  **"author": "Author Name",**  **"authorAvatar": "Author AvatarUrl",**  **"pics": [pic1\_url,pic2\_url,pic3\_url,……],**  **"visible\_status": "public",**  **"createdAt": "2024-12-29T12:00:00Z",**  **"likes": 120,**  **"stars": 50,**  **"isLiked": false,**  **"isStarred": true,**  **"comments": [**  **{**  **"commentId":1,**  **"commentAuthorName":"comment Author",**  **"commentAuthorId":"comment Author ID",**  **"content":"this is a comment",**  **"createdAt":"2024-12-29T18:00:00Z",**  **"likes": 13,**  **"isLiked": false,**  **}**  **],**  **}**  **],}** |

**2.4.2 论坛互动子系统**

**（1）点赞帖子：**

|  |  |
| --- | --- |
| **方法** | **POST** |
| **URL** | **/api/post/likePost** |
| **功能** | **给帖子点赞** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **postId** | **string** | **是** | **帖子ID** | | **userID** | **String** | **是** | **当前登录用户的ID** | |
| **响应参数** | **{**  **"status": "success",**  **"likes": 121 // 当前点赞数**  **}** |

**（2）取消点赞帖子：**

|  |  |
| --- | --- |
| **方法** | **POST** |
| **URL** | **/api/post/dislikePost** |
| **功能** | **取消对帖子的点赞** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **postId** | **string** | **是** | **帖子ID** | | **userID** | **String** | **是** | **当前登录用户的ID** | |
| **响应参数** | **{**  **"status": "success",**  **"likes": 120 // 当前点赞数**  **}** |

**（3）收藏帖子：**

|  |  |
| --- | --- |
| **方法** | **POST** |
| **URL** | **/api/post/starPost** |
| **功能** | **收藏帖子** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **postId** | **string** | **是** | **帖子ID** | | **userID** | **String** | **是** | **当前登录用户的ID** | |
| **响应参数** | **{**  **"status": "success"**  **}** |

**（4）取消收藏帖子：**

|  |  |
| --- | --- |
| **方法** | **POST** |
| **URL** | **/api/post/cancelStarPost** |
| **功能** | **取消收藏帖子** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **postId** | **string** | **是** | **帖子ID** | | **userID** | **String** | **是** | **当前登录用户的ID** | |
| **响应参数** | **{**  **"status": "success"**  **}** |

**2.4.3 帖子管理子系统**

**（1）发布帖子：**

|  |  |
| --- | --- |
| **方法** | **POST** |
| **URL** | **/api/post/publishAPost** |
| **功能** | **发布帖子** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **userID** | **string** | **是** | **当前登录用户的ID** | | **title** | **string** | **是** | **帖子标题** | | **content** | **string** | **是** | **帖子内容** | | **picsUrl** | **list** | **否** | **帖子图片** | | **postkind** | **string** | **是** | **帖子类别** | |
| **响应参数** | **{**  **"status": "success",**  **"postId": 123**  **}** |

**（2）删除帖子：**

|  |  |
| --- | --- |
| **方法** | **DELETE** |
| **URL** | **/api/post/deletePost** |
| **功能** | **删除帖子** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **postId** | **string** | **是** | **帖子ID** | |
| **响应参数** | **{**  **"status": "success"**  **}** |

**（3）修改帖子可见状态：**

|  |  |
| --- | --- |
| **方法** | **POST** |
| **URL** | **/api/post/modifyStatus** |
| **功能** | **修改帖子的可见性（公开或私密）** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **postId** | **string** | **是** | **帖子ID** | | **visibleStatus** | **string** | **是** | **帖子可见性** | |
| **响应参数** | **{**  **"status": "success"**  **}** |

**（4）举报帖子：**

|  |  |
| --- | --- |
| **方法** | **POST** |
| **URL** | **/api/post/reportPost** |
| **功能** | **举报帖子** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **postId** | **string** | **是** | **帖子ID** | | **userID** | **string** | **是** | **当前登录用户的ID** | | **reportReason** | **string** | **是** | **举报原因** | |
| **响应参数** | **{**  **"status": "success",**  **}** |

**2.4.4 评论管理子系统**

**（1）评论帖子：**

|  |  |
| --- | --- |
| **方法** | **POST** |
| **URL** | **/api/post/commentOnPost** |
| **功能** | **对帖子进行评论** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **userID** | **string** | **是** | **当前登录用户的ID** | | **postId** | **string** | **是** | **当前帖子ID** | | **content** | **string** | **是** | **评论内容** | |
| **响应参数** | **{**  **"status": "success",**  **"commentId": 1**  **}** |

**（2）回复评论：**

|  |  |
| --- | --- |
| **方法** | **POST** |
| **URL** | **/api/post/commentOnComment** |
| **功能** | **回复某条评论** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **userID** | **string** | **是** | **当前登录用户的ID** | | **commentId** | **string** | **是** | **评论ID** | | **content** | **string** | **是** | **回复内容** | |
| **响应参数** | **{**  **"status": "success",**  **"commentId": 2**  **}** |

**（3）删除评论：**

|  |  |
| --- | --- |
| **方法** | **DELETE** |
| **URL** | **/api/post/deleteComment** |
| **功能** | **删除帖子** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **postId** | **string** | **是** | **帖子ID** | | **commentId** | **string** | **是** | **评论ID** | |
| **响应参数** | **{**  **"status": "success"**  **}** |

**（4）点赞评论：**

|  |  |
| --- | --- |
| **方法** | **POST** |
| **URL** | **/api/post/likeComment** |
| **功能** | **给评论点赞** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **commentId** | **string** | **是** | **评论ID** | | **userID** | **String** | **是** | **当前登录用户的ID** | |
| **响应参数** | **{**  **"status": "success",**  **"likes": 121 // 当前点赞数**  **}** |

**（5）取消点赞评论：**

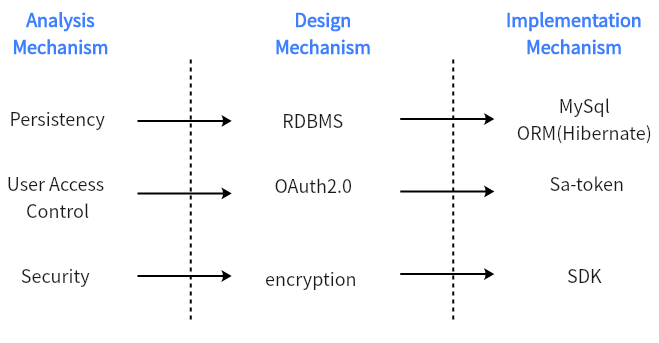
|  |  |
| --- | --- |
| **方法** | **POST** |
| **URL** | **/api/post/dislikeComment** |
| **功能** | **取消对评论的点赞** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **commentId** | **string** | **是** | **评论ID** | | **userID** | **String** | **是** | **当前登录用户的ID** | |
| **响应参数** | **{**  **"status": "success",**  **"likes": 120 // 当前点赞数**  **}** |

**（6）举报评论：**

|  |  |
| --- | --- |
| **方法** | **POST** |
| **URL** | **/api/post/reportComment** |
| **功能** | **举报评论** |
| **请求参数** | |  |  |  |  | | --- | --- | --- | --- | | **参数名** | **类型** | **是否必填** | **描述** | | **postId** | **string** | **是** | **帖子ID** | | **commentId** | **string** | **是** | **评论ID** | | **userID** | **string** | **是** | **当前登录用户的ID** | | **reportReason** | **string** | **是** | **举报原因** | |
| **响应参数** | **{**  **"status": "success",**  **}** |

## **设计机制**

下图展示了系统的设计机制，从分析到实现的逐步细化过程：分析机制提出需求（如持久化、用户访问控制、安全性），设计机制提供技术解决方案（如 RDBMS、OAuth 2.0、加密），实现机制则将设计落地为具体工具或技术（如 MySQL、Sa-token、SDK）。



核心分析类涉及的分析机制和说明如下：

|  |  |  |
| --- | --- | --- |
| 分析类 | 分析机制 | 解释 |
| User | UserAccessContorl  Persistency  Security | * 普通用户登录进入游客界面，拥有普通权限，其信息需要持久化保存在数据库 * 涉及到个人隐私的信息需要加密存储 |
| Administrator | UserAccessContorl  Persistency  Security | * 系统管理员登录进入到管理员界面，拥有较高权限 * 其信息需要持久化保存在数据库 * 涉及到个人隐私的信息需要加密存储 |
| Campground | Persistency | * 营地信息需要持久化保存在数据库中 |
| OutdoorProduct | Persistency | * 户外用品信息需要持久化保存在数据库中 |
| Post | Persistency | * 帖子信息需要持久化保存在数据库中 |
| Order | Persistency  Security | * 订单信息需要持久化保存在数据库 * 涉及到个人隐私的信息需要加密存储 |

### 数据持久化机制

我们的营地预约网站涉及多种类型的数据，包括用户信息、营地数据（如设施类型、可用状态、营位数量等）、交易数据（支付记录、退款记录等）以及用户交互数据（评论、评分、互动记录等）。这些数据构成了系统实现核心功能的基础，包括预约管理、支付处理和用户互动。因此，为了确保数据的安全存储和高效管理，设计一个可靠的数据持久化方案尤为重要。

考虑到系统的数据特点和功能需求，我们选择了 MySQL 作为主要的数据库存储解决方案。MySQL 是一种高效、稳定且成熟的关系型数据库，具备以下优势：

1. 易于扩展：MySQL 支持分布式部署，能够适应系统未来的业务扩展需求。
2. 与 Hibernate 的兼容性：MySQL 与 Hibernate 框架高度兼容，能够充分发挥 Hibernate 的对象关系映射能力。

特别地，对于系统中“营地”这样的非结构化数据（如详细描述、附加服务、特殊规则和各种“是否”类的属性，如是否允许宠物、是否提供Wi-Fi等），我们选择使用 JSON 格式存储。JSON 格式具有高度的灵活性，能够方便地存储键值对和嵌套数据，避免因营地属性数量多且变化频繁导致的表结构复杂化或字段稀疏问题，例如当需要新增或删除营地的某些特性（如添加“是否有露营设备租赁”）时，无需修改数据库表的结构，只需在 JSON 中新增或删除对应的键值对，减少了表结构变更的成本。此外，它支持 MySQL 的 JSON 查询功能，可以高效地进行属性筛选和动态扩展，为系统未来的功能扩展提供了便利。

为了进一步提高开发效率和代码的可维护性，我们在系统中引入了 Hibernate 作为 ORM框架。Hibernate 能够将内存中的对象与 MySQL 数据库中的关系数据自动映射，大大简化了数据操作流程。具体来说，Hibernate 提供了以下优势：

1. 自动化的 SQL 生成：Hibernate 可以根据对象的操作（如保存、更新、删除）自动生成相应的 SQL 语句，避免了开发人员手动编写复杂 SQL 的负担，降低了错误发生的可能性。
2. 数据库独立性：通过 Hibernate，系统的业务逻辑与数据库操作解耦。如果未来需要切换数据库类型，仅需调整配置文件，无需大规模修改代码。
3. 强大的事务支持：Hibernate 提供了 完善的事务管理，能够确保数据的一致性和完整性，特别是在复杂的支付或多步骤操作中尤为重要。

通过结合 MySQL 和 Hibernate，我们的数据访问层能够以高效、稳定的方式存储和管理各类数据。在系统运行过程中，开发人员只需关注业务逻辑，而无需考虑底层的数据存储细节。例如，当用户提交新的预约时，开发者只需创建一个预订对象并设置相关属性，Hibernate 会自动将其转换为 SQL 插入语句，并将数据存储到 MySQL 数据库中。这种架构设计不仅提高了开发效率，还为系统的扩展性和维护性奠定了良好的基础。

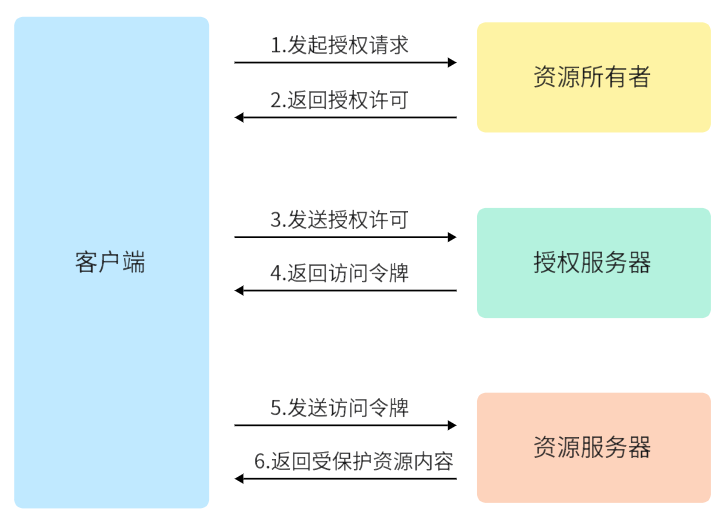
### 用户访问控制机制

在我们的营地预约系统中，用户访问控制是一个至关重要的部分。系统涉及三类主要角色：普通用户、商户和管理员，这些角色拥有不同的功能需求和访问权限。例如，普通用户需要查看营地信息、发起预约和完成支付；商户需要管理营地信息、查看订单和处理预约请求；而管理员则负责对系统的整体运营和维护，包括用户管理、商户审核以及营地信息的审核等。为了确保系统安全性、用户隐私保护以及功能的正常运行，我们需要设计一个全面、高效的用户访问控制机制。通过明确角色权限，不仅可以避免越权访问和非法操作，还能够为每个用户提供个性化的服务体验，同时提升系统的稳定性和安全性。

为了实现高效、安全的用户访问控制，我们采用了多种技术和协议的结合来构建完善的认证与授权机制。系统采用 OAuth 2.0 作为基础的认证授权协议，确保用户可以安全地访问资源，同时支持第三方服务的授权。为实现多系统间的统一认证，我们引入了 SSO，通过一次登录便可访问多个子系统或服务，提升用户体验。认证过程中使用 JWT作为令牌的传输格式，确保令牌的跨平台、轻量化和高效性，同时具备自包含特性，可快速验证用户身份信息。为了进一步增强令牌的安全性，我们通过 HMAC签名对 JWT 进行加密，确保数据在传输中的完整性和防篡改性。整个访问控制机制基于 Sa-Token 框架实现，该框架集成了登录认证、权限认证、会话管理和 OAuth 2.0 支持，为系统提供了灵活、易用的权限控制功能。通过这些技术的协同工作，我们构建了一个安全性高、扩展性强、用户体验优良的访问控制体系。

**OAuth2.0**

OAuth 2.0 是一种开放的授权协议，允许第三方应用在不直接接触用户凭据（如用户名和密码）的情况下，获取用户的授权来访问其受保护的资源。它通过分离客户端与资源服务器的角色，提供了一种更加安全、灵活的授权机制，同时支持多种设备和场景的访问控制。



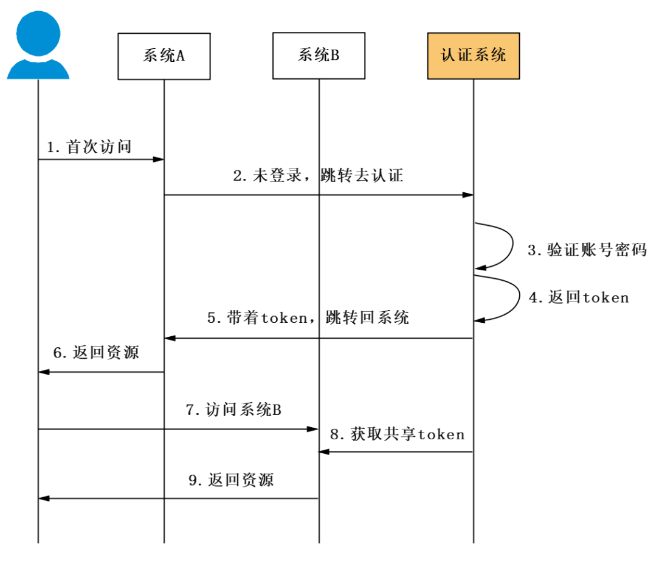
这张图清晰地展示了 OAuth 2.0 的典型授权流程：

1. **发起授权请求：**客户端向资源所有者发起授权请求，用户（资源所有者）决定是否授予客户端访问权限。
2. **返回授权许可：**如果资源所有者同意授权，则返回授权许可（Authorization Grant）给客户端。授权许可是客户端后续获取访问令牌的凭证。
3. **发送授权许可：**客户端将获得的授权许可发送给授权服务器，请求访问令牌（Access Token）。
4. **返回访问令牌：**授权服务器验证授权许可的有效性，如果验证通过，则向客户端颁发访问令牌。
5. **发送访问令牌：**客户端携带访问令牌，向资源服务器请求访问受保护的资源。
6. **返回受保护资源内容：**资源服务器验证访问令牌的有效性，如果令牌有效，则返回受保护资源内容给客户端。

这一流程确保了资源的安全访问，同时避免了用户凭据直接暴露给客户端，提高了系统的安全性和灵活性。

**SSO**

单点登录（SSO，Single Sign-On）是一种认证机制，允许用户在多个关联系统中使用一次登录操作即可访问所有系统，而无需在每个系统中重复登录。同理用户只需注销一次，就可以从多个应用系统退出登录。。SSO 的核心是通过集中式的认证服务，将用户的登录状态在不同系统之间共享，从而提升用户体验和操作效率，同时减少多次登录的安全风险。



单点登录的实现原理说明如下：

1. 用户首次访问系统A时，需要进行登录。
2. 系统A带着用户登录信息重定向给认证系统。
3. 认证系统验证用户登录信息。
4. 验证通过后，返回一个token（Tip：token类似一种内部的通行证，包含了用户身份信息、登录状态和过期时间，在各个系统间共享。）
5. 认证系统带着token重定向给系统A，得知用户是已登录状态。
6. 系统A向用户返回请求的资源。
7. 用户访问系统B时，需要进行登录。
8. 系统B通过共享的token，得知用户是已登录状态。
9. 系统B向用户返回请求的资源。

**JWT与HMAC**

JWT（JSON Web Token） 和 HMAC（基于密钥的消息认证码） 通常一起使用来保证数据传输的安全性和完整性。

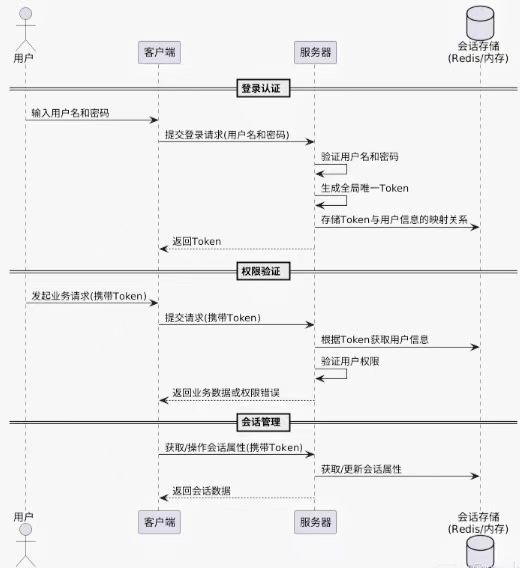
JWT 是一种轻量化的令牌格式，包含三部分：头部（Header）、载荷（Payload）和签名（Signature）。HMAC 则用于生成 JWT 的签名部分，以确保令牌在传输过程中不会被篡改。

JWT 在生成时，通过 HMAC 使用一个共享密钥（Secret Key）对头部和载荷进行哈希计算，生成签名。这个签名是令牌的一部分，用来验证令牌的完整性。生成的 JWT 包含签名后被发送到客户端或其他服务。接收方（如服务器）使用相同的共享密钥和 HMAC 算法重新计算签名，将计算结果与 JWT 中的签名部分对比。如果匹配，则证明令牌未被篡改且可信。

**Sa-Token**

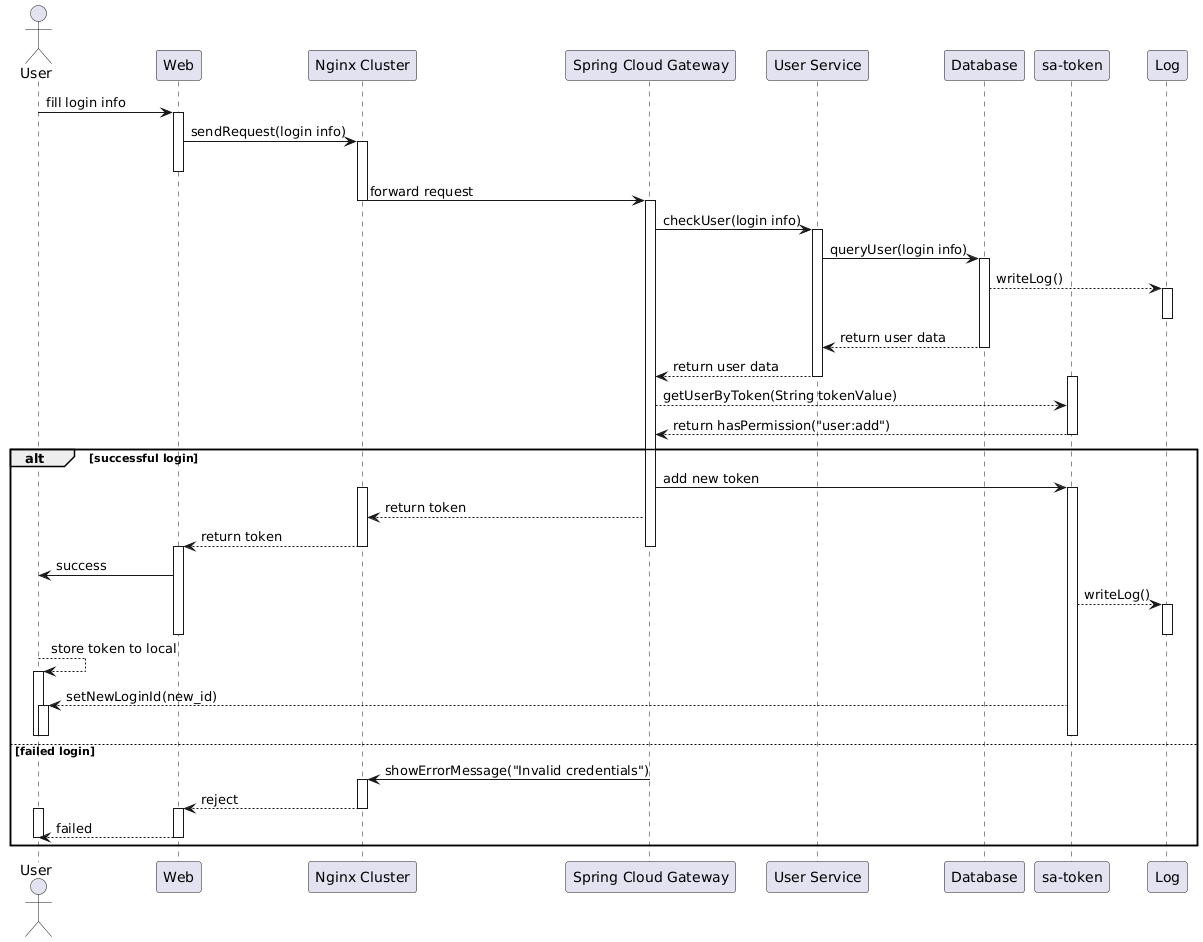
在我们的营地预约系统中，Sa-Token 是核心的权限认证框架，它简化了用户身份验证和权限控制流程，确保不同角色（普通用户、商户、管理员）能够安全地访问其对应的功能模块。结合 OAuth 2.0 和 SSO，Sa-Token 实现了统一的认证机制，用户只需登录一次，即可访问多个子系统。同时，系统使用 JWT 作为令牌的传输格式，并通过 HMAC 签名增强令牌安全性，确保数据传输的完整性和防篡改性。Sa-Token 的高效会话管理能力，使我们的系统既安全可靠，又能提供流畅的用户体验。

下图是Sa-Token的基本工作流程：

****

## 用例实现

### 用户登录



登录是所有用户通过合法账户访问系统的重要功能，其具体流程在上方的序列图中进行了展示。

根据系统设计，登录请求由用户通过前端页面输入信息发起，信息通过 Nginx 集群转发到 Spring Cloud Gateway。后端网关接收到请求后，将其传递给用户服务（User Service），并通过数据库查询账户的合法性。在安全认证方面，系统使用了轻量级认证框架 Sa-Token，不仅对用户账户的登录权限进行检查，还对登录状态进行持续监控。

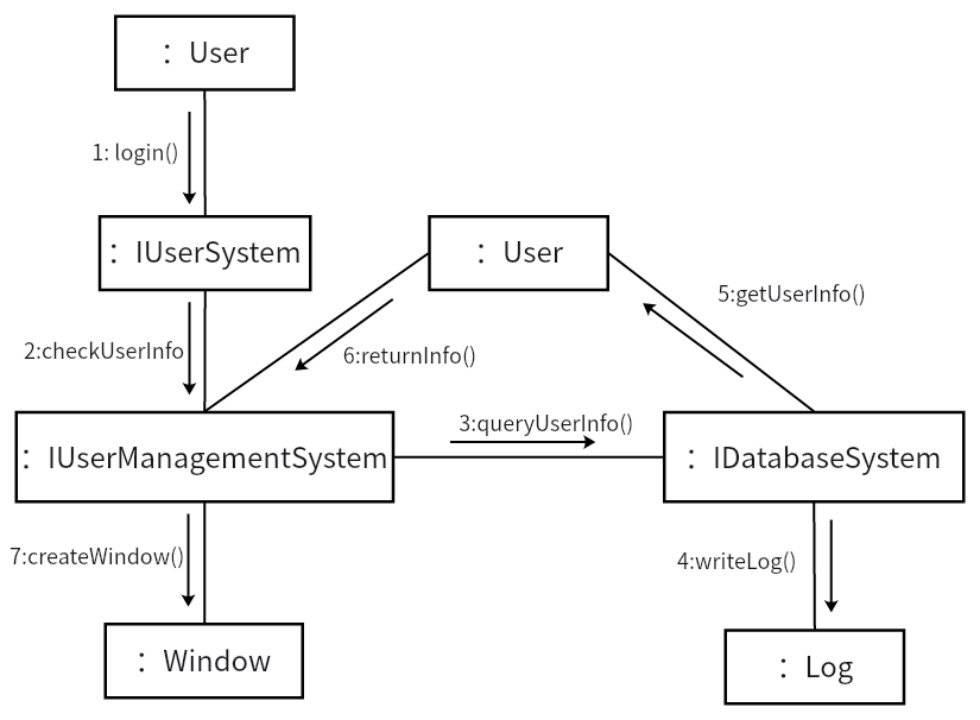
用户首先在登录页面输入用户名和密码。前端页面通过 HTTP 请求将登录信息发送至 Nginx 集群。随后，Nginx 将请求转发给后端的 Spring Cloud Gateway。网关收到请求后调用用户服务的 checkUser(login info) 方法，用户服务通过 REST 接口将登录信息转发给数据库，数据库会查询用户实体以验证用户名和密码的正确性。如果数据库查询到有效用户信息，则返回用户数据，并记录此次操作到日志系统中。

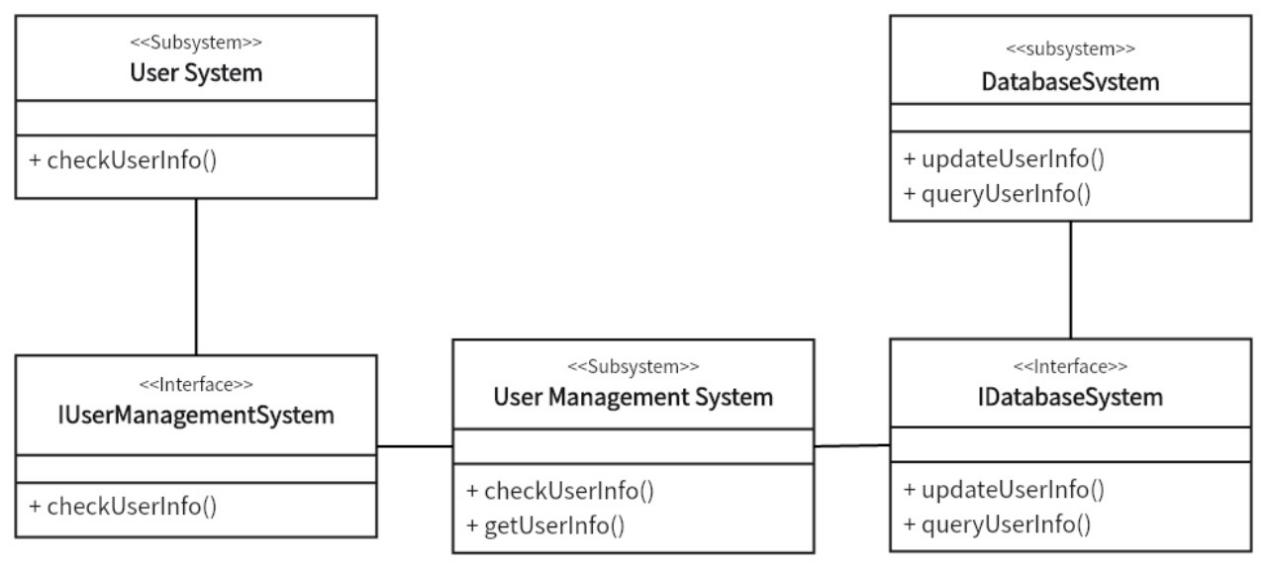
一旦用户信息被验证为正确，用户服务会调用 Sa-Token 的 getUserByToken 方法，检查用户令牌的合法性，并判断用户是否具备登录权限（如是否被列入黑名单）。若用户权限合法，系统将生成一个新的登录令牌，并通过 addNewToken 方法将其添加到会话中。生成的令牌将通过网关返回给前端页面，同时系统会将登录状态更新到本地存储中，确保用户能够持续访问系统。

如果用户输入的信息不正确或用户权限验证失败（例如用户名或密码错误，或者用户被禁用），系统会通过前端页面显示错误提示信息，例如“Invalid credentials”，并拒绝本次登录请求。无论成功或失败，系统都会通过日志服务记录此次登录操作。

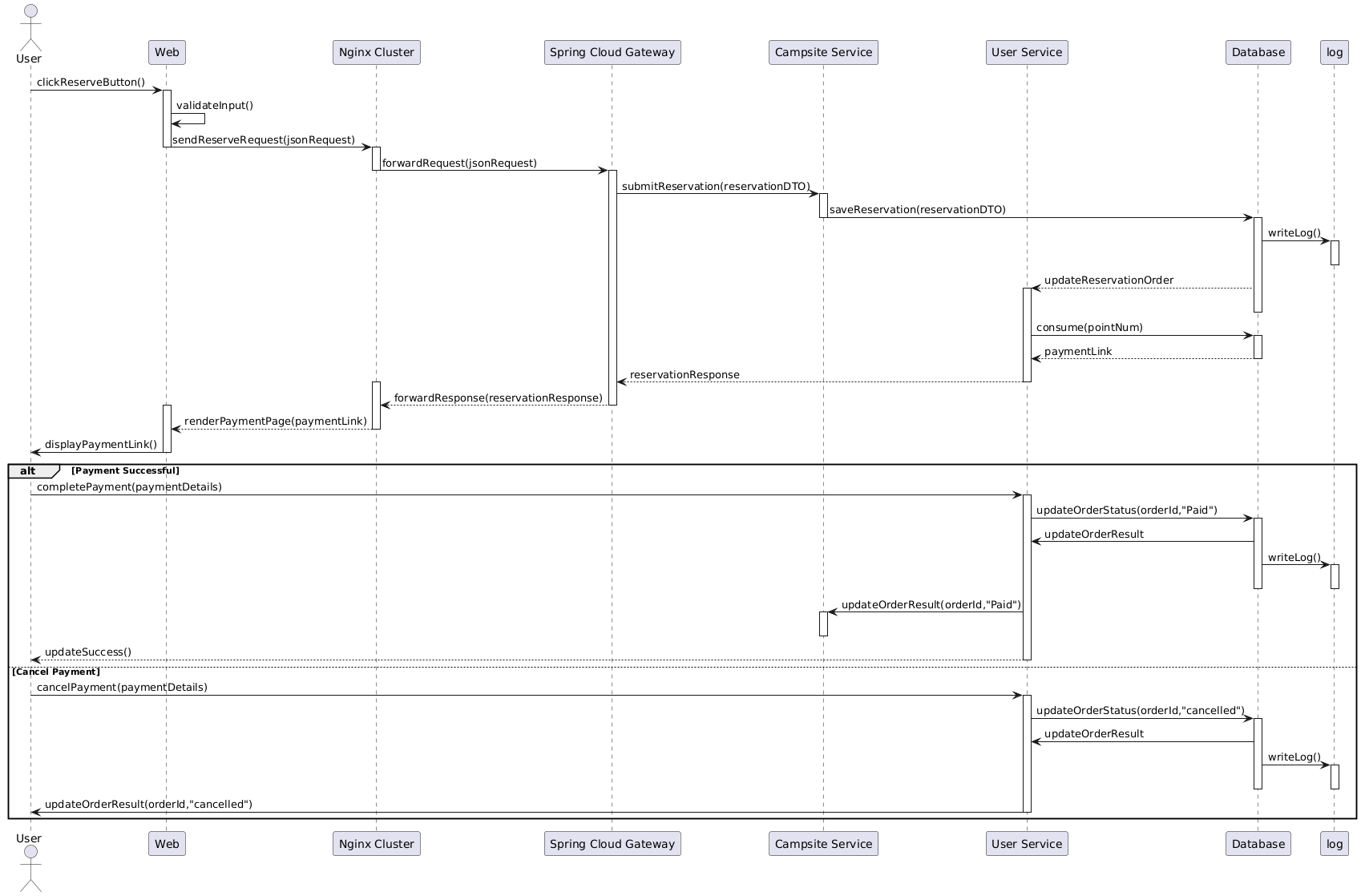
通过以上流程，系统确保了用户身份的安全验证与权限管理，并通过 Sa-Token 框架持续监控用户的登录状态，在发生系统故障或其他异常时能够有效处理登录问题。

用户登录用例的通信图和类图如下：





### 营地预约

****

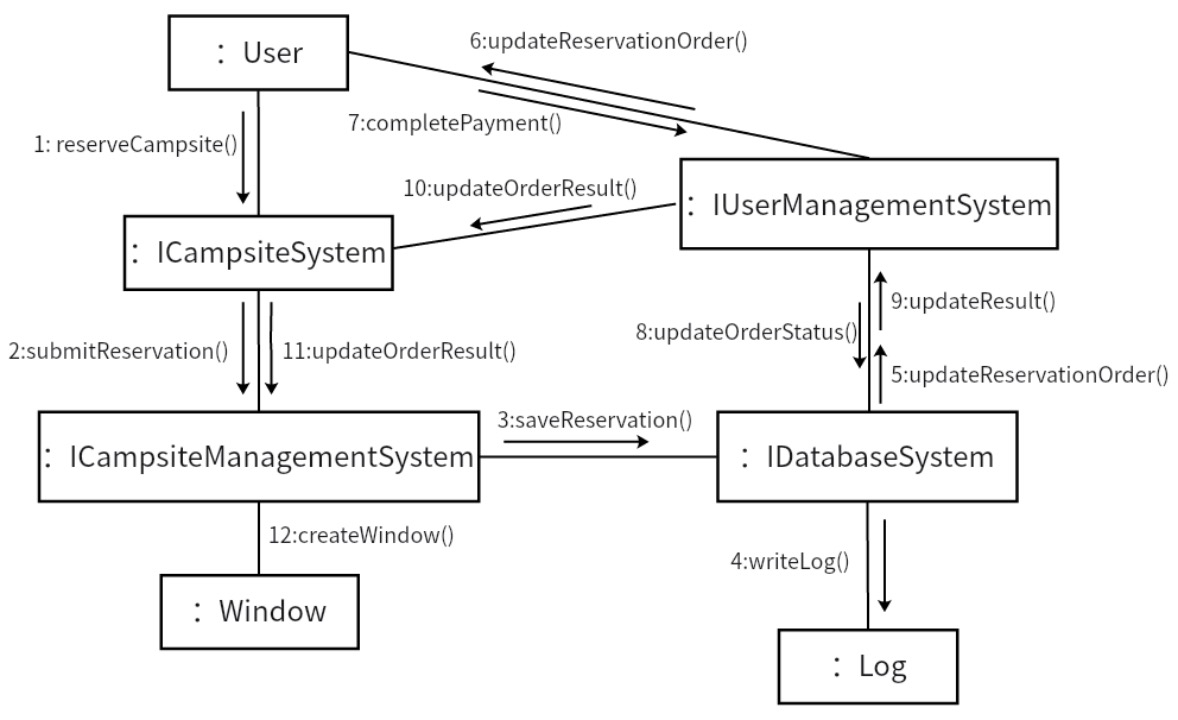
营地预约流程是用户通过系统实现营地预订和支付的重要功能，其具体流程在上方的序列图中进行了展示。

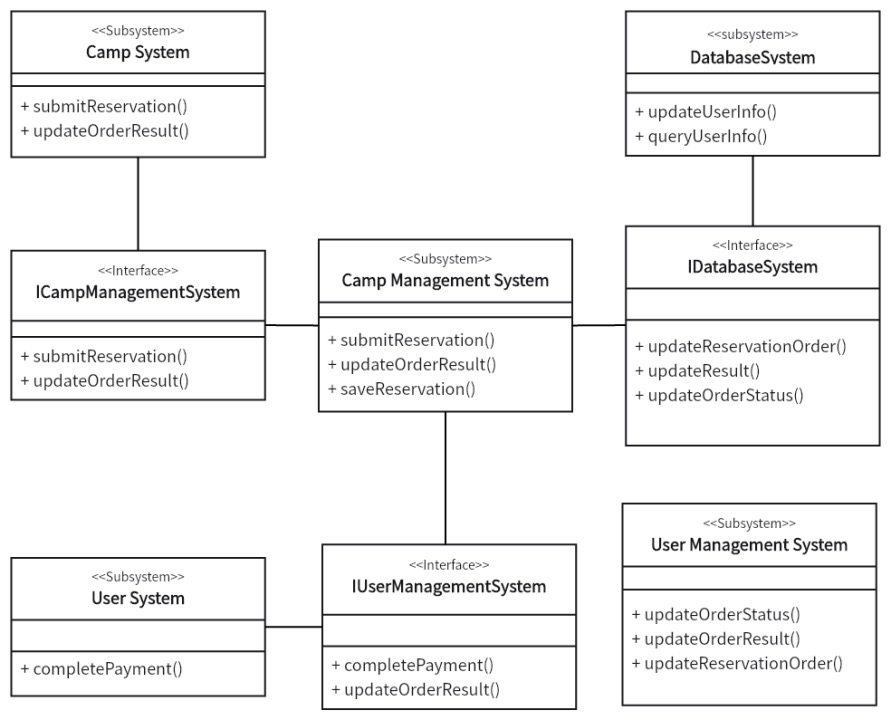
根据系统设计，用户点击“预约按钮”以发起预订请求。前端页面首先验证用户输入的预约信息是否正确。验证通过后，系统将预约信息封装成 JSON 请求，通过 Nginx 集群转发至 Spring Cloud Gateway。后端网关接收到请求后，将其传递至 Campsite Service，负责处理预约请求。Campsite Service 保存预约信息到数据库，同时调用 User Service 扣除用户的钱包中的余额并生成支付链接（此处采取扣除用户在钱包中充值的点数，而不接入第三方交易平台API，可以减少调用外部支付网关的次数，降低支付延迟，也能减少调用外部支付网关的手续费）。

数据库在接收到预约信息后，记录相关操作到日志系统中。预约处理完成后，Campsite Service 返回预约响应信息至 Spring Cloud Gateway，随后通过 Nginx 转发至前端页面，向用户展示支付页面或支付链接。

如果用户完成支付操作，系统会进一步更新订单状态。支付完成后，前端将支付详情提交给网关，网关将支付信息转发至 User Service。User Service 验证支付详情，并更新订单状态为“已支付”，同时记录支付操作至日志系统。如果支付失败或用户取消支付，系统则将订单状态更新为“已取消”，并记录取消操作。

通过以上流程，系统实现了从预约发起到支付完成的完整闭环，包括数据验证、预约保存、支付生成和订单状态管理。系统设计确保了高效的数据流转、安全的支付流程以及操作的可追踪性。

营地预约用例的通信图和类图如下：



## 架构风格与设计决策

### 架构风格

在项目中，我们采用了微服务架构风格来构建整个系统。微服务架构是一种将大型应用拆分为一组小型自治服务的架构方式，这些服务协同工作以满足业务需求。

##### 微服务架构特点

（1）单一职责：每个微服务专注于解决特定的业务问题，确保职责清晰。

（2）松耦合：服务之间通过明确的接口通信，保持相对独立性。

（3）分布式数据管理：每个微服务拥有独立的数据存储，微服务可以根据需求选择最合适的数据库或存储方案。

（4）基础设施自动化：依赖自动化部署、监控和扩展，能更好应对系统复杂性。

##### 5.1.2. 微服务架构优势

（1）弹性与可扩展性：支持对每个服务进行独立的按需扩展，无需扩展整个系统。这使得系统更具弹性，能够应对高并发和流量波动，并且支持水平扩展，确保高负载下的稳定运行。

（2）独立开发与部署：每个微服务都是自治的，能够由独立的团队进行开发、测试和部署。这种解耦特性提高了开发效率，使团队能够独立迭代和发布服务，避免了整体停机的风险。

（3）技术多样性：微服务架构允许不同服务使用不同的技术栈和编程语言，团队可以根据具体需求选择最适合的技术方案，从而激发创造力和提高生产力。

（4）高可用性与容错性：通过将系统拆分为多个服务，确保了单一服务的故障不会影响整个系统的可用性。服务之间清晰的接口设计有助于实现超时、故障切换等容错机制。

**5.1.3. 微服务适用场景**

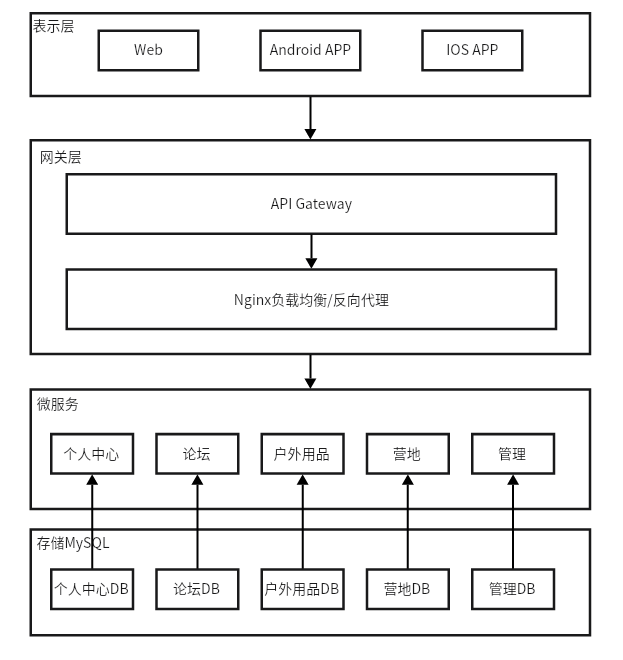
（1）需要快速迭代和发布新功能的项目。

（2）支持多平台、多设备的系统。

（3）需要应对业务需求变化并支持扩展的环境。

（4）希望提升团队独立性和协作效率的项目。

**5.1.4. 微服务分层架构**



### 关键设计决策

* + 1. **平台选择**

我们通过完整地评估流程来选择最适合的平台，以满足我们的系统需求。经过仔细考虑，我们选择了VSCode及其相关扩展来支持我们的开发和部署需求。这个平台与我们的目标操作系统、编程语言和云服务提供商提供了强大的兼容性，使我们能够实现可扩展性、性能和可用性目标。

* + 1. **数据存储解决方案**

为了满足我们的数据存储需求，我们对不同的数据库选择进行评估。经过仔细考虑，我们决定使用Mysql数据库管理系统。这个选择使我们能够高效地存储和管理数据，确保了我们系统所需的可扩展性、数据一致性和查询性能。

##### 安全策略

我们通过实施强大的安全策略来优先考虑系统的安全性和用户数据。为了实现这一点，我们采用了OAuth 2.0安全框架。这个框架提供了一套全面的认证机制、访问控制策略、数据加密方法和漏洞管理程序。

##### 用户界面设计

我们通过专注于直观和视觉上吸引人的用户界面来优先考虑用户体验。利用Vue3框架和ElementPlus组件库，我们创建了与用户工作流程、响应性、可访问性和视觉美学相一致的用户界面。

##### 环境和资源管理

Docker和Kubernetes的结合使用，优化了容器化应用的部署和管理，提升了资源利用率和应用的可移植性，增强了系统的可扩展性和弹性。

##### 数据备份与恢复

为了保障数据的安全性和可恢复性，我们实现了定期数据备份和灾难恢复策略。通过采用分布式存储和备份机制（如 AWS S3 和 数据库快照），确保了在发生硬件故障或系统崩溃时能够快速恢复数据，并最大限度地减少数据丢失。

## 已解决的非功能需求

### 可扩展性

我们的系统需要处理逐渐增长的用户量和多样化的业务需求，为确保系统能够在高并发情况下稳定运行，必须具有良好的可扩展性。 因此我们通过采用微服务架构，将业务逻辑拆分为多个独立的服务模块（如用户服务、论坛服务、营地服务等），每个模块均可以独立部署和扩展。使用Spring Cloud Gateway和Nginx进行负载均衡，支持动态路由和流量控制。通过分布式数据库（如MySQL）和Redis缓存机制，优化数据存储和查询性能，提高系统的弹性和处理能力。

### 安全性

我们的平台涉及用户敏感信息（如登录凭据、支付信息等），需要防止未授权访问和数据泄露。系统通过实施OAuth 2.0认证协议，确保用户授权过程的安全性；采用JWT令牌机制，实现用户身份的轻量化认证和访问控制。同时通过HMAC签名对JWT令牌进行加密，防止数据篡改。系统还引入多因子身份验证（如短信验证）和基于角色的权限管理（RBAC），进一步提升系统安全性。

### 高可用性和性能

平台需要确保在任何时候都能为用户提供快速响应和稳定的服务，尤其是在高流量时刻（如节假日预约高峰）。因此，使用Nginx集群实现负载均衡，分担服务器请求压力；在微服务层中集成Spring Cloud Gateway，用于管理服务的动态扩展。监控工具（如Sky Walking）被用来跟踪服务调用链路和性能指标，确保快速发现并解决性能瓶颈问题。

### 用户体验优化

良好的用户体验是平台成功的关键之一，用户需要直观、高效的交互界面。我们使用Vue.js框架开发前端界面，结合ElementPlus组件库，设计出美观、响应迅速且直观的用户界面。优化了用户登录、预约营地和支付流程，提供清晰的操作引导，减少用户学习成本。

## 原型进展

## 设计模型中的开放问题

### 接口与第三方服务集成的兼容性

**- 如何确保与不同第三方服务（如支付网关、短信服务）的稳定集成？**

· 需进一步验证API调用的兼容性和健壮性，尤其是跨区域或特殊网络环境下的调用可靠性。

### 数据准确性与一致性

**- 在高并发情况下，如何确保用户数据（如预约状态、支付状态）的一致性？**

• 需要对分布式数据库的事务机制进行优化，同时明确如何处理数据延迟或一致性问题。

### 安全性强化

**- 如何更好地防御分布式拒绝服务（DDoS）攻击？**

• 需要进一步评估当前安全策略的覆盖范围，以及是否引入更高级的威胁检测机制。

### 用户体验中的多样化需求

**- 针对不同用户群体（普通用户、商户、管理员），如何进一步优化界面和功能？**

• 需要进行更多用户研究和测试，以平衡多角色的需求差异。

## 项目自我反思

**2251654 付宝莹：**

完成系统设计与分析后，我感到非常自豪。这是一个从无到有的过程，我深入理解了系统架构和设计模式，尤其是在微服务架构的拆分和安全机制的设计中，学到了模块化的重要性，同时也认识到如何在满足复杂需求的同时保持系统简洁。

在过程中也遇到一些挑战，比如需求和实现的平衡、技术选型的犹豫以及时间管理的不足。为了保证功能完善，我们花费了较多时间在细节上，导致部分模块推进稍显迟缓。这让我意识到，合理分配时间和阶段性成果验收非常重要。

在设计过程中我收获了系统性思维能力的提升，能够更全面地考虑模块之间的关系。同时，团队协作让我认识到多视角交流的重要性，用户导向的设计让我更加注重实际需求。未来，我会加强时间管理，更注重用户反馈，并持续学习技术趋势。

**2252954 文彧婕：**

完成系统设计后，我对自己的成长有了更深的认识。这次任务让我理解到软件设计不仅仅是技术实现，而是一种以用户需求为核心的创造性工作。通过系统架构的分层设计、微服务拆分和接口设计，我学会了如何在复杂的需求中寻求最优解，同时确保系统的灵活性和可扩展性。

最大的收获在于对整体设计思维的提升。特别是在考虑系统性能、安全性和用户体验时，我开始能够站在更高的视角平衡技术实现和用户需求。同时，团队的讨论让我认识到协作和交流的重要性，不同的观点能为设计提供更多可能性。

**2251926罗诗雨：**

这次系统分析与设计的作业让我更加清晰地理解了从需求分析到系统设计的整体流程。通过对架构、技术选型和安全策略的分析，我不仅加深了对设计理论的理解，还意识到在实际应用中必须权衡多方面的因素。虽然过程中遇到了一些挑战，但这些经验对我未来从事系统设计工作具有很大的帮助。我相信，通过不断学习和实践，我能在未来的项目中做出更加优化和成熟的设计决策。

## 成员分工

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | 姓名 | 任务 | 占比 |
| 2252092 | 毕雅琳 | 原型进展、开放问题 | 100% |
| 2251654 | 付宝莹 | 架构优化、非功能需求、终期汇报 | 100% |
| 2251926 | 罗诗雨 | 系统设计进展概述、架构风格与设计决策 | 100% |
| 2252954 | 文彧婕 | 设计机制、用例实现、汇报PPT制作 | 100% |