架构设计

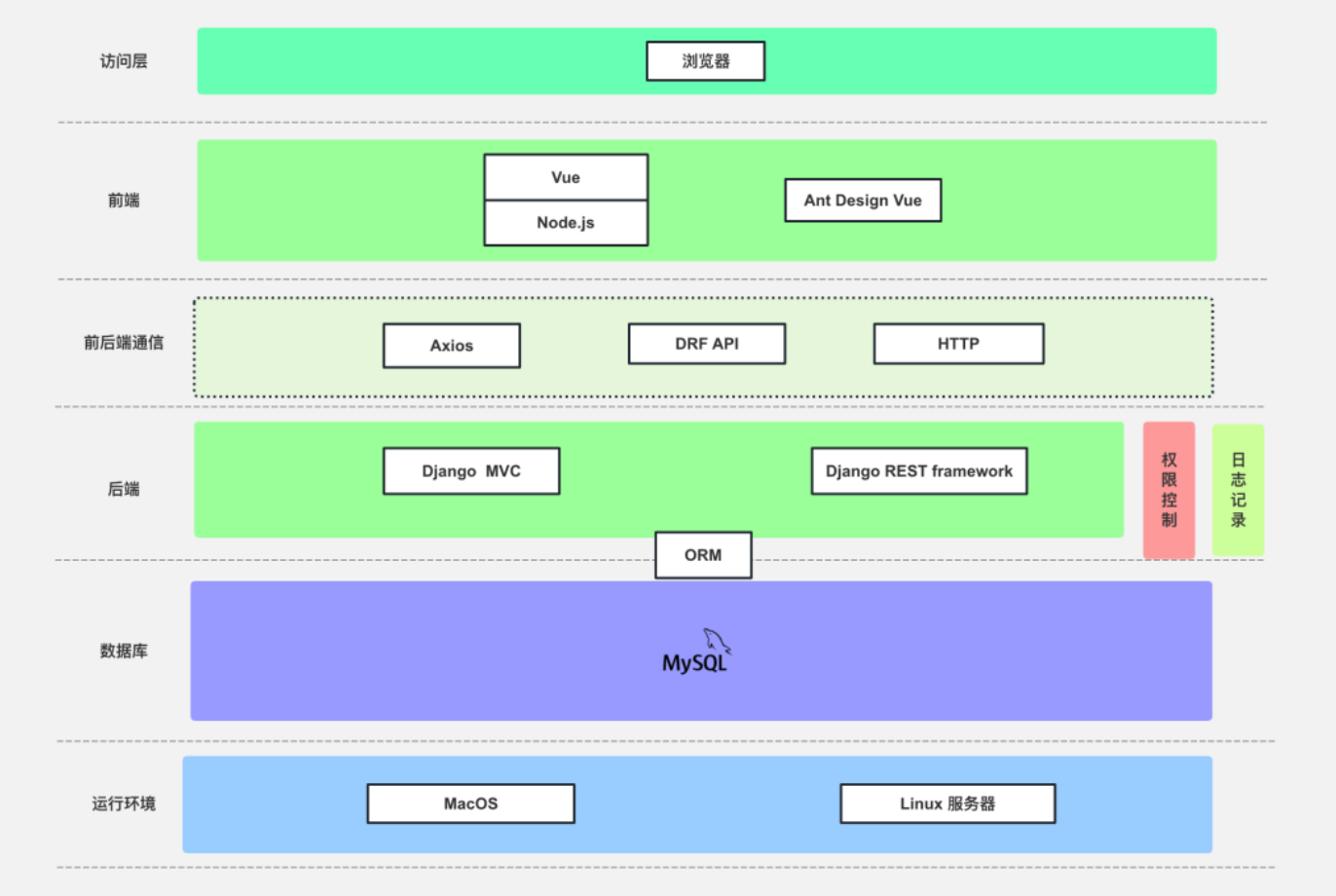
系统架构设计文档

酒店管理系统采用B/S模式，即浏览器/服务器模式。这种设计模式使得酒店管理系统的开发更加安全、高效和快捷。传统的管理模式基于手工处理，管理效率低下，并且无法满足当前用户的需求。随着信息化时代的来临，酒店管理系统的开发成为必然趋势。

技术栈

- **前端技术栈：** 基于 Node.js 和 Vue 框架实现，运用了主流的 Ant Design UI 框架。 Vue.js 作为前端主要的 JavaScript 框架，提供了组件化的开发模式，使得前端代码更易于维护和拓展。 Ant Design Vue 框架该框架提供了一套优雅、美观、且高度可定制的 UI 组件，使得前端界面设计更加高效和符合设计规范。
- **前后端通信：**前端使用 Axios 发起 HTTP 请求，后端 Django REST framework 匹配前端请求执行操作，通过序列化器将 API 请求转换为 JSON 格式并返回给前端。 Vue 前端通过 HTTP 请求调用后端提供的 RESTful API，使用 Django REST framework 处理这些请求。这种方式使得前后端能够独立开发，通过定义清晰的接口规范，实现彼此之间的松耦合。数据以 JSON 格式在前后端之间交换。前端发送请求，后端返回 JSON 格式的数据，实现数据的有效传递和解析。
- **后端技术栈：**基于 Django 框架实现， Django 通过 Model-View-Controller 的 MVC 模式处理请求，并通过 ORM 与数据库通信，通过 DRF 返回 API 请求。作为后端主要的 Web 框架， Django 提供了强大的开发框架和一系列的工具，用于简化开发过程，包括 ORM（Object-Relational Mapping）、路由、模板引擎等。 Django REST framework 提供了一种灵活、强大且易于使用的方式，用于处理Web 请求和响应，使得前后端之间的通信更加简便。
- **数据库：**基于 MySQL，存储关系模式和数据。作为关系型数据库管理系统， MySQL 提供了可靠的数据存储和检索机制。通过 Django 的 ORM，可以轻松地在后端与数据库之间建立映射，实现数据的持久化存储。

系统架构图



模块/子系统设计

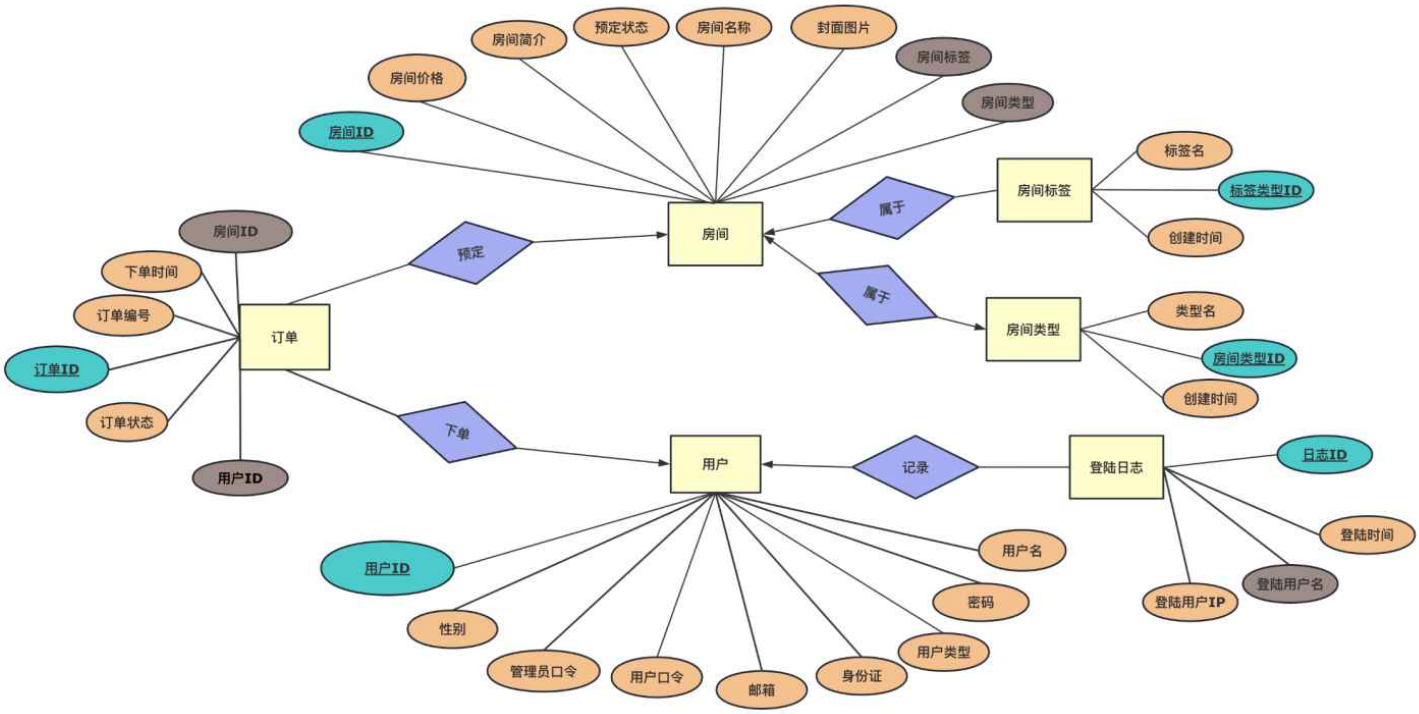
由于本项目前后端分离、是高度解耦的，故将项目分解为数据库设计、前端设计与开发、后端设计与开发这三个模块进行设计。

具体设计见核心模块设计文档。

核心模块设计文档

数据库部分

- 六个实体集：订单、房间、用户、房间标签、房间类型、登陆日志。
- 五个关系集：订单和房间实体的预定关系、订单和用户实体的下单关系、房间和房间标签实体间的属于关系、房间和房间类型实体间的属于关系、登陆日志和用户间的记录关系。



ER图

在实际使用 Django 框架时，可以在创建数据库 models 定义了数据库表的结构后，通过运行 makemigrations 和 migrate 命令自动根据模型创建相应的数据库表。

前端部分

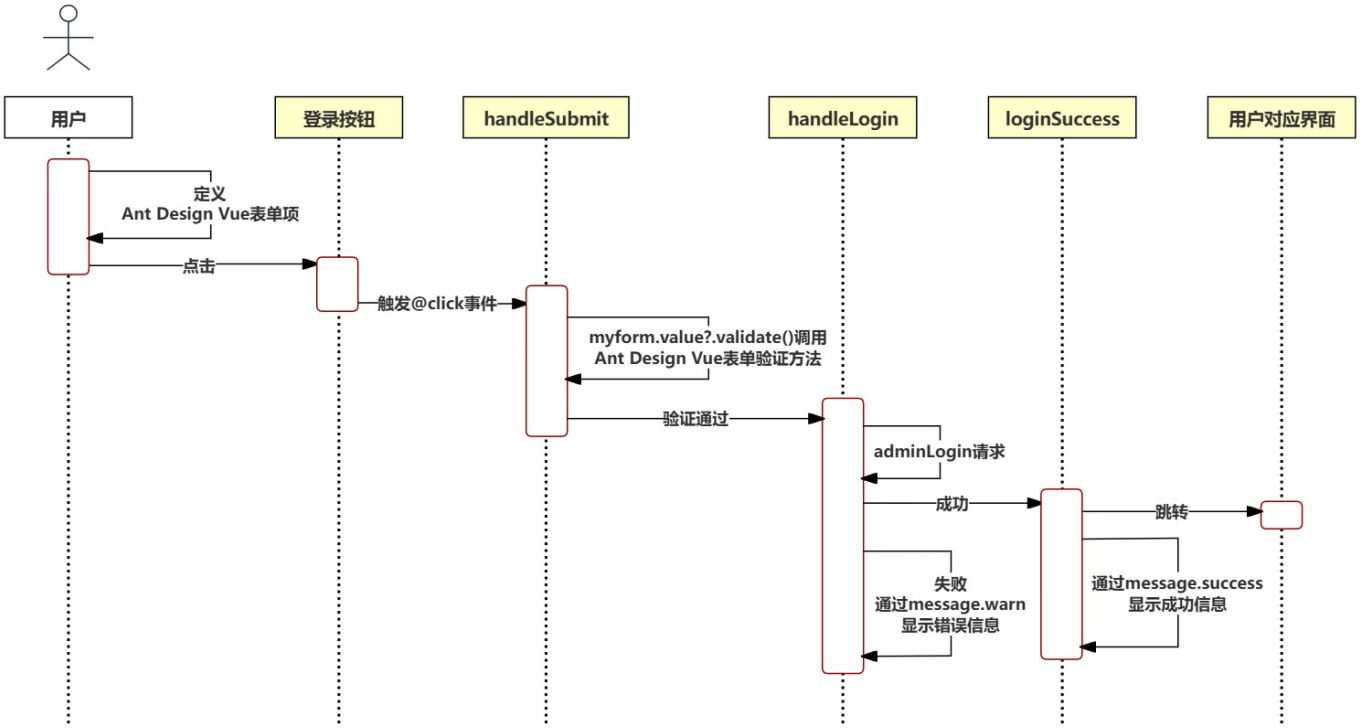
在 Vue 框架下分为三个模块进行开发。

1.用户登陆功能模块

登陆模块的设计目标是提供简洁、直观的登录界面。可以通过 Ant Design 组件库提供的组件实现用户友好的输入和交互。为确保用户输入的安全性，采用安全的密码传输和存储方式。

主要的页面结构分为顶部横栏、主体容器部分。顶部横栏包含酒店图标和登陆系统名称。主体容器包含背景图片，中间有管理员登录表单。

在主体的登陆表单中提供账号和密码输入框，采用 Ant Design 的 a-input 组件。输入框具有清晰的 placeholder 提示。使用 a-form 进行表单验证，确保数据完整性。登录按钮使用 a-button 组件，提供 loading 状态，防止重复点击。



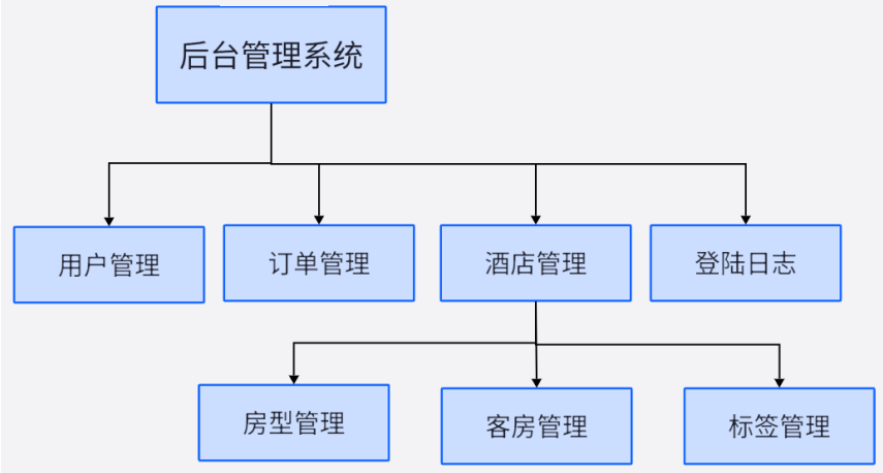
用户登录UML时序图

2.后台管理系统模块

本项目的后台管理系统的前端页面基于 Ant Design Vue 实现。 Ant Design Vue 是 Ant Design设计语言在 Vue.js 框架下的实现，是一套由蚂蚁金服团队开发的丰富的企业级 UI 组件库。

Vue 部件名称	Vue 部件用法
main.vue	主页模块
order.vue	订单模块
loginLog.vue	登录日志模块
classification.vue	分类模块
tag.vue	标签模块
user.vue	用户模块
room.vue	房间模块

主要vue部件

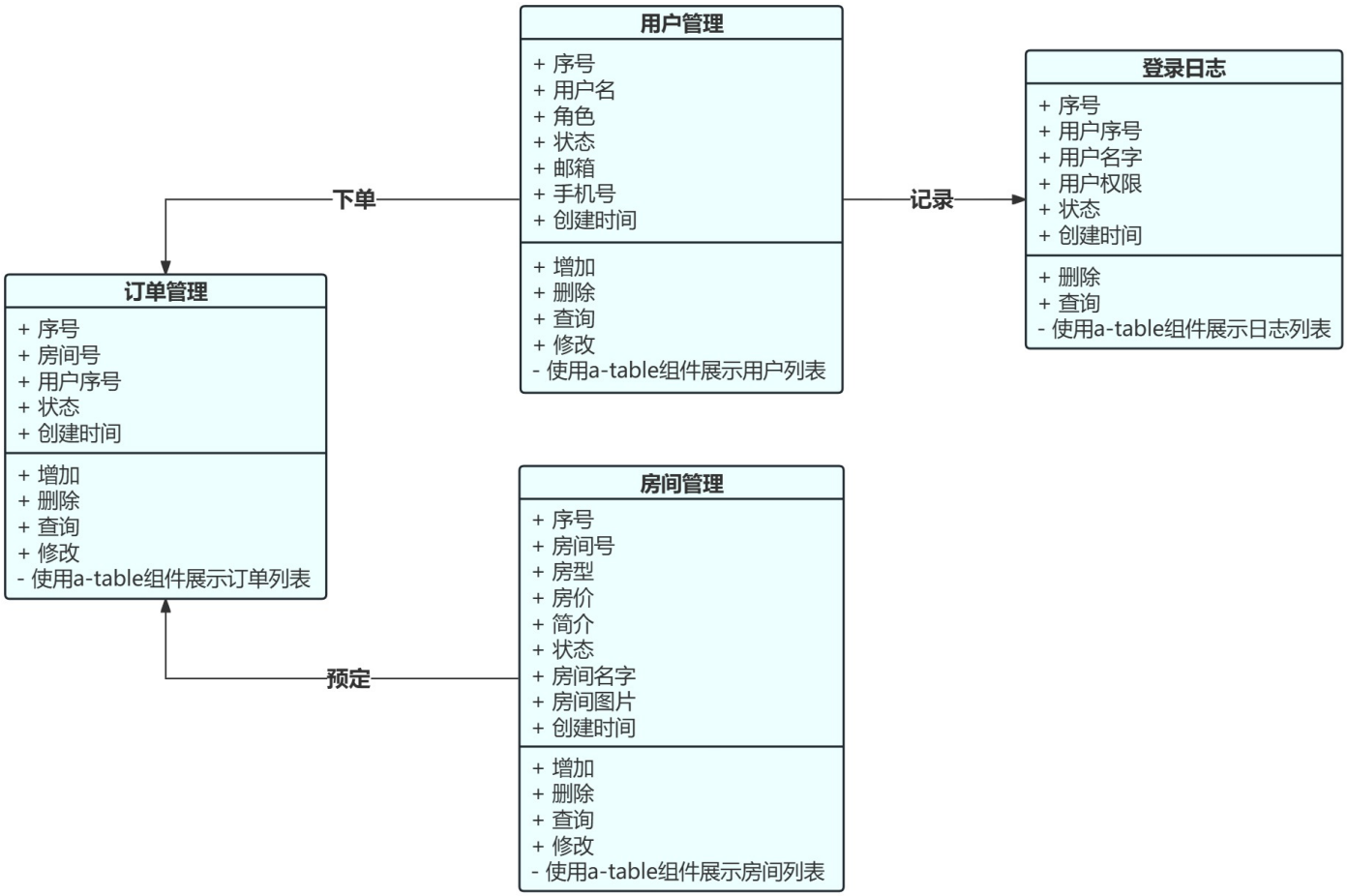


每个vue模块对应功能和界面实现

后台管理系统采用了响应式的 Ant Design Vue 框架，分为顶部栏（Header）、侧边栏（Sider）、主体内容（Content）三大部分。顶部栏包含酒店图标、系统名称、前台预定系统入口按钮、管理员退出按钮。我们将系统名称命名为“学人馆后台管理系统”，并在顶部栏中提供链接到前台预定系统的按钮，便于管理员进行管理。同时设计了退出按钮用于管理员注销当前登录状态进行重新登陆。

侧边栏是后台管理系统的主要导航工具，使管理员能够快速定位和操作相关功能。我们使用子菜单的方式将酒店管理的相关功能进行了归类以提高系统的整体可读性。侧边栏包含了系统的导航菜单，主要通过 Ant Design 的部件实现，采用了部分折叠和展开的设计，主要集成了用户管理、订单管理、酒店管理、登录日志等功能模块。

主体内容区域通过 Vue Router 实现页面切换，根据用户点击的菜单项加载相应的组件，从而实现管理功能的切换。点击菜单项时，使用 Vue Router 进行页面切换，实现单页面应用的无刷新加载



后台管理UML图

3.前台预定系统模块

同样采用 Vue.js 框架进行开发，通过 Vue 组件化的方式搭建页面。

前台预定系统设置的相对简易，登陆后首先导航到门户页面，在门户页面中展示了房间图片。用户可以在门户页面根据房间类型选择理想的房间。点击房间链接后，进入房间详情页可以看见房间简介信息。

Vue 部件名称	Vue 部件用法
potal.vue	门户页面模块
datail.vue	房间详情模块
confirm.vue	预定确认模块

主要Vue部件

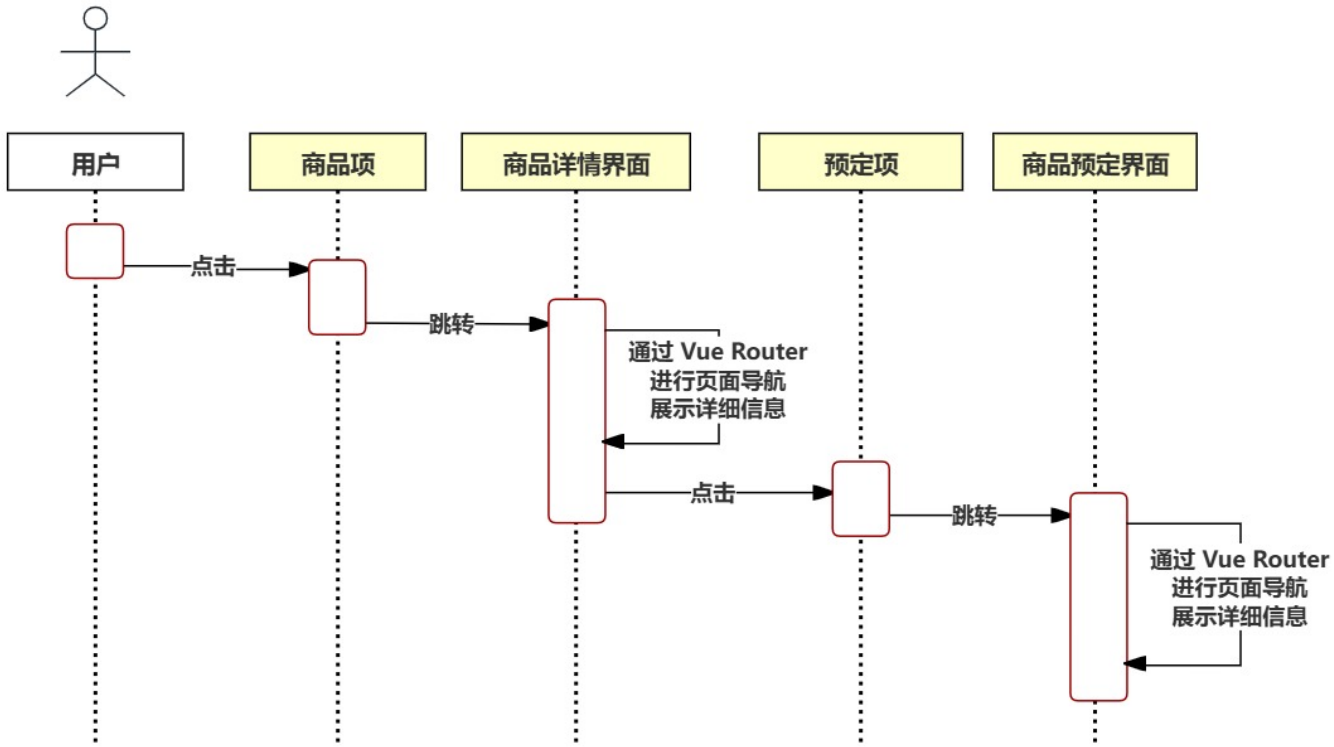
门户页面设计

门户页面模块整体架构分为两个主要组件： Header、 Content。其中， Header 用于显示系统顶部导航， Content 包含门户页面的具体内容。

内容区域包括左侧和右侧两个部分，分别展示房型分类和商品列表。左侧通过树形结构展示房型分类，右侧则展示商品列表，并支持分页浏览。用户可以点击商品进行详细查看或立即预订。

系统通过异步请求获取房型分类和标签数据，并在页面初始化时进行加载。左侧房型分类使用a-tree组件展示，支持点击选择不同房型分类，触发商品列表的刷新。

商品列表通过 a-spin 组件显示加载状态，确保在数据加载期间提供友好的用户体验。商品列表采用响应式设计，支持分页浏览，并在页面底部显示分页器。



用户预定UML时序图

详情页面设计

房间详情页面由头部、内容区域两个部分构成，其中内容区域包含上下两个主要部分，分别展示房间的基本信息和详细内容。

房间信息展示分为顶部和底部两个部分。顶部部分主要展示房间的基本信息，包括房间状态、名称、价格等。底部部分采用 Tab 切换，展示房间的简介等内容。

在布局上，采用了 Flex 布局，实现页面在不同设备上的自适应。通过 Less 预处理器定义样式，使得样式代码更加清晰、易维护。

预定确认页面设计

由于预定功能涉及生成订单、支付等复杂功能，这里并没有实现真正的预定功能。但是考虑到一个预定系统的合理性，设计了一个文本框来提示以下信息“线上预定功能暂不开放！如果需要预定请致电123456 预定！”。

后端部分

Django 框架下通过 ORM 和数据库映射，通过 DRF 向前端提供接口。

1.后端与数据库互连模块

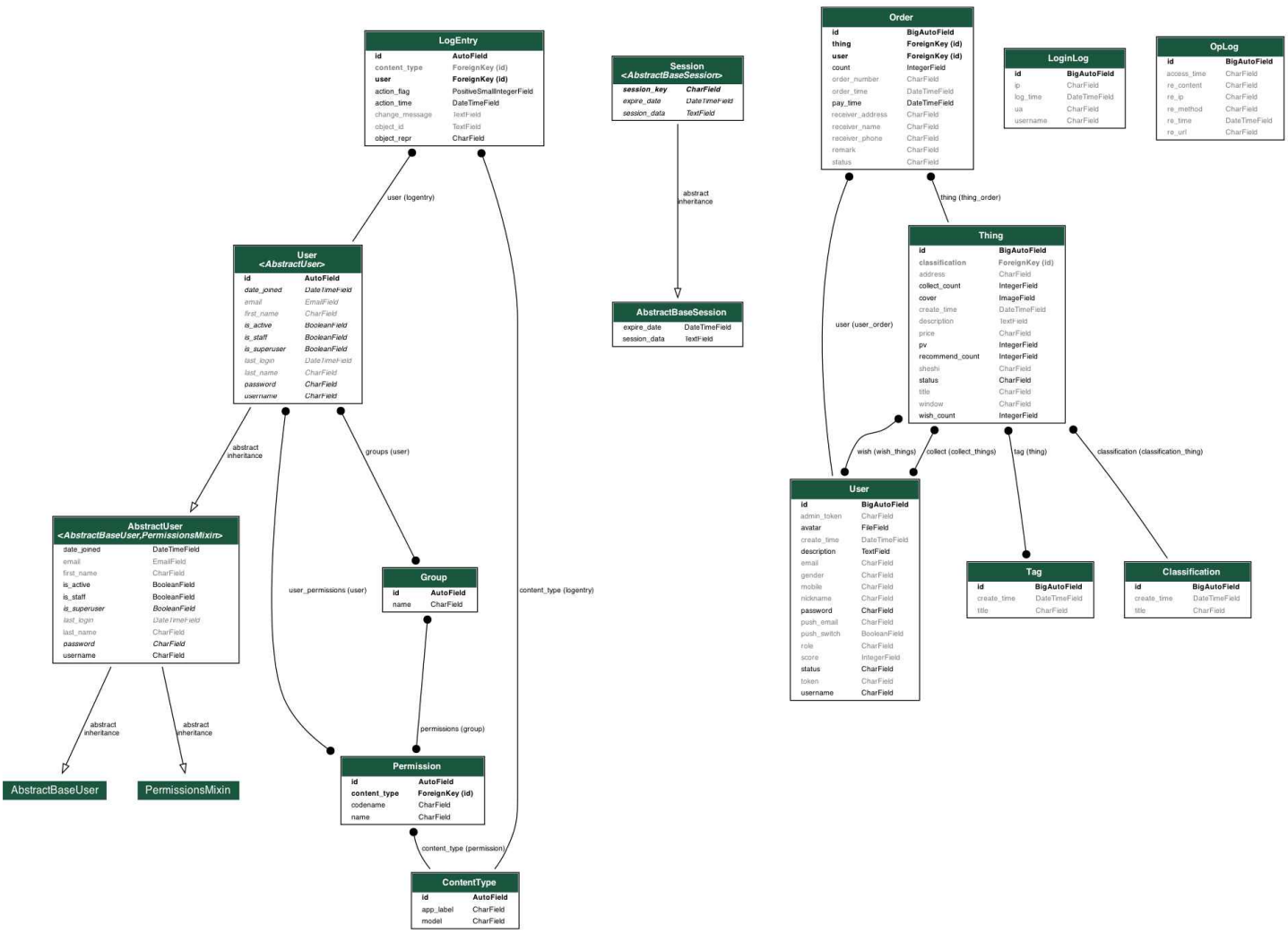
模型 Models 在 myhotel/models.py 中给出，通 ORM 系统，用 python 编写 Django 模型类来作为数据库的关系表。

ORM 系统即 Object-Relational Mapping，它允许开发者通过使用 Python 代码而不是 SQL 语句来操作数据库，可以将数据库中的表映射到 Python 对象，从而更加方便地进行数据库操作。还提供了强大的数据迁移工具，允许开发者在数据库结构发生变化时进行同步，而不会丢失数据。这使得数据库的升级和维护变得更加容易。

2.功能与接口设计模块

URL 分发器在 urls.py 中定义了 Web 应用程序的 URL 路由，它可以将特定的 URL 页面请求映射到相应的视图函数进行处理。视图 Views 在 myhotel/views 中给出，不同的角色（管理员、用户）对应不同的视图函数。

在将前端请求的 URL 映射到 Django REST Framework 定义的视图函数后， DRF 视图函数负责处理 Web 请求，根据请求的类型（GET、 POST 等）执行相应的操作，一般的操作流程是：从请求中获取参数，对请求的数据进行验证，并进行序列化处理，将数据转换成合适的格式；执行视图函数的主要业务逻辑，可能涉及数据库操作、调用其他服务、或其他相关操作；最后，根据业务逻辑的执行结果，构建合适的 Web 响应并返回给客户端。



后端Django框架UML类图

设计模式

- **状态模式：**酒店管理系统中有多种不同的状态，例如酒店房间的空闲状态、预定状态和入住状态等。状态模式可以用来描述这些状态之间的转换过程，例如当一个房间从空闲状态转换到预定状态时，就需要触发一些特定的行为。
- **观察者模式：**酒店管理系统中有多个组件需要监控某些状态的变化，例如前台需要知道哪些房间已经入住，哪些房间还有空余。观察者模式可以用来实现这种功能，让前台成为房间状态的观察者，并在状态发生变化时及时得到通知。
- **命令模式：**酒店管理系统中有多种不同的命令，例如预定房间、取消预定、办理入住等。命令模式可以用来描述这些命令的执行过程，包括命令的执行者和接收者以及命令执行过程中可能出现的一些异常情况。
- **迭代器模式：**酒店管理系统中有多种不同的数据结构，例如房间列表、客户列表等。迭代器模式可以用来描述这些数据结构的遍历过程，使得代码更加清晰简洁。
- **单例模式：**酒店管理系统中有多个组件需要共享某些资源，例如数据库连接池、日志记录器等。单例模式可以用来实现这些组件的万共享，避免资源浪费和冲突问题。
- **工厂模式：**酒店管理系统中有多种不同类型的房间，可能需要不同的初始化过程，而工厂模式可以在创建对象时将实例化过程抽象出来，这样在创建不同类型的房间对象时，就不需要直接使用构造函数。
- **策略模式：**酒店管理系统中有不同的用户类型，如预定者和管理者等，它们享有对系统不同的访问权限。策略模式可以用于定义一系列算法，并将每个算法封装起来，使得它们可以相互替换，不同的用户可以对应不同的使用策略。
- **装饰器模式：**酒店管理系统中，不同的房间类型可能会对应不同的附加服务，这些服务可以动态地添加到房间上。装饰器模式允许向一个对象动态添加职责，而不影响其他对象。通过使用装饰器模式，可以灵活地为各个类型的房间添加或删除附加服务。
- **代理模式：**酒店管理系统中，某些操作可能需要控制访问权限，例如客户只有在登录后才能查看预订信息和进行预定。代理模式提供一个代理对象来控制对实际对象的访问。代理对象可以在访问实际对象前后执行一些额外操作，例如权限检查或缓存。
- **责任链模式：**酒店管理系统中，客户的请求可能需要经过多个步骤处理，例如订房请求需要经过选择房型、确认预定、电话预定、验证身份、确定支付方式、完成预定等步骤。责任链模式使多个对象都有机会处理请求，从而避免请求发送者与接收者的耦合。将这些对象连成一条链，并沿着这条链传递请求，直到有对象处理它为止。