

Assignment 3

医疗综合服务系统



小组成员

肖筱游

莫健文

袁枫烨

陈敬麒

崔敬文

目录

[1 架构设计 1](#_Toc106395625)

[2 子系统和接口 3](#_Toc106395626)

[3 分析机制设计 15](#_Toc106395627)

[4 类图更新 18](#_Toc106395628)

[5 用例实现 22](#_Toc106395629)

[6 设计模型中的问题 25](#_Toc106395630)

[7 反思 25](#_Toc106395630)

[8 小组分工 25](#_Toc106395631)

### 

### 1 架构设计

#### 1.1 实现平台和框架

在实际开发过程中，我们将采用微服务架构。微服务架构能够实现由小型自主服务组成一个整体应用，各个组成部分之间是松耦合的，复杂性低，各个部分可以独立部署，修复bug或者引入新特性更容易，能够独立扩展，不同技术栈之间可以使用不同框架、不同版本库甚至不同的操作系统平台。我们的医疗综合服务平台，实际是由多个相对独立的功能集成实现的，采用微服务架构，在开发过程中容易实现独立开发，保证各个组件的高可用性和稳定性，并且拥有强大的灵活性，能更有针对性地解决问题。为了实现微服务框架，我们将采用一系列有效的技术和框架来构建该系统，具体内容如下：

1.web应用：运用vue.js+node.js、npm以及UI组件库（如Vuetify、Vant）和其他常用前端工具搭建前端，并使用AJAX实现不刷新页面而更新页面上的信息的功能。

2.API网关：利用API网关作为统一的外部接口，应用Spring Cloud实现微服务。Spring Cloud包括了很多的组件，功能齐全，为微服务架构提供了非常完整的支持，且其适用于各种环境的开发、部署，我们可以通过使用Spring Cloud轻松开发分布式应用服务。

3.数据存储管理：使用Oracle关系数据库管理系统作为主要的数据库来存储数据；使用Redis来满足高并发场景中数据的安全性和一致性；使用高性能、高扩展性的FastDFS分布式设计文件系统对文件进行管理，解决大容量的文件存储和高并发访问的问题，提高文件存取的性能和效率；使用ElasticSearch作为分布式搜索引擎，实现高性能的分布式数据搜索和文件存储；最后使用NIFI处理和分发数据，自动化管理系统间的数据流。

4.服务器框架：采用Spring boot作为Web后端的开发框架，用JAVA进行后端开发。Spring boot作为强大的web开发框架，具有高扩展性和优异的性能，拥有极高的开发、部署效率，对主流开发框架的无配置集成，与云计算的天然集成，且是一个很好的微服务开发的框架基础，很适合作为当前项目的服务器框架。

#### 1.2关键决策

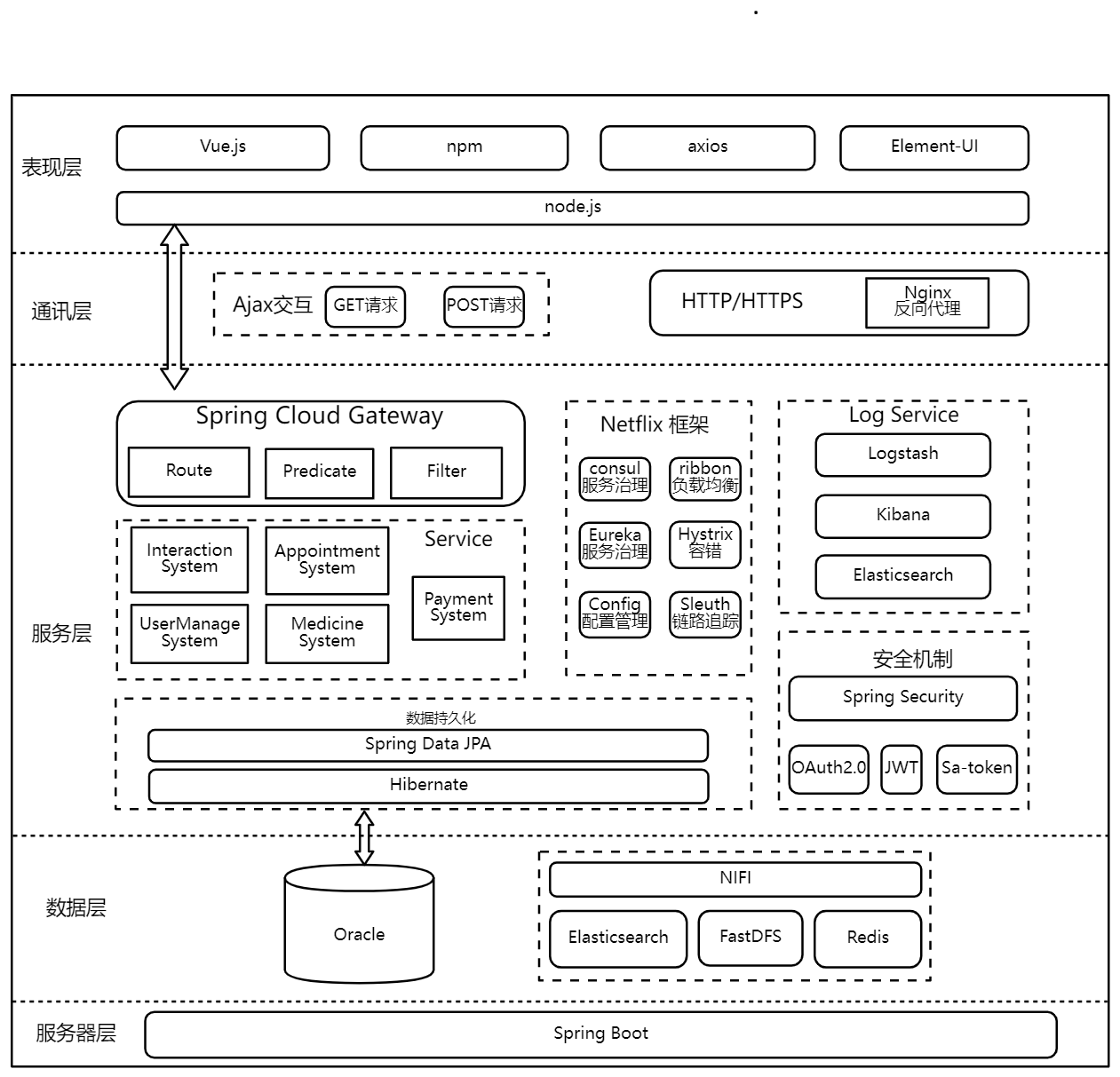
基于该程序所面向的用户群体广泛、数量较多，我们采用了微服务架构进行实现，保证了系统的灵活性，适用于多种平台上。

应用Spring Cloud实现微服务。Spring Cloud包括了很多的组件，功能齐全，为微服务架构提供了非常完整的支持，且其适用于各种环境的开发、部署，我们可以通过使用Spring Cloud轻松开发分布式应用服务。

由于本系统应用于患者就诊，涉及到患者的历史病历、药品信息查询等，需要数据永久保存以实现系统功能。为此我们引入了因此引入了持久化机制，对于持久性的实现，我们使用关系数据库管理系统（RDBMS）作为设计机制，运用关系数据库存储数据。

由于本系统涉及用户数量众多，需要确保每个参与者都能正常获得自己的服务或资源。因此，该系统需要提供全面、合理的访问控制安全机制。针对安全性，我们将访问控制安全机制作为我们的重要设计机制。我们使用Sa-Token权限认证框架，要解决登录认证、权限认证、Session会话、单点登录、OAuth2.0、微服务网关鉴权等一系列权限相关问题，同时集成了登录认证、权限认证和数十个权限相关功能，比较适合开发本系统。

#### 架构改进



系统采用微服务架构。开发时采取前后端分离的形式，前端使用Vue、npm等进行开发，后端使用Spring Cloud进行开发，数据层采用Oracle进行数据存储，数据持久化主要由Spring Data JPA实现。1. ElasticSearch，通常配合logstash、kibana，三者一起用于日志分析系统，即log search。2. Nacos是针对微服务架构中的服务发现、配置管理、服务治理的综合型解决方案。具体功能有：服务发现和服务健康监测、动态配置服务、动态 DNS 服务、服务及其元数据管理，常用于微服务框架的服务管理。

1. 微服务安全性保证：

JWT：全称是：json web token。它将用户信息加密到token里，服务器不保存任何用户信息。服务器通过使用保存的密钥验证token的正确性，只要正确即通过验证。可以支持各种形式的web，且不需要在服务端保存会话信息，特别适用于分布式微服务。

OAuth2.0是一个关于授权（authorization）的开放网络标准，

简单来说就是客户端应用程序（通常是web浏览器），代表用户（得到了用户的批准）去访问受保护的资源

Sa-Token 是一个轻量级 Java 权限认证框架，主要解决：登录认证、权限认证、Session会话、单点登录、OAuth2.0、微服务网关鉴权等一系列权限相关问题。

### 2 子系统和接口

#### 2.1 子系统和接口列表

1. 用户管理子系统

(1) 功能说明：管理用户信息，包括医生账户和患者账户信息

(2) 接口：

① UserLogin(String account,String password):bool

② UserRegister(String acount,String password):bool

③ UserPasswordsChange

(String account,String password,String newpassword):bool

④ UserInfoGet(String account):UserInfo

⑤ UserInfoEdit(String account):UserInfo

⑥ DocInfoGet(String name,String department):UserInfo

⑦ PatientInfoGet

(String patientname,String patientaccount):UserInfo

2. 医患交流子系统

(1) 功能说明：提供线上交流平台，包含视频和文字服务

(2) 接口：

① SendTextMessge(String text, String account):bool

② SendVoiceMessge(Voice voice, String account):bool

③ SendImgMessge(Img image, String account):bool

④ GetTextMessge(String account):String

⑤ GetVideoMessge(String account):Video

⑥ GetImgMessge(String account):Img

⑦ CreateVideoCall(String account\_1,String account\_2):bool

3. 药品管理子系统

(1) 功能说明：查询药品数量、药品使用申请以及药品库存管理。

(2) 接口：

① GetMedicineNum(String medicine\_id):int

② MedicineApplication(String medicine\_id,Int amount):bool

③ IncreaseMedicine(String medicine\_id,Int amount):bool

④ DeclineMedeicnie(String medicine\_id,Int amount):bool

⑤ AddMedicine(String medicine\_id,Int amount):bool

⑥ DeleteMedicine(String medicine\_id):bool

4. 就诊服务子系统

(1) 功能说明：提供预约挂号、排队等待等业务逻辑功能的接口

(2) 预约功能接口：

① SubmitAppointment

(String patient\_account,String doc\_account,Date time):appointment

② CancelAppointment(String appointment\_id):bool

③ ViewAppointment(String appointment\_id):appointment

④ GetQueueInfo(String patient\_account):int

⑤ GetMedicalRecord(String patient\_account):medicalrecord

⑥ FinishAppointment(String account):void

⑦ FinishInspection(String account):void

5. 在线支付子系统

(1) 功能说明：为患者提供支付功能接口，包括诊疗支付和挂号支付

(2) 接口：

① IsPaid(String account):bool

② Pay(String appointment\_id):bool

#### 2.2 子系统和接口规范

**1、数据类说明**

**(1) UserInfo 户信息类型**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数名称 | 参数类型 | 参数描述 |
| account | 用户账号 | String | 用户账号 |
| name | 用户姓名 | String | 用户姓名 |
| sex | 用户性别 | String | 用户性别 |
| password | 用户密码 | String | 用户密码，被加密，通常不可见 |
| birth | 用户生日 | Date | 生日日期 |
| identity | 用户属性 | String | 用户的身份类型，如医生或者患者等 |
| level | 用户权限等级 | int | 用户权限等级，限制用户操作范围 |
| Phone | 用户电话 | String | 用户电话 |

**（2）Date 日期类型**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数名称 | 参数类型 | 参数描述 |
| year | 年份 | int | 年份 |
| month | 月份 | String | 月份 |
| day | 日期 | String | 日期 |

**（3）TextInfo 文本数据类型**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数名称 | 参数类型 | 参数描述 | 是否为空 |
| Code | 响应编码 | String | 响应编码 | N |
| Msg | 响应消息 | string | 响应消息 | N |
| Systime | 系统响应时间 | String | 系统响应时间 | N |
| avatar | 用户头像 | Img | 用户头像 | N |
| IsRead | 读标志 | bool | 消息是否被读 | N |
| IsOneself | 本人发送标志 | bool | 是否是自己发送  的消息 | N |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| msgID | 消息 ID | string | 消息 ID，用于  区分不同消息 | N |
| NickName | 用户昵称 | string | 用户昵称 | N |
| userCode | 用户编码 | string | 用户编码 | N |
| Type | 消息类型 | string | 消息类型 | N |
| content | 消息内容 | string | 消息内容 | N |

**（4）VoiceInfo 音频数据类型**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数名称 | 参数类型 | 参数描述 | 是否为空 |
| Code | 响应编码 | String | 响应编码 | N |
| Msg | 响应消息 | string | 响应消息 | N |
| Systime | 系统响应时间 | String | 系统响应时间 | N |
| avatar | 用户头像 | Img | 用户头像 | N |
| IsRead | 读标志 | bool | 消息是否被读 | N |
| IsOneself | 本人发送标志 | bool | 是否是自己发送  的消息 | N |
| msgID | 消息 ID | string | 消息 ID，用于  区分不同消息 | N |
| NickName | 用户昵称 | string | 用户昵称 | N |
| userCode | 用户编码 | string | 用户编码 | N |
| Type | 消息类型 | string | 消息类型 | N |
| content | 消息内容 | Int[] | 语音消息内容 | N |
| Size | 语音内存容量 | Int | 语音内存容量 | N |

**（5）ImageInfo 图片数据类型**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数名称 | 参数类型 | 参数描述 | 是否为空 |
| Code | 响应编码 | String | 响应编码 | N |
| Msg | 响应消息 | string | 响应消息 | N |
| Systime | 系统响应时间 | String | 系统响应时间 | N |
| avatar | 用户头像 | Img | 用户头像 | N |
| IsRead | 读标志 | bool | 消息是否被读 | N |
| IsOneself | 本人发送标志 | bool | 是否是自己发送  的消息 | N |
| msgID | 消息 ID | string | 消息 ID，用于  区分不同消息 | N |
| NickName | 用户昵称 | string | 用户昵称 | N |
| userCode | 用户编码 | string | 用户编码 | N |
| Type | 消息类型 | string | 消息类型 | N |
| content | 图片内容 | Int[][][3] | 图片内容 | N |
| ext | 图片类型 | string | 图片类型 | N |
| width | 图片宽度 | int | 图片宽度 | N |
| height | 图片高度 | int | 图片高度 | N |

**（5）Appointment 预约表单数据类型**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数名称 | 参数类型 | 参数描述 |
| account | 用户账号 | String | 用户账号 |
| name | 用户姓名 | String | 用户姓名 |
| sex | 用户性别 | String | 用户性别 |
| date | 预约生日 | Date | 预约日期 |
| doc\_account | 医生账号 | String | 被预约的医生账号 |

**（6）Medicalrecord历史病历数据类型**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数名称 | 参数类型 | 参数描述 |
| account | 用户账号 | String | 用户账号 |
| name | 用户姓名 | String | 用户姓名 |
| sex | 用户性别 | String | 用户性别 |
| date | 就诊生日 | Date | 就诊日期 |
| disease | 诊断疾病 | String | 患者被诊断的疾病 |
| doc\_account | 医生账号 | String | 被预约的医生账号 |
| prescription | 开具处方 | Vector<String> | 医生所开具的药物及检查 |

**2、接口规范**

以用户信息管理子系统中的UserInfoGet接口为例子，详细展示其接口的访问方式、具体方法的输入输出等规范。

**（1）请求方式：get**

**（2）请求域名：https://localhost:2022/api/UserInfoGet**

**（3）输入参数**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 参数名称 | 类型 | 描述 |  |
| account | 用户账号 | String | 用户描述 |  |

**（4）返回参数**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数名称 | 参数类型 | 参数描述 |
| account | 用户账号 | String | 用户账号 |
| name | 用户姓名 | String | 用户姓名 |
| sex | 用户性别 | String | 用户性别 |
| password | 用户密码 | String | 用户密码，被加密，通常不可见 |
| birth | 用户生日 | Date | 生日日期 |
| identity | 用户属性 | String | 用户的身份类型，如医生或者患者等 |
| level | 用户权限等级 | int | 用户权限等级，限制用户操作范围 |

**（5）返回示例 (以表中只有一条记录为例，返回json文件）**

|  |
| --- |
| {  “ result”:  {  “account”:”2051977”  “name”:”垣酆野”  “sex”:”male”  “password”:”114514”  “birth”:”2002-2-28”  “identity”:”Doc”  “level”:”1”  }        } |

#### 2.3 接口示例

以就诊服务子系统为示例，展示了外部系统如何调用接口完成相应场景中的业务逻辑，其具体API如下**：**

|  |
| --- |
| **接口编号** |
| **001** |
| **接口** |
| **SubmitAppointment**  (String patient\_account,String doc\_account,Date time):String |
| **参数** |
| String patient\_account,String doc\_account,Date time |
| **返回值** |
| **String** |
| **接口介绍** |
| **此接口功能是接受前端返回的预约表单的信息，向系统提交表单，如果成功返回预约表单的信息。** |

|  |
| --- |
| **接口编号** |
| **002** |
| **接口** |
| CancelAppointment(String appointment\_id):bool |
| **参数** |
| String appointment\_id |
| **返回值** |
| **bool** |
| **接口介绍** |
| **此接口的功能是接受前端取消预约的请求和表单编号，调用业务逻辑处理，返回成功或者失败** |

|  |
| --- |
| **接口编号** |
| **003** |
| **接口** |
| ViewAppointment(String appointment\_id):appointment |
| **参数** |
| String appointment\_id |
| **返回值** |
| appointment |
| **接口介绍** |
| **此接口的功能是接受前端查看预约表单的请求，调用业务逻辑处理，返回预约表单** |

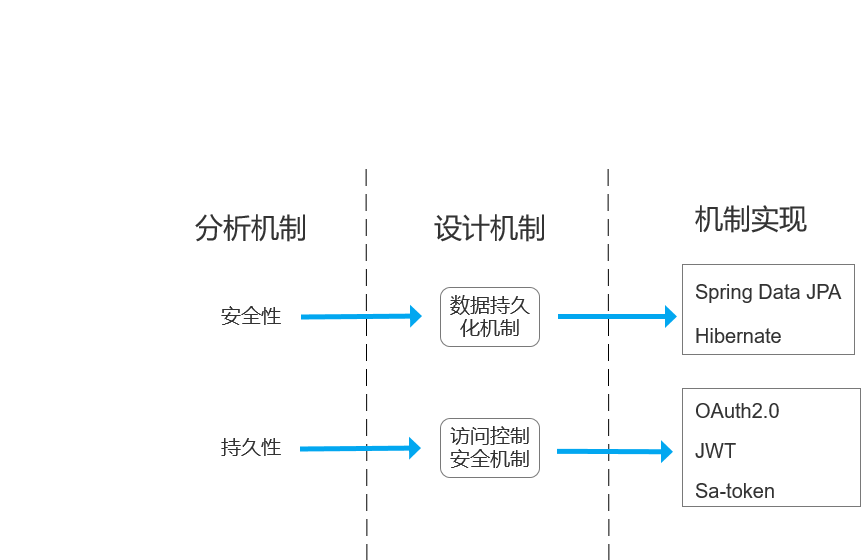
|  |
| --- |
| **接口编号** |
| **004** |
| **接口** |
| GetMedicalRecord(String patient\_account);medicalrecord |
| **参数** |
| String patient\_account |
| **返回值** |
| medicalrecord |
| **接口介绍** |
| **此接口的功能是接受前端查看用户的病历记录，调用业务逻辑，返回包含就诊数据的病历记录。** |

|  |
| --- |
| **接口编号** |
| **005** |
| **接口** |
| GetQueueInfo(String patient\_account):int |
| **参数** |
| String patient\_account |
| **返回值** |
| **int** |
| **接口介绍** |
| **此接口的功能是通过用户账号进行查找排队信息，调用业务逻辑，并返回当前队列中用户前的剩余排队人数,队列包括就诊、检查和取药队列。** |

|  |
| --- |
| **接口编号** |
| **006** |
| **接口** |
| FinishAppointment(String account):void |
| **参数** |
| String account |
| **返回值** |
| **void** |
| **接口介绍** |
| **此接口的功能是处理已经完成预约就诊的业务逻辑，更新预约等待队列** |

|  |
| --- |
| **接口编号** |
| **007** |
| **接口** |
| FinishInspection(String account):void |
| **参数** |
| String account |
| **返回值** |
| **void** |
| **接口介绍** |
| **此接口的功能是处理已经完成医疗服务（如取药、验血等）的业务逻辑，更新医疗服务等待队列** |

### 分析机制设计



在讨论设计机制之前，我们先考虑分析机制的问题。在我们的系统中，需要用到两个分析机制来评价系统，分别是持久性和安全性。

**1.持久性与数据持久化机制**

持久性是指保证能够把内存中的数据对象保存到外存中，让以后能够取回，实现永久保存的性质。持久性是必要的，因为内存掉电后数据会丢失，但有一些对象是无论如何都不能丢失的，如：银行账户。一方面，内存通常无法永远不断电，另一方面即使对象不需要永久保存，也会因为内存的容量限制不能一直呆在内存中，需要持久化来缓存到外存。

我们的医院综合服务平台系统中有存储有大量的数据信息，如：药品信息、医生信息、患者账户信息及病历信息等，由于数据需要永久保存以实现系统功能，因此引入了持久化机制。对于持久性的实现，我们使用关系数据库管理系统（RDBMS）作为设计机制，运用关系数据库存储数据。

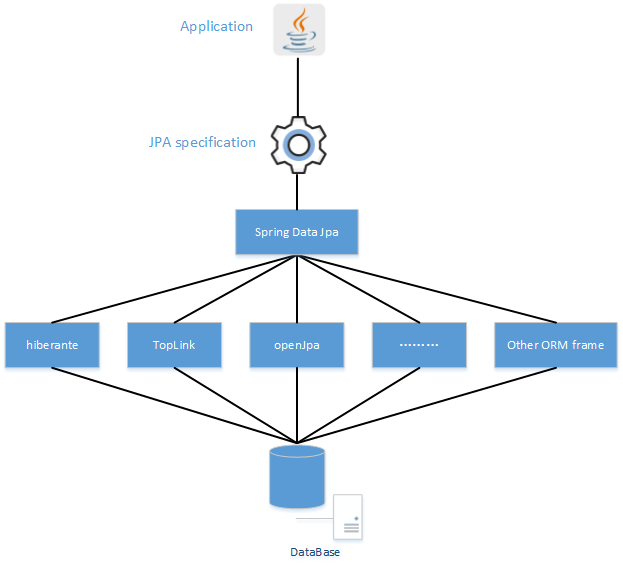
关系数据和对象是业务实体的两种形式。业务实体在内存中表示为对象，在数据库中表示为关系数据。内存中的对象之间存在许多关联和继承关系，这些关系不能用关系数据来表示。因此，数据持久化需要对象关系映射（ORM）。为了实现ORM，我们在业务层和数据库之间添加一个持久层。

持久层是一种中间件，它可以自动将对象存储在数据库中，也可以将数据从数据库加载到内存中。在持久层中，则会使用持久化技术。持久化技术除了保证数据的永久储存，还封装了数据访问细节，为大部分业务逻辑提供面向对象的API。使用持久化技术一方面可以减少访问数据库数据次数，增加应用程序执行速度；另一方面，其代码重用性高，能够完成大部分数据库操作；此外，还能实现松散耦合，使持久化不依赖于底层数据库和上层业务逻辑实现，更换数据库时只需修改配置文件而不用修改代码。通过持久层的引入，程序将通过会话间接访问数据库。

**技术实现：**

Java持久层框架访问数据库的方式大致分为两种。一种以SQL核心，封装一定程度的JDBC操作，例如MyBatis。另一种以Java实体类为核心，将实体类和数据库表之间建立映射关系，也就是所说的ORM框架，如Hibernate、Spring Data JPA。

我们的项目持久层主要采用的是Spring Data JPA框架实现持久层，实现框架图如下所示：



**JPA:**

JPA全称是Java Persistence API(Java持久层API)，是一种Java持久化规范。它为Java开发人员提供了一种对象/关联映射工具，来管理Java应用中的关系数据。JPA吸取了目前Java持久化技术的优点，旨在规范、简化Java对象的持久化工作。很多ORM框架实现了JPA的规范，如Hibernate、EclipseLink。JPA统一了Java应用程序访问ORM框架的规范

**Spring Data JPA：**

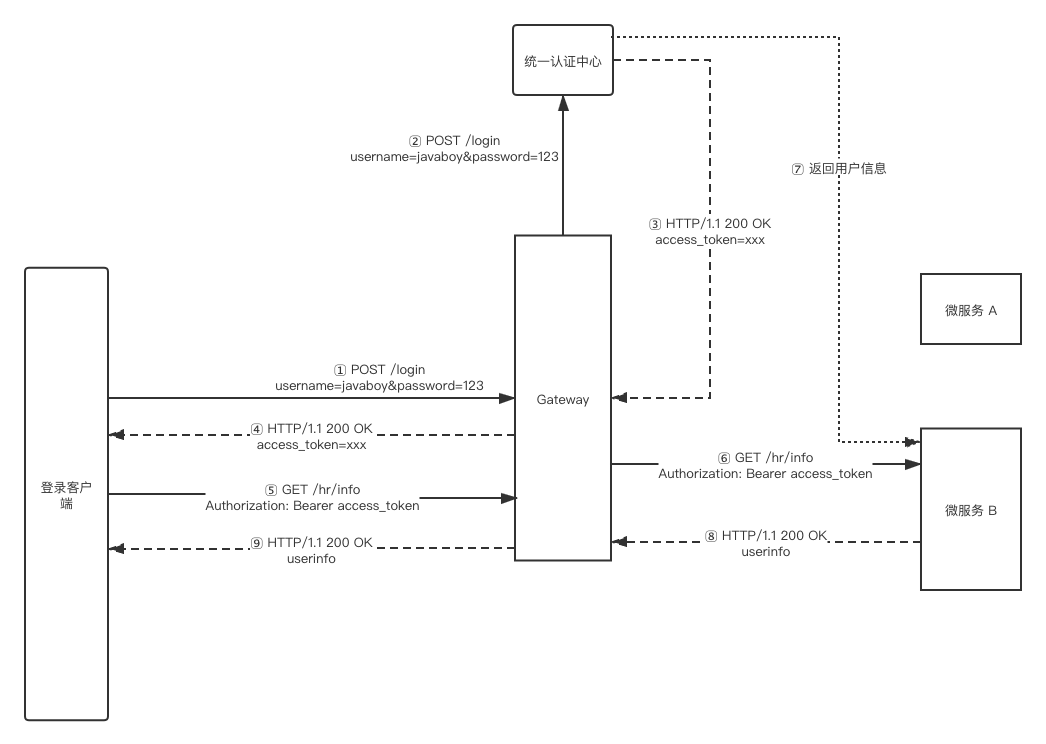
Spring Data JPA是在实现了JPA规范的基础上封装的一套JPA应用框架。虽然ORM框架都实现了JPA规范，但是在不同的ORM框架之间切换仍然需要编写不同的代码，而使用Spring Data JPA能够方便大家在不同的ORM框架之间进行切换而不需要更改代码。在我们的系统中，Spring Data JPA是在JPA规范下提供了Repository层的再次封装，并提供了更多可用的接口，在底层的ORM框架则主要通过Hibernate实现。同时，由于SpringData JPA是全自动框架，大大简化了持久层的开发。

**Hibernate：**

Hibernate是一个采用ORM（Object/Relation Mapping对象关系映射）机制持久层的开源框架，它用非常轻量级的对象封装JDBC，生成SQL语句并自动执行它们。这样的设计能够减少对象与DBMS之间的耦合，使得框架具有完整的日志系统和更高的可移植性。

**2.安全性与访问控制安全机制**

安全性是对于一个系统开发而言十分重要的点，因此我们引入安全性作为我们的分析机制之一。我们的系统涉及许多角色，包括医生、患者、管理者等。不同的角色有不同的权限，而且统一角色根据其个人关联关系不同，也会拥有对信息的不同的访问权限。例如，管理者可以查看所有医生信息，而其他角色则无权；进行线上问诊的医生或与特定患者关联的医生可以访问特定患者的病历，而其他无关医生则无此权限。我们需要确保每个参与者都能正常获得自己的服务或资源，因此毫无疑问，该系统需要提供全面、合理的访问控制安全机制。针对安全性，我们将访问控制安全机制作为我们的重要设计机制。



访问控制安全机制中最重要的一环就是权限认证机制。如上图所示是微服务架构的权限认证流程，对于普通的认证标准，往往安全性存在问题，而这个安全性漏洞可以通过引入OAuth 2.0来解决。在我们的系统中，我们采用基于OAuth 2.0标准协议和API网关的微服务架构模式来响应用户的请求，同时引入JWT来解决OAuth2.0存在的一些漏洞，并引入sa-token作为权限认证框架。

**OAuth2.0：**

OAuth2.0是一个开放标准，该标准允许用户让第三方应用访问该用户在某一网站上存储的私密资源（如头像、照片、视频等），而在这个过程中无需将用户名和密码提供给第三方应用。实现这一功能是通过提供一个令牌（token），而不是用户名和密码来访问他们存放在特定服务提供者的数据。采用令牌的方式可以让用户灵活的对第三方应用授权或者收回权限。与基于cookie的身份验证相比，基于令牌的身份验证是无状态的，可以进行跨站点请求伪造（CSRF），并且支持多站点使用。OAuth2.0认证权限的流程如下：

1. 客户端向资源所有者请求授权。客户端可以直接向资源所有者发起授权请求，然后根据客户端请求授权的方法和授权服务器支持的类型接收到对应的授权信息。
2. 根据获取的授权信息，客户端与授权服务器进行身份验证，并请求访问令牌。授权服务器验证客户端的身份，并在通过身份验证时发出访问令牌。
3. 客户端从资源服务器请求受保护的资源，并通过提供访问令牌进行身份验证。如果令牌有效，将处理客户端的请求。

**JWT：**

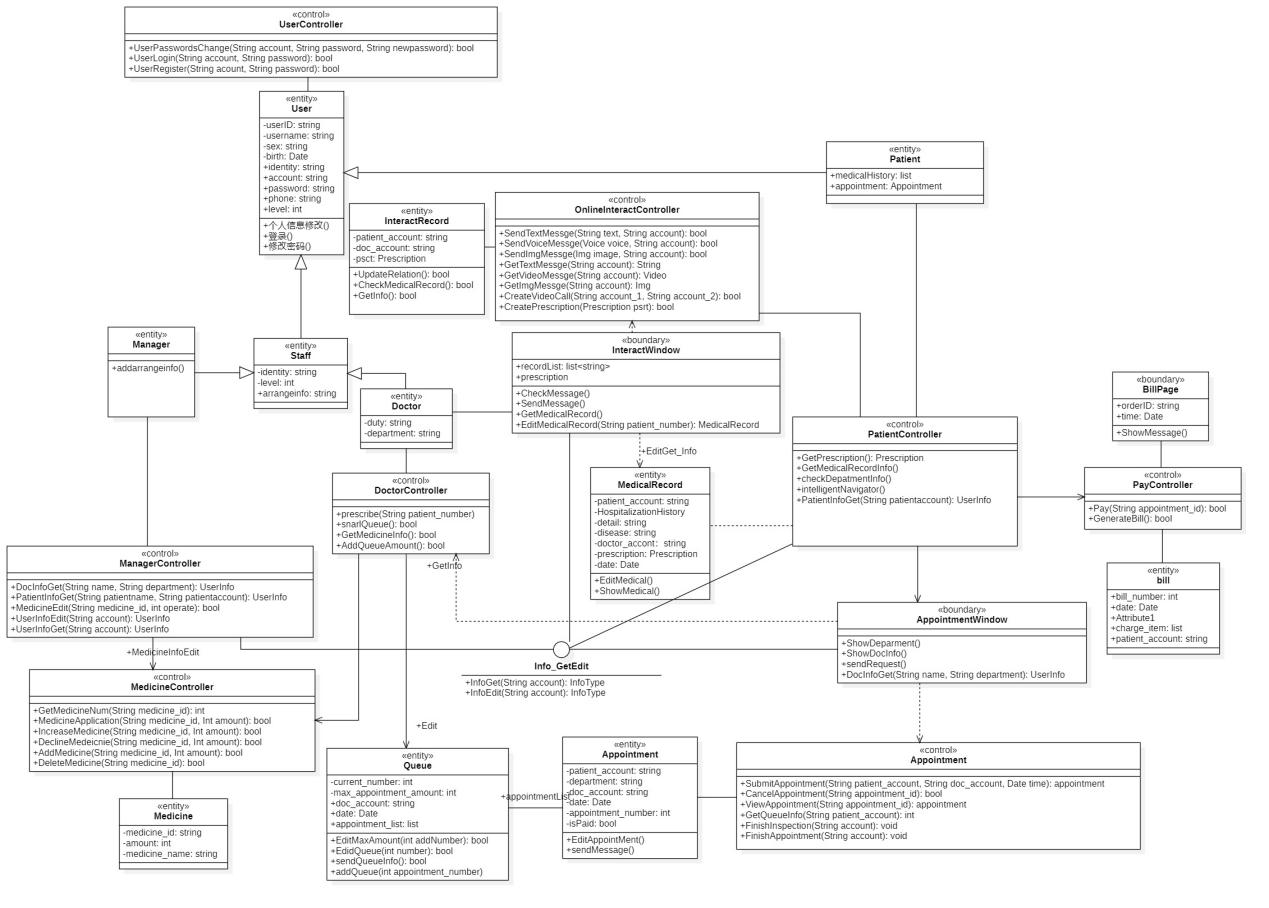
全称是：json web token，是一种 JSON 风格的轻量级的授权和身份认证规范，可实现无状态、分布式的 Web 应用授权。它将用户信息加密到token里，服务器不保存任何用户信息。服务器通过使用保存的密钥验证token的正确性，只要正确即通过验证。可以支持各种形式的web，且不需要在服务端保存会话信息，特别适用于分布式微服务。在高并发环境下，OAuth2.0的校验方式存在一定的问题，而如果结合 JWT，用户的所有信息都保存在 JWT 中，则可以有效改善该问题。

**Sa-Token：**

Sa-Token 是一个轻量级 Java 权限认证框架，主要解决登录认证、权限认证、Session会话、单点登录、OAuth2.0、微服务网关鉴权等一系列权限相关问题。与其他认证框架相比，轻量级Sa-Token是一个轻量级的零配置启动框架，开发更简单；它是为前后端分离而设计的框架，在微服务架构模式下，Sa-token集成了登录认证、权限认证和数十个权限相关功能，比较适合开发本系统；同时，许多高级功能可以通过简单的API调用来实现。

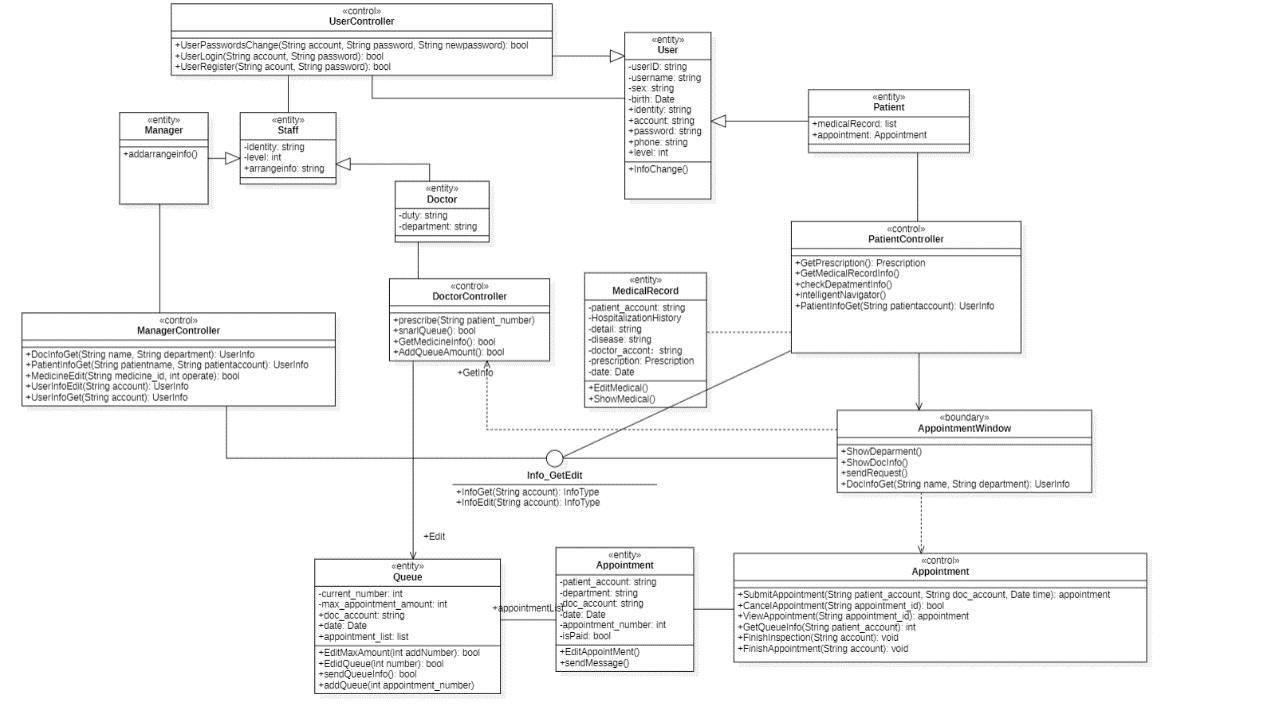
### 类图更新

#### 总类图

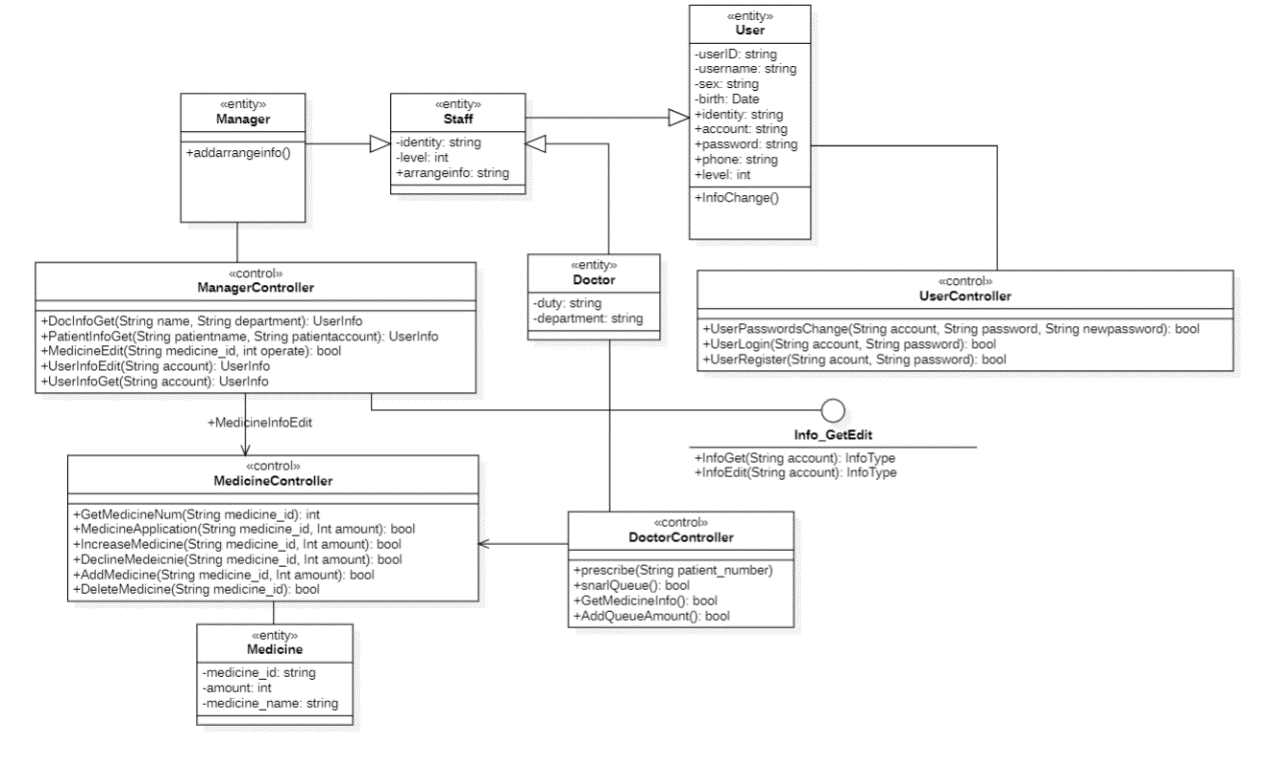
在对类图进行再设计时，我们采用了MVC模式进行类图重构。MVC 模式代表 Model-View-Controller（模型-视图-控制器） 模式。这种模式用于应用程序的分层开发，将视图层与业务层分离，有利于组件的重用、开发的分工。同时也更易于维护与修改。

#### 子系统类图

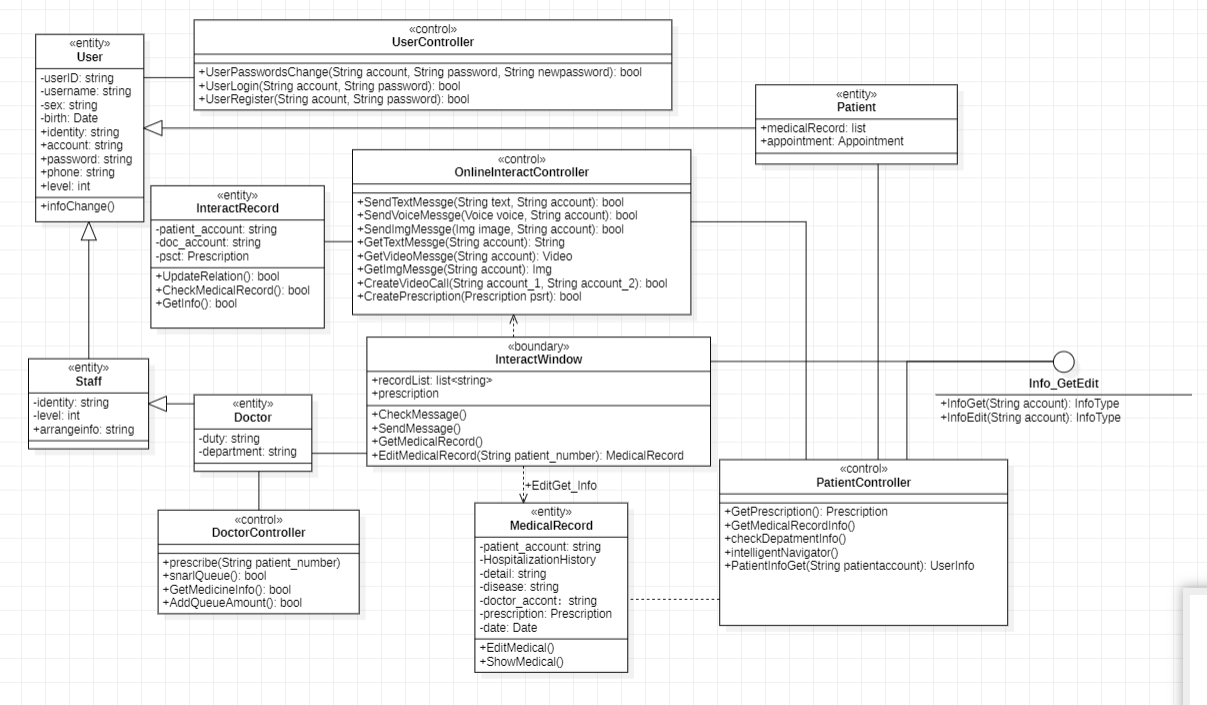
就诊服务子系统



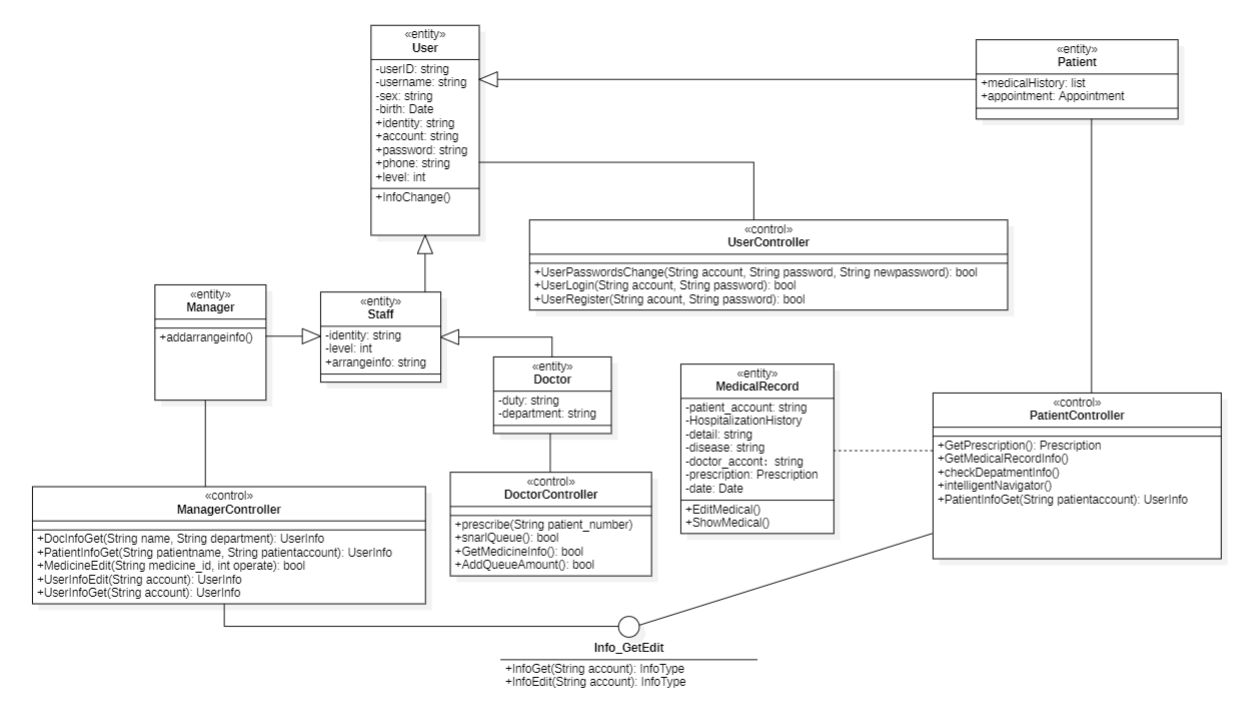
药品管理子系统



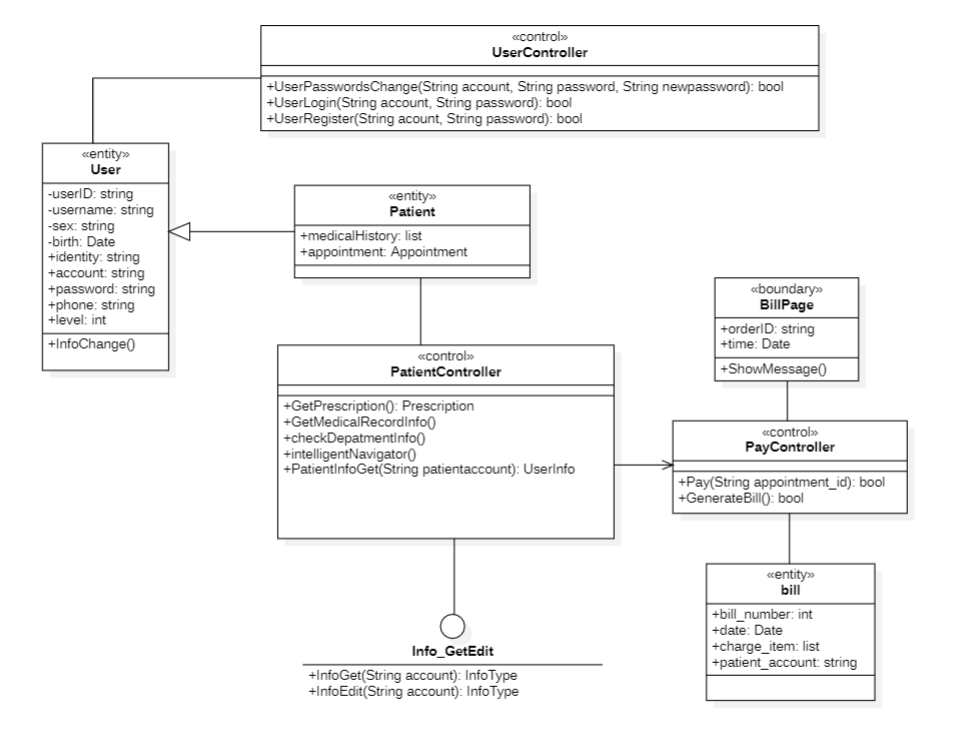
医患交流子系统



用户管理子系统

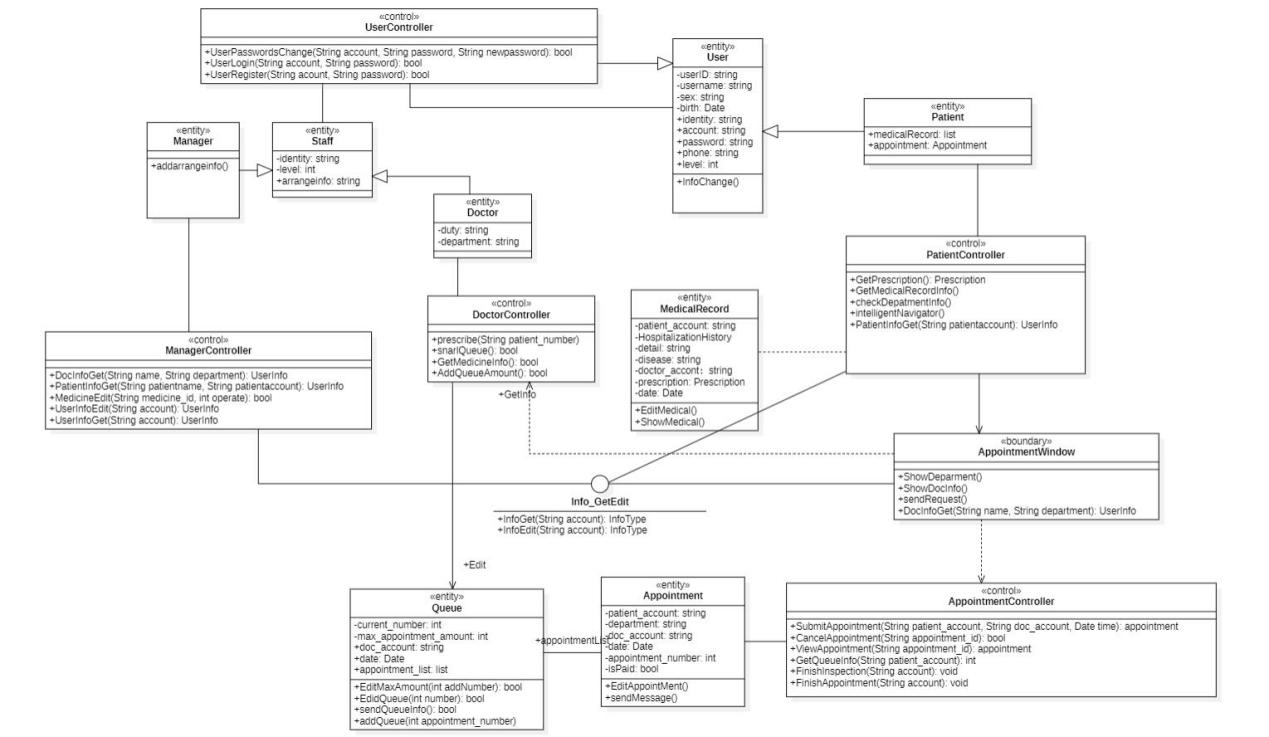


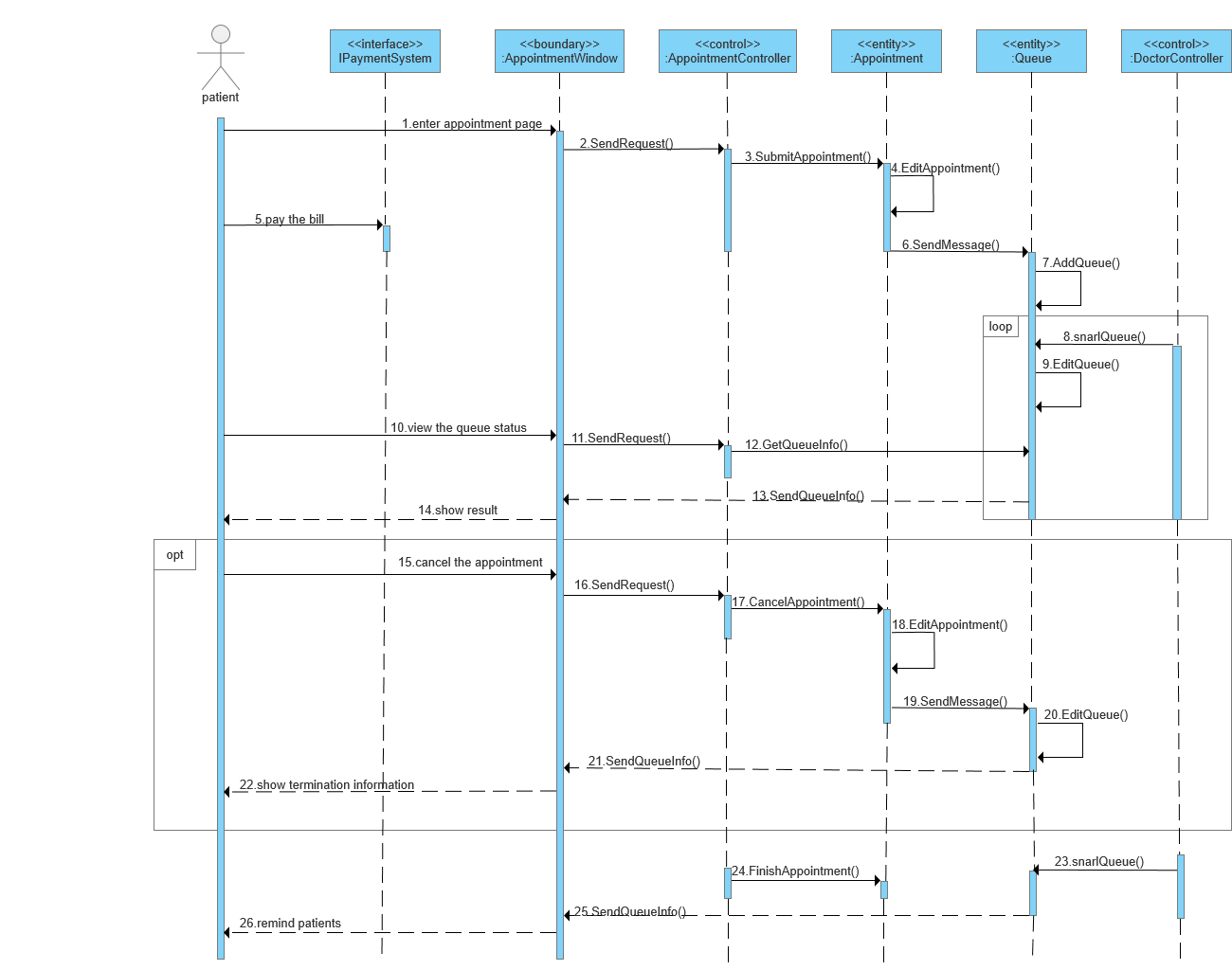
在线支付子系统



### 用例实现

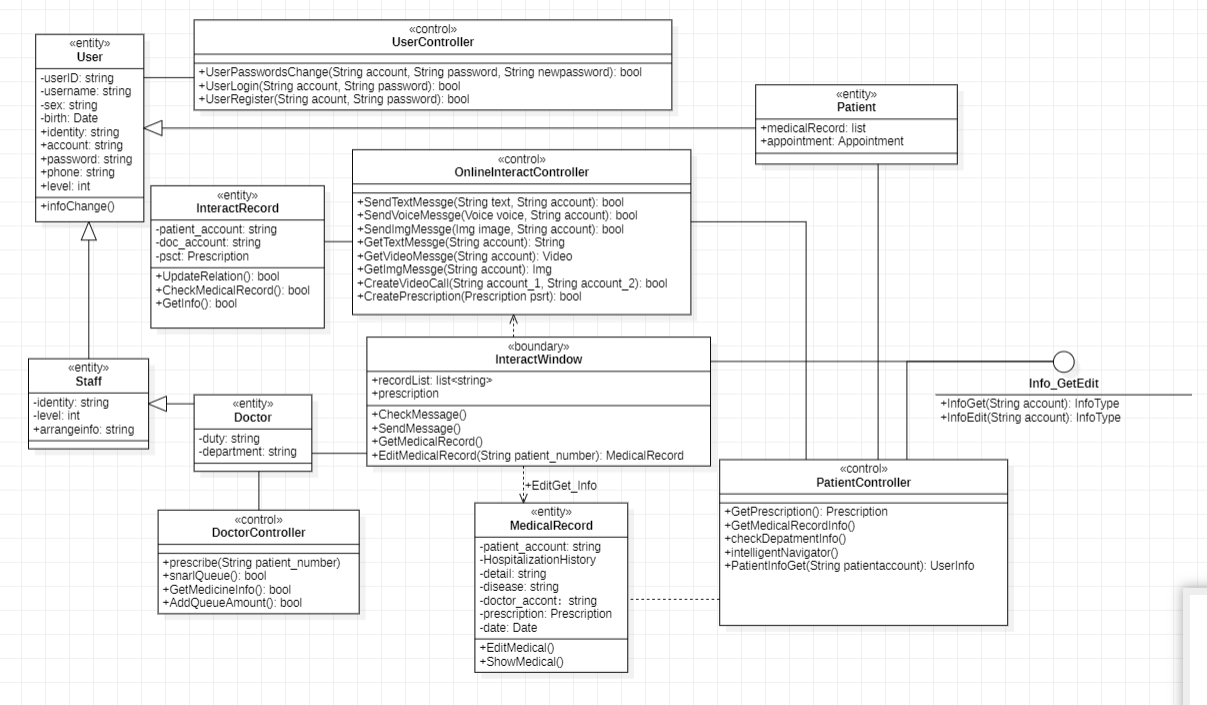
#### 就诊服务用例实现

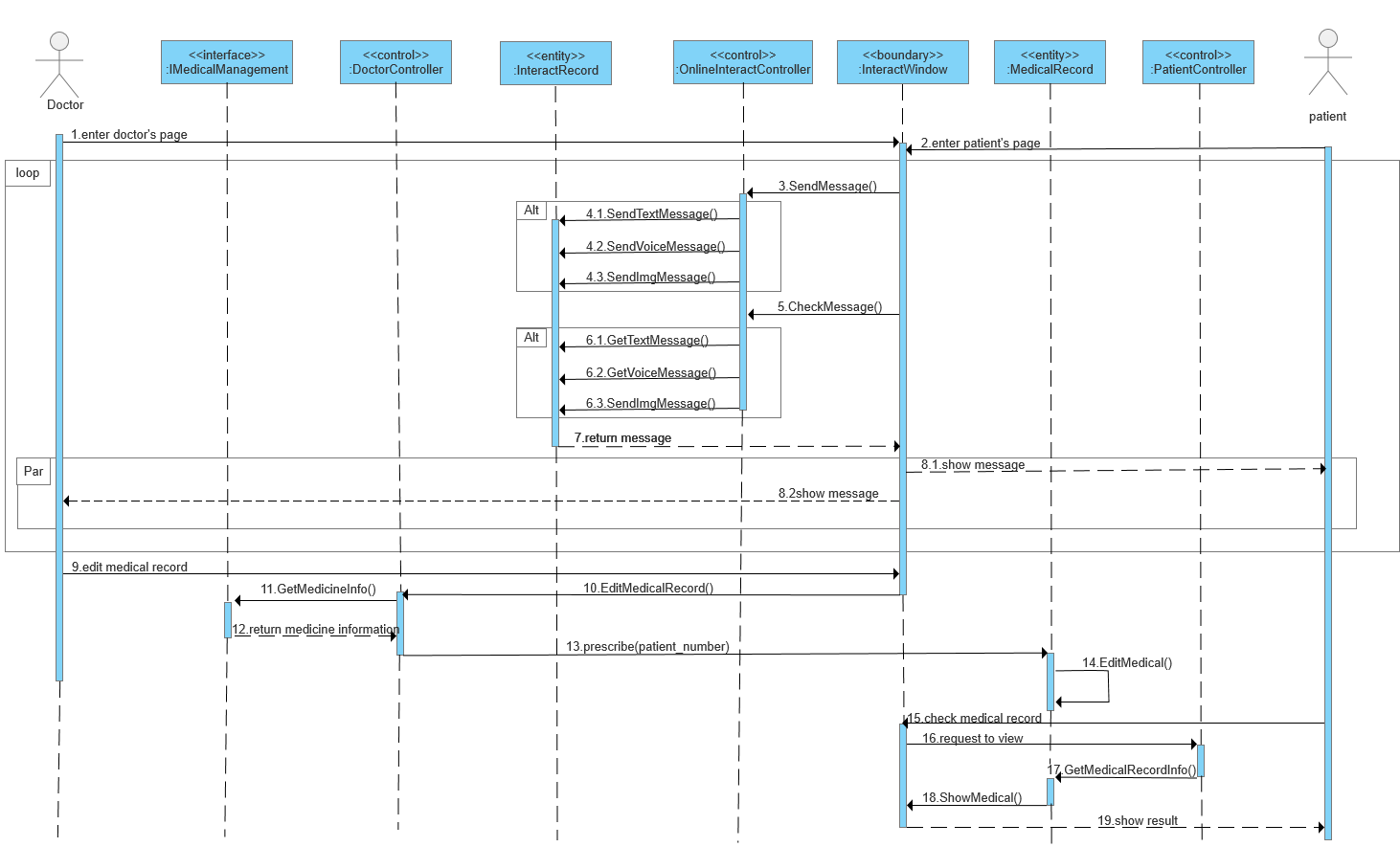




该时序图具体阐述了预约挂号用例实现时系统内部的交互流程。该用例主要基于就诊服务系统，同时引入了在线支付子系统接口。具体交互流程如下：1.用户与任务页面交互，完成挂号操作。2.任务控制器向任务、队列实体提交任务信息，队列、任务信息更新。3.用户进入在线支付子系统完成支付。4.队列接收医生控制器信息不断更新，用户随时可以查看排队信息，也可以取消预约。5.医生控制器发出叫号信息，任务状态更新为完成，并反馈给用户，提醒用户就医

#### 医患交流用例实现





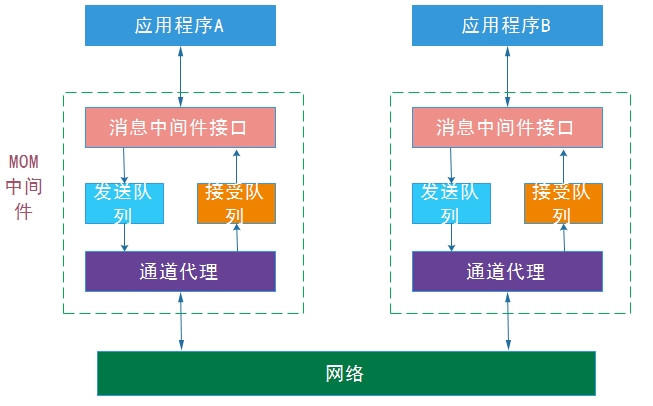
该时序图具体阐述了线上问诊用例实现时系统内部的交互流程。该用例主要基于医患交流系统，同时引入了药品管理子系统接口。具体交互流程如下：1.医生用户进入交互界面开始问诊，系统将双方交互信息存储到交流记录实体中，并不断反馈到交互界面。2.完成问诊，医生开始修改病历，医生控制器与药品管理子系统交互，完成开药操作，并更新患者病历。3.患者查看问诊记录报告与病历，患者控制器控制病历实体信息显示到界面上，反馈给患者。

### 设计模型中的问题

#### 排队信息更新的问题

在预约挂号的用例中，我们采用的是直接通过更新数据库中队列实体信息的方式来更新具体的排队信息的。但是这种方式往往存在许多问题（更新速度慢、同步信息问题等），比较影响系统的性能。

通过调研，我们了解到，当前排队信息的更新，通常是使用一个中间件消息队列的方式来解决的。消息队列是指利用高效可靠的消息传递机制进行与平台无关的数据交流，并基于数据通信 来进行分布式系统的集成。当前使用较多的消息队列有RabbitMQ、RocketMQ、ActiveMQ、Kafka、ZeroMQ、MetaMQ 等，而部分数据库如 Redis、MySQL 以及 phxsql 也可实现消息队列的功能。



#### 架构设计存在的问题

在架构设计当中，我们对于整体的系统进行了粗略的设计。但是受限于知识储备，以及对于一些具体实现的技术、平台、框架的不熟悉，我们选用的一些具体实现技术会存在一定程度的问题。例如：

1. 不同技术之间的兼容性问题。有一些技术可能由于环境设置、功能相冲突等原因，导致出现技术之间的冲突，在我们选择制作架构图时并没有很好的考虑到。
2. 可能采用了一些冗余的框架技术。在我们的系统中，有一些需求可能不是十分的必要，但在我们的架构图中依然添加了对应的技术实现。

### 反思

莫健文：在本课程中深切体会到一个项目的分析设计的流程和步骤，从最开始的一点点构想到用例模型设计，再到类图实现，接口实现，逐渐将系统的架构自顶向下一步步刨析清楚。再UI设计的过程中，也自学到一些界面设计的方法，从草图绘制到低保真的界面，最后再设计成高保真的界面，也是一步步迭代更新，才有最后的成品。

崔敬文：对类图的设计有了更深入的认识的学习和体会。在类图的更新以及与子系统对接的过程中，对于整个系统的逻辑层次以及一些实现方面的细节有了更清晰的认知。进一步熟练掌握starUML软件绘制类图的过程。

袁枫烨：在本次课程中，深刻体会到了一个项目从无到有的分析设计过程，了解并掌握了UML建模的具体流程与操作，对于一个项目的设计方式、设计逻辑都有了新的理解，并逐渐学会如何分析、设计、搭建项目的架构。另外，我还接触了许多具体实现的平台与框架，学习了很多新的概念，宏观上对系统分析与设计有了更深入的了解。

陈敬麒：深刻体会了软件架构的设计流程，对，积累了从零搭建项目架构的经验，了解到了微服务架构由小型自主服务组成且各个组成部分之间松耦合的优势，以及各个层次的技术、SpringCloud微服务框架；对类图的设计有了更清晰的了解，更进一步通过划分实体类，控制类，边界类的设计了系统的逻辑结构。

### 8 Assigment3小组分工

莫健文：子系统和接口设计、文档归纳整理

崔敬文：类图设计与更新

袁枫烨：实现平台与框架资料收集、架构改进设计、时序图更新

陈敬麒：类图设计与更新、架构图设计