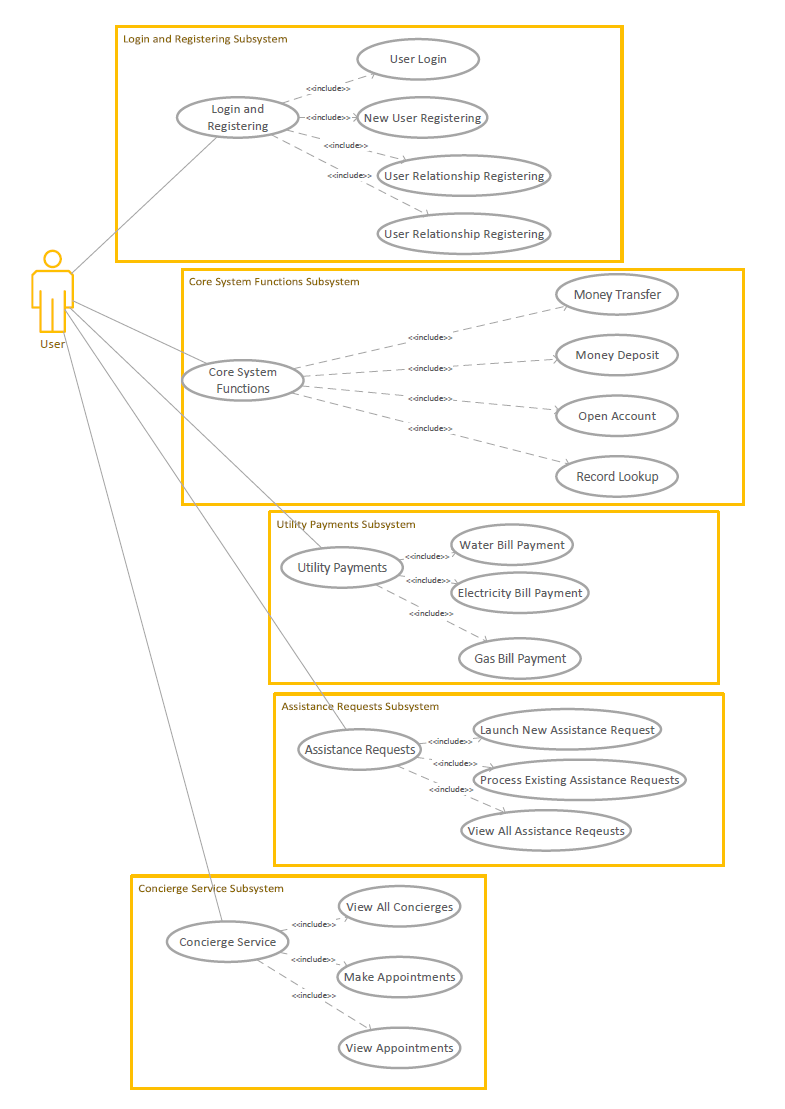
**技术文档**

**一、需求分析与说明**

系统用例图



1. **登录和注册**

在我们的网上银行系统中，用户可以在首页轻松登录和注册账户。

1. **用户登录**

用户在空表中输入自己的身份证号和密码，然后他们会得到一个新的令牌存储在本地存储中，该令牌将在 15 天后到期。

1. **新用户注册**

用户在空表中输入自己的身份证号、密码和校验密码，然后他们会得到一个新的令牌存储在本地存储中，该令牌将在 15 天后过期。

1. **用户关系注册**

儿子/女儿填写关系登记表，他/她的父母将获得信息。

1. **用户关系确认**

父母收到确认并决定是否接受。

1. **核心系统功能**
2. **汇款和存款**

用户通过系统模拟进行存款、存款和转账。

1. **开户口**

用户可以通过系统添加新的银行卡账户。

1. **记录查找**

用户可以查看所有交易记录，包括存款、取款、转账、支付等流量信息。

1. **公用事业付款**

在我们的网上银行系统中，用户可以使用有效的银行账户方便地支付水电费。

1. **水费缴付**

用户使用他们现有的银行账户之一，该账户有足够的余额来支付他们的水费。

1. **电费支付**

用户可以使用他们现有的银行账户之一，该账户有足够的余额来支付电费。

1. **煤气费支付**

用户可以使用他们现有的银行账户之一，该账户有足够的余额来支付他们的煤气费。

1. **协助请求**

通过发起协助请求，用户可以帮助其注册亲属使用某些系统功能，例如支付水电费和进行转账。实际受益人的亲属发起的援助请求将被系统存储，并在实际受益人批准后，通过即时消息呈现给实际受益人。最终，援助请求的实际受益人可以批准或拒绝该请求。如果协助请求被批准，则相应的系统功能实际上将由标准业务逻辑执行。否则，拒绝或未处理的协助请求将与标签一起存储在数据库中。

1. **发起协助请求**

我们的网上银行系统目前有两种可用的援助请求类型，即水电费支付援助和汇款援助。

1. **发起新的水电费支付援助请求**

用户只需提供房主的身份证号码，即可在我们的系统中启动新的水电费支付。此外，还需要有效的银行账户信息。

1. **发起新的汇款援助请求**

用户可以通过提供将其资金转出该账户的账户所有者的 ID 号，在我们的系统中发起新的转账协助请求。此外，还需要两个有效的银行账户信息。

1. **指定亲属提出的协助请求**

每当用户的注册亲属为其发起援助请求时，用户都会收到有关该请求的消息。然后，用户将在协助请求处理页面上看到此请求。然后用户可以选择是否批准此请求。如果选择是，则该请求实际上将成为系统中的有效操作，将执行所有相应的业务逻辑。但是，如果用户认为这个帮助请求有问题或不恰当，他或她可以简单地拒绝这个请求，系统将忽略请求中打算执行的操作。

1. **指定亲属生成的所有协助请求**

用户可以查看他们曾经发生过的所有帮助请求，无论它们是否被批准。

1. **礼宾服务**
2. 查看所有礼宾

当您查询礼宾列表时，您将了解有关服务的礼宾的大量信息。

1. 与礼宾部预约

通过启动礼宾服务，用户可以根据时间、性别和语言偏好与礼宾预约。

1. 查看约会

预约后，此页面将显示您的预约信息。

**二、架构设计**

1. **系统架构设计**

系统架构设计分为三个大层次，通过层次结构的有效划分，可以充分展示整个应用系统的设计思想。

**用户层**

用户层是指使用系统通过有线或无线网络与服务层进行通信以执行相应工作的各类用户。本系统的用户层包括终端显示层。终端展示层是指系统通过PC终端向用户展示相应的数据、服务、应用程序和协议，这些数据和服务需要通过服务层中的请求处理层请求来获取。该系统的用户主要是能够简单操作电脑但不能熟练使用应用程序的老年人。前端以 Vue 为框架，通过axios与后端交互，以 Json 格式传输信息，完成基本逻辑功能。

**Tomcat 服务器**

系统布置在Tomcat Web Server上，整体采用SpringBoot框架，分为请求处理层、业务逻辑层和数据访问层三个层次，采用消息队列中间件。

**请求处理层**

请求处理层也可以称为Web层，负责接收终端显示层的请求并响应终端显示层。终端展示层使用HTTP协议发送请求，请求处理层接收HTTP请求并完成响应。 Spring Controller负责接收请求，同时调用业务逻辑层进行业务处理，然后将处理结果响应终端展示层。

**业务逻辑层**

业务逻辑层也可以称为Service层，体现了系统架构的核心价值。该层使用SpringBoot框架，通过调用数据访问层接口来实现数据处理，同时提供了请求处理层的接口，实现了previous和next之间的联系。该层是实现业务逻辑和提供服务功能的主要层。

**数据访问层**

数据访问层也可以称为Dao层，它是一个抽象层，便于应用程序进行数据读写访问。该层使用Mybatis框架，可以通过Mapper层实现数据库的CRUD，解决访问数据库的问题。

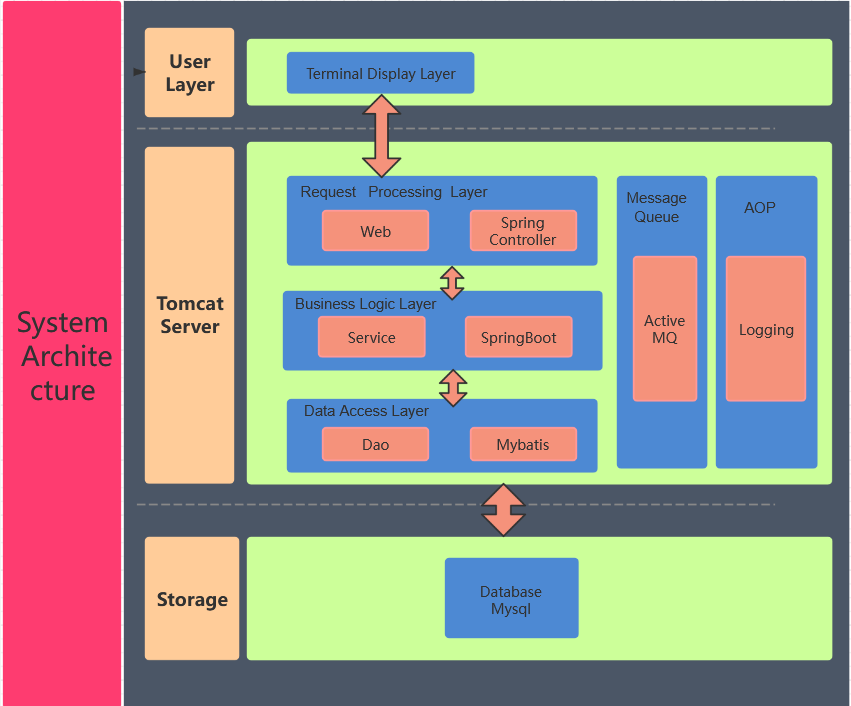
**Java 消息服务**

本系统采用消息队列中间件ActiveMQ，可为系统提供高效灵活的消息同步和异步传输处理、存储转发和可靠传输。

**贮存**

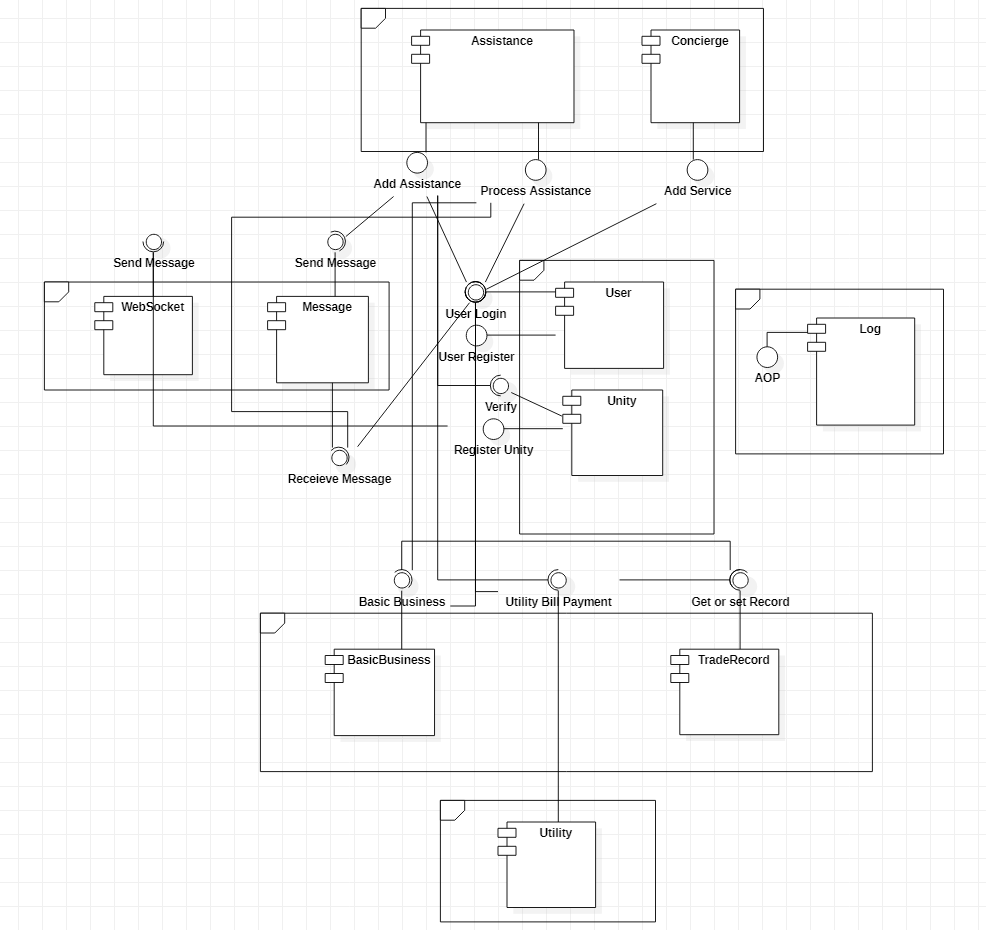
系统数据的存储使用Mysql数据库，完成与数据访问层的交互。

系统架构设计分为三个大层次，通过层次结构的有效划分，可以充分展示整个应用系统的设计思想。



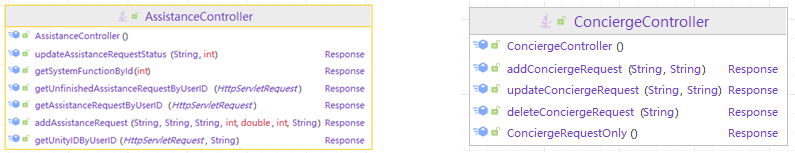
1. **组件设计**

为了更好地组织和实施我们的银行系统，我们进行了详细的组件设计。我们将系统划分为六大子系统，即协助子系统、消息子系统、用户子系统、基础业务系统、公用事业支付系统和日志子系统。每个特定子系统都包含几个组件来帮助实现其功能，并且来自不同子系统的所有组件可以通过一系列接口相互协作。



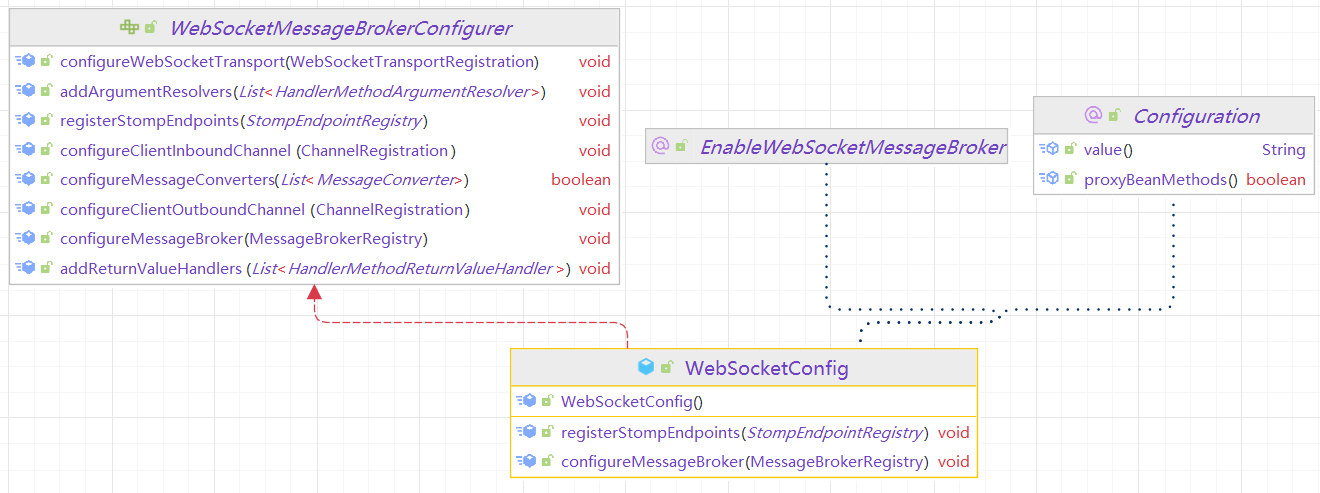
1. **辅助子系统**

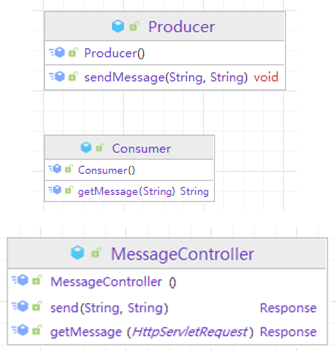
在辅助子系统中，有两个主要组件：辅助组件和礼宾组件。协助组件负责处理用户的协助请求并帮助用户处理。 concierge 组件主要负责用户在办理银行业务时预订礼宾服务的请求。

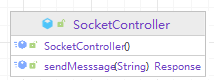


1. **消息子系统**

消息子系统中有两个组件，即消息组件和websocket组件。消息组件利用activemq实现JMS服务。每次用户发起帮助请求时，消息组件都会向目标用户发送消息，通知他或她有待处理的请求。此外，除了 ActiveMQ 的消息队列之外，我们还使用 WebSocket 作为一种工具，在他们的亲戚试图发起帐户关联请求时向指定用户发送消息。

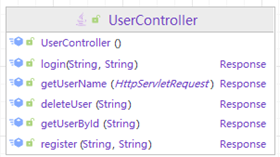


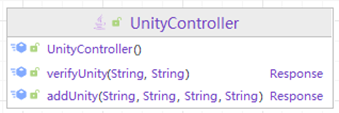




1. **用户子系统**

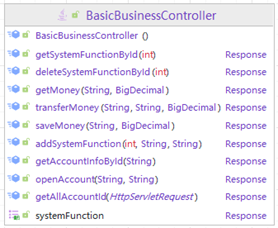
用户子系统包含两个组件。用户组件主要负责普通用户的登录和注册，以及用户token的管理。 unity组件负责账户关联的注册。只有关联账户中的账户才允许发起其注册亲属可以看到的援助请求。





1. **基础业务子系统**

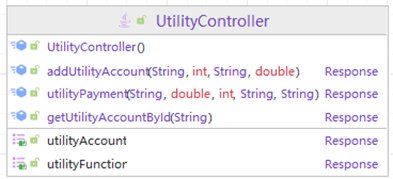
在基本业务子系统的上下文中有两个组件。基本业务子系统负责处理常见的银行服务，如汇款或网上开户。交易记录组件负责在用户每次在系统中进行支付时生成用户的交易记录。





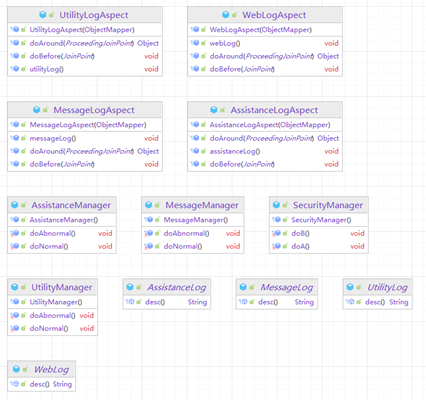
1. **公用事业支付子系统**

公用事业支付子系统包含一个单一的组件，即公用事业组件。公用事业组件负责处理用户支付水费、电费或煤气费等公用事业费用的请求。



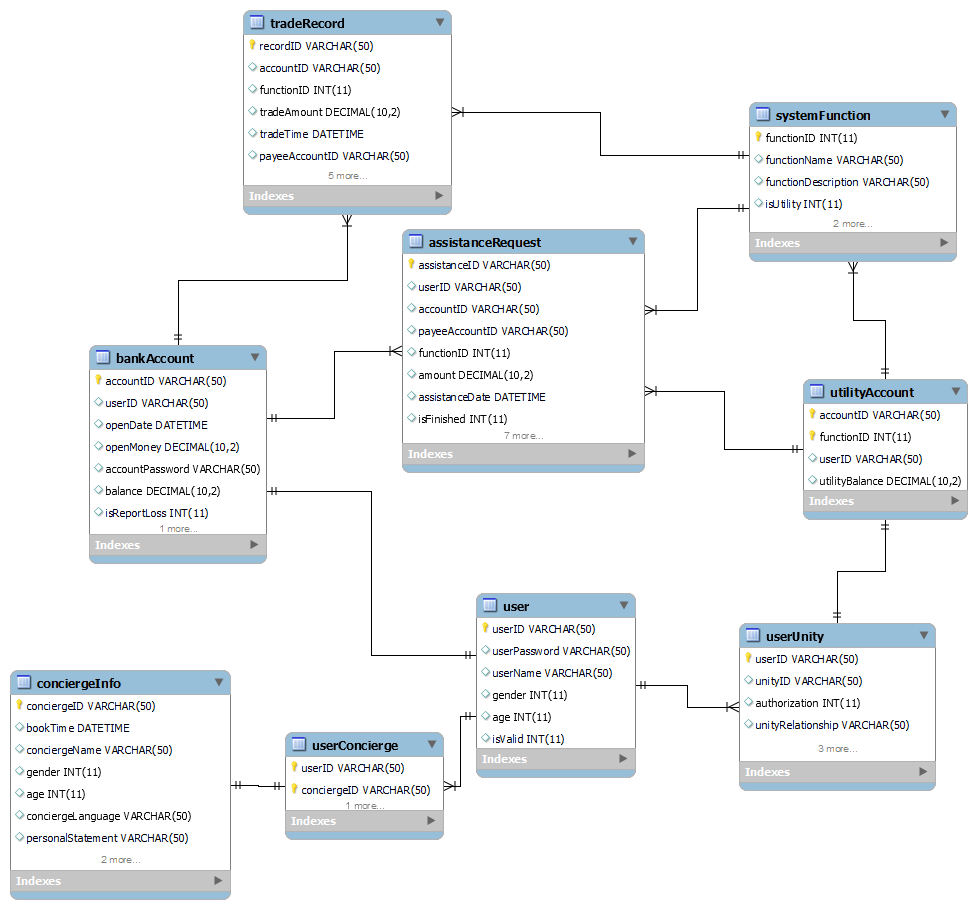
1. **日志子系统**

日志子系统包含负责系统日志记录的日志组件。由于面向方面的编程方法，我们能够记录其url 、IP 地址、请求持续时间甚至参数的每个请求。



1. **数据库设计（ER图）**

本系统使用的数据库为MySQL数据库，表设计及关系见下图。



**三、其他技术细节**

1. **系统部署**
2. **Springboot部署**

感谢阿里云的 Elestic Computing Service，这个系统的后端服务连同MySql数据库和ActiveMQ中间件被部署到IP地址为的服务器上

106.14.25.240。

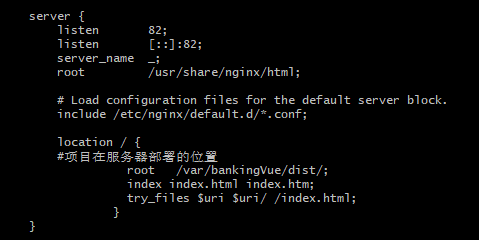


我们已经使用 Maven 打包了我们的Springboot服务，并将包上传到了服务器上。为了保持Springboot服务运行，我们使用了 Nohup来运行包。



1. **Vue部署**

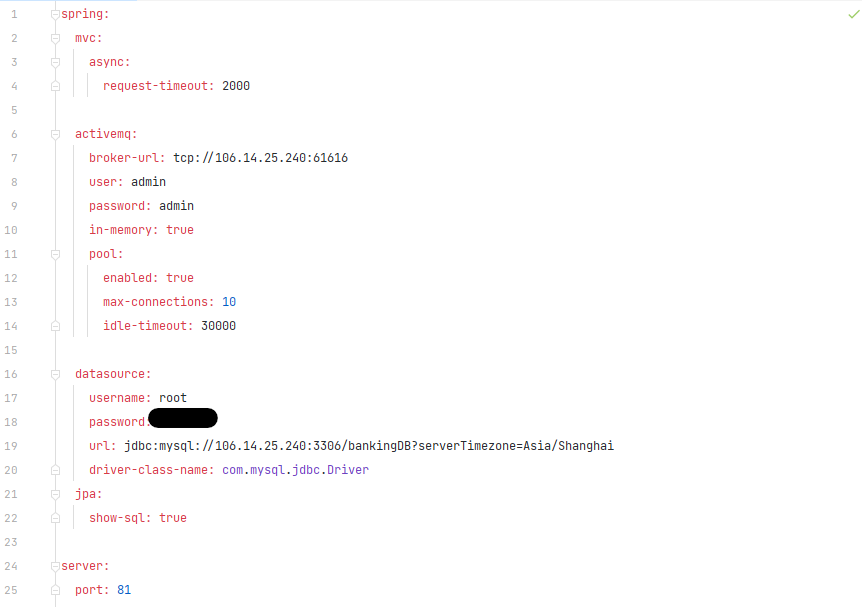
我们还使用 Nginx 将我们的网站部署到了远程服务器。用户可以通过82端口使用服务器的IP地址访问我们的网站。



1. **系统配置**

由于整个Springboot包以及我们系统的后端和前端的Vue包已经部署完毕，我们的系统也进行了相应的配置。

在服务器端，61616 端口是为activemq Java 消息服务保留的。端口 3306 为mySql数据库保留。并且端口 81 保留用于 Web 请求。此外，端口 82 保留用于访问我们的网页。



1. **使用 Java 消息服务传递用户通知**

如上所述，每当用户的注册亲属向用户发起新的协助请求时，被协助的用户都会收到一条消息，通知他或她有新的请求等待处理。这要归功于强大的 Java 消息服务。



为了实现这个功能，我们使用了activemq作为中间件，为每个用户创建了一个消息队列。每次为用户发起新的帮助请求时，都会将一条消息传递到相应的消息队列中。然后，此请求的受益人将在他或她打开网页以处理援助请求时自动收到此消息。

1. **Logging中面向切面编程的实现**

由于银行系统的特殊性，日志记录至关重要。所以我们引入了面向方面的编程方法来实现每个重要接口的日志记录。也就是说，每当用户从网页触发请求时，我们的服务器上都会出现相应的日志，记录包括用户IP地址、请求URL和请求参数在内的重要信息。

Generated

我们已经编写了 Before 和 Around 方法来根据它们的参数监控请求。如果应检测到异常的空参数或异常参数，则会出现日志记录中的警告。

Generated

1. **代币的使用**

作为使我们的网站更安全的一项措施，我们使用令牌来保护用户活动。用户登录时会向用户发放一个令牌，其中包含用户名和时间戳信息。每次用户从网站发出请求时，都需要token里面的信息来确保发出请求的用户是合法登录的。我们使用了强大的工具Hutool从后端生成用户token，并且一旦用户获得令牌，令牌将存储在网络浏览器的本地存储中，直到用户注销或令牌本身过期。



1. **AES加密的使用**

在我们的银行系统中，有许多用户场景需要密码来验证付款。因此，始终需要将密码传输到后端以与存储在数据库中的真实密码进行比较。为了确保密码在传输和存储过程中的安全，我们在密码发送到后端之前使用 AES 加密对密码进行了加密。结果，存储在数据库中的用户密码也被加密了。我们充分利用了强大的工具crypto来实现这个功能。



1. **交易的使用**

Spring 支持程序化事务管理和声明式事务管理。声明式事务管理基于 AOP。其本质是拦截方法前后，然后在目标方法之前创建或添加事务，并在目标方法执行后根据执行情况提交或回滚事务。声明式事务只需要在配置文件中（或通过@Transactional注解）进行相关的事务规则声明，即可将事务规则应用到业务逻辑中。

声明式事务管理使业务代码免受污染。与程序化事务相比，声明式事务唯一的缺点是后者的最细粒度只能应用于方法级别，而不能像程序事务一样应用于代码块级别。

声明式事务管理也有两种常见的方式，一种是基于tx和aop命名空间的 xml 配置文件，另一种是基于 @Transactional 注解。这里使用@Transactional，简单清爽。

1. **Websocket的使用**

由于每个用户如果想从援助服务中受益，都需要将他们的帐户与其指定的亲属关联，因此我们使用 Web 套接字作为一种工具来向其帐户关联的用户发送消息。一旦用户发起了将其帐户与另一个用户关联的请求，另一端的用户将立即收到一条消息，通知他或她即将到来的请求。