**概要设计规约**

**修订历史记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编写日期 | 说明 | 作者 | 评审时间 | 评审人员 |
| 2021/12/4 | 概要设计引言编写 | 蒋汶霖 | 12.5 | 李林洲 |
| 2021/12/5 | 总体架构编写 | 吴渭 | 12.5 | 蒋汶霖 |
| 2021/12/6 | 界面设计 | 李林洲 | 12.6 | 何子硕 |
| 2021/12/7 | 数据库初步设计 | 蒋汶霖、吴渭 | 12.8 | 蒋汶霖、吴渭 |
| 2021/12/8 | 数据库设计完善 | 蒋汶霖、吴渭 | 12.9 | 蒋汶霖、吴渭 |
| 2021/12/10 | 系统错误设计 | 何子硕 | 12.10 | 蒋汶霖 |

1. **引言**

**1.1. 概要设计依据**

本概要设计规约依据前期编写的需求规约和需求分析规约进行编写

**1.2. 参考资料**

参考设计建模书籍：SoftwareEngineering-A Practitioner's Approach.8thEdition.RogerPressman.2014

本项目已完成的文件：需求规约、需求分析规约

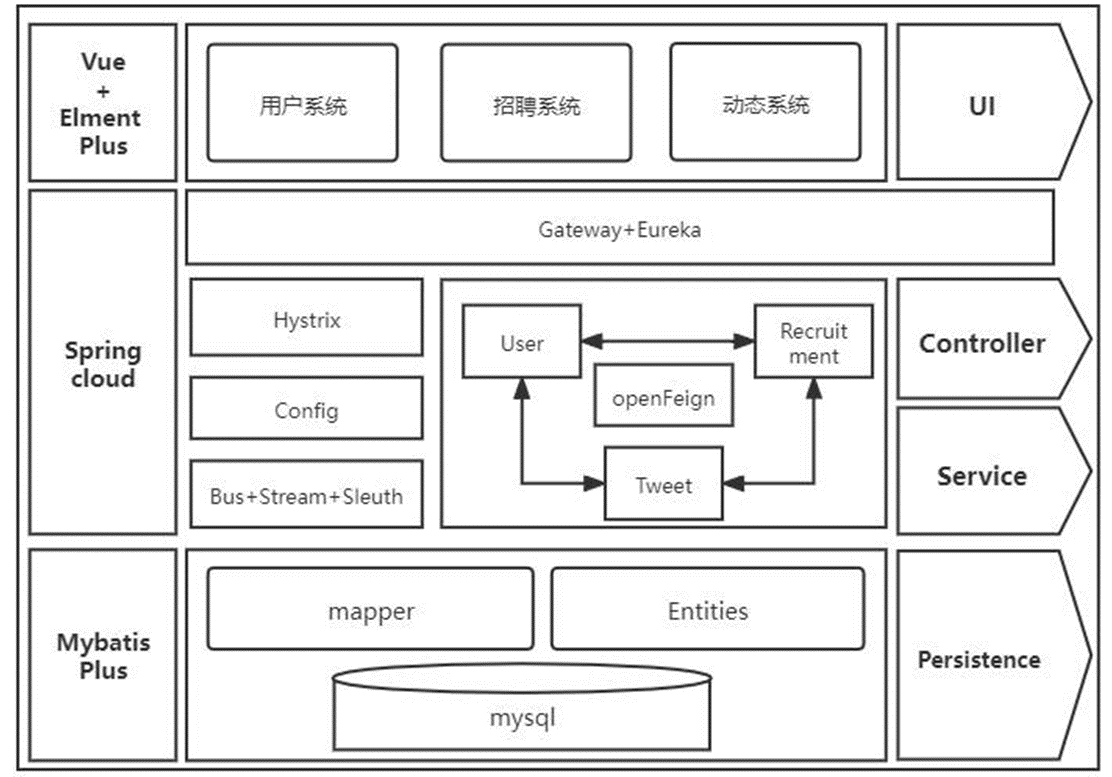
GB/T 8567-2006《计算机软件文档编制规范》

**1.3. 假定和约束**

|  |  |
| --- | --- |
| 经费限制 | 3台服务器租赁费用<1000/年 |
| 开发期限 | 10.29-12.30（教学周第八周周五-第十七周周四） |
| 设备条件 | 四台工作电脑 |
| 用户资料准备 | 从数据超市中获取数据，或采用python从现有招聘网站中爬取数据 |
| 交流问题 | 采用飞书团队和qq微信线上交流与线下交流结合 |
| 编程语言 | 后端Java和springcloud框架，前端采用vue脚手架和elementUI |
| 运行环境 | 服务端JVM和node.js，客户端使用浏览器访问 |
| 人力资源 | 2名前端人员，2名后端人员 |

1. **概要设计**

**2.1. 系统总体架构设计**



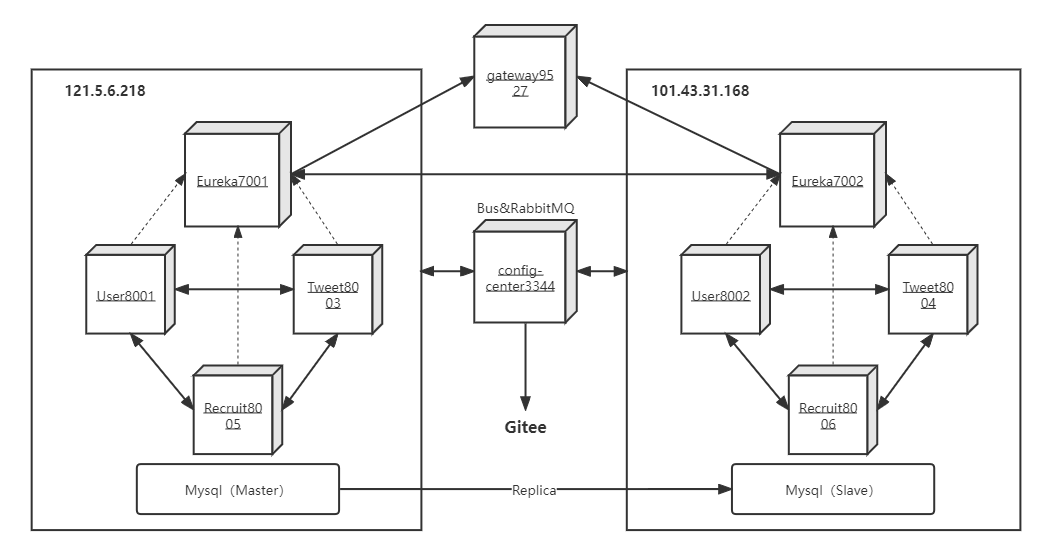
我们的项目架构分为传统的三层架构：

* 数据库持久层：Mysql数据库，采用Mybatis-plus作为ORM框架
* 服务层：按照业务逻辑分为三大模块：用户模块（User），动态模块（Tweet）和招聘模块（Recruitment）
* 表示层：采用Vue响应式框架，辅以Element组件库

**2.2. 系统软件结构设计**

基于spring cloud搭建微服务架构：

* Eureka负责服务注册与治理
* 分布式部署，在101.43.31.168与121.5.68.218两台服务器上部署所有模块，通过两个Eureka相互注册，构成多节点高可用的服务集群
* OpenFeign实现服务间相互调用（自带负载均衡）
* Hystrix实现服务降级与服务熔断
* Config分布式配置中心，从Gitee仓库中实时爬取配置信息
* Bus消息总线和RabbitMQ消息队列，实现配置信息的动态刷新
* Gateway网关，对前端请求做token鉴权与日志记录，转发到服务模块的同时启用轮询负载均衡



数据架构

在两台服务器都启动了Mysql服务，采用**Mysql Replication**主从复制策略，以实现数据库层面的多节点高可用集群。MySQL主从复制涉及到三个线程，一个运行在主节点（log dump thread），其余两个(I/O thread, SQL thread)运行在从节点：

* 主节点 **binary log dump** 线程

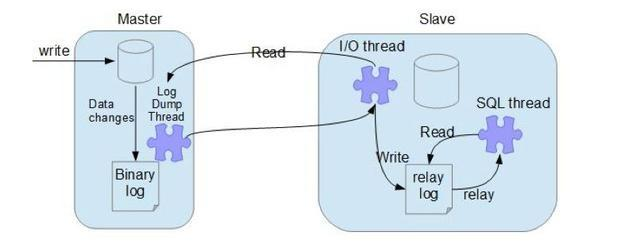
当从节点连接主节点时，主节点会创建一个log dump 线程，用于发送bin-log的内容。在读取bin-log中的操作时，此线程会对主节点上的bin-log加锁，当读取完成，甚至在发动给从节点之前，锁会被释放。

* 从节点 **I/O** 线程

当从节点上执行start slave命令之后，从节点会创建一个I/O线程用来连接主节点，请求主库中更新的bin-log。I/O线程接收到主节点binlog dump 进程发来的更新之后，保存在本地relay-log中。

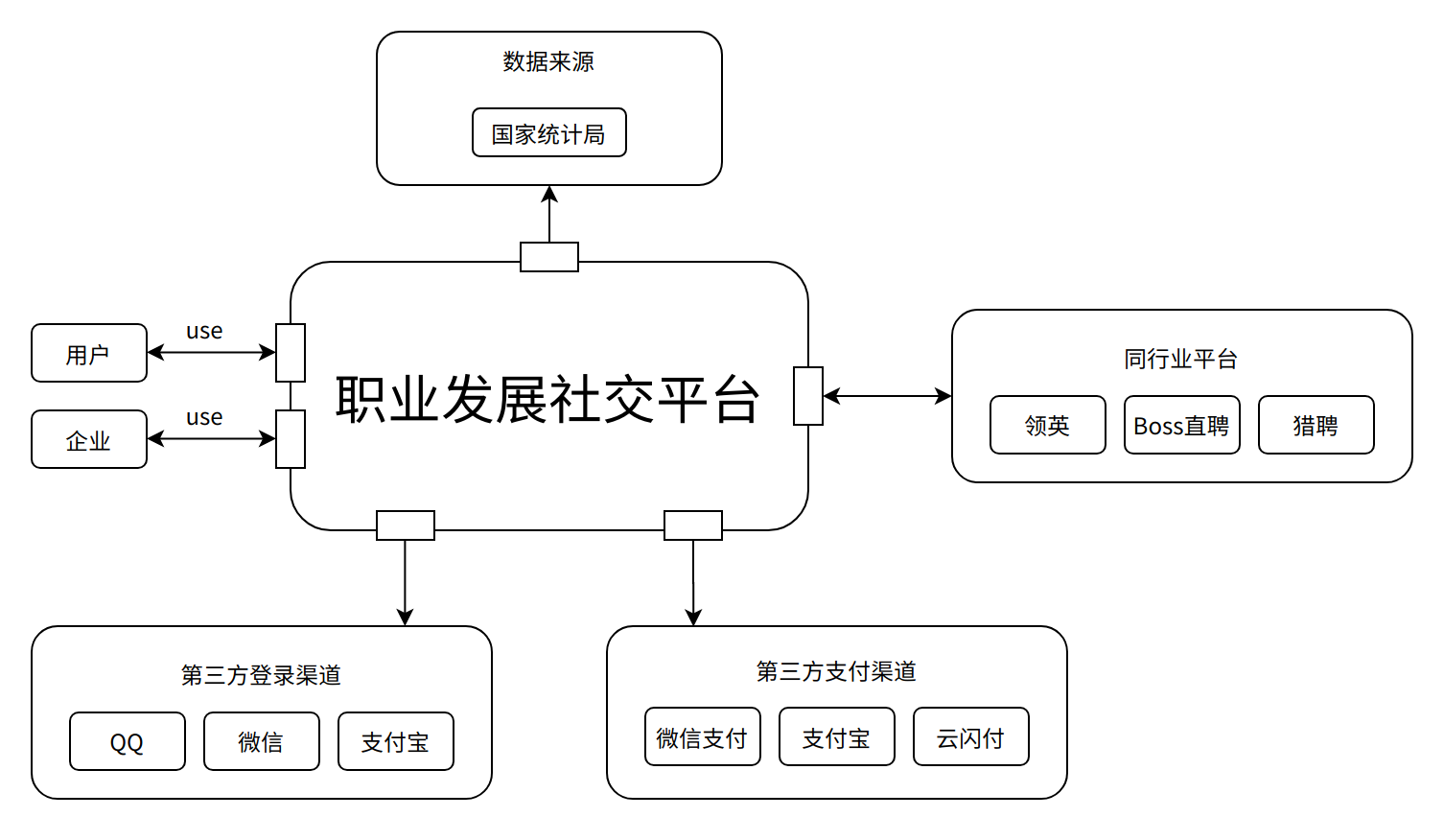
* 从节点 **SQL** 线程

SQL线程负责读取relay log中的内容，解析成具体的操作并执行，最终保证主从数据的一致性。



**2.3. 接口设计**

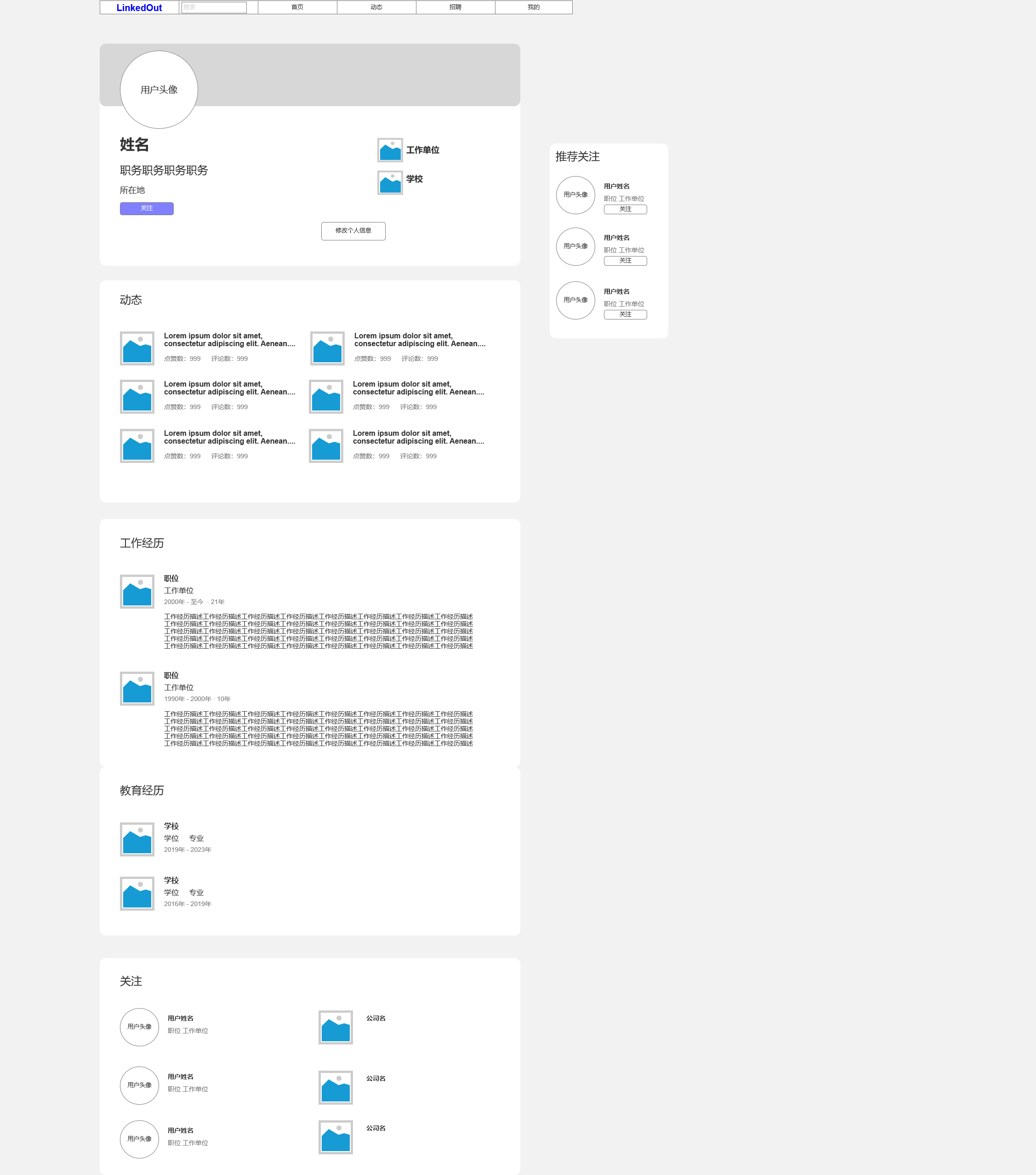
本系统后期需要考虑对接其他已有平台的数据或接口，以吸引用户，为用户提供更加优质的内容，在，项目的初始接口设计阶段应预留相应接口，方便后期对接开发。其中由于平台需要大量的数据来支撑用户的日常使用和吸引企业的入驻，因此需要找到一些数据来源，丰富系统内容。另外由于目前已有一些同类型平台，如果能和这些平台进行数据互通，能够极大提升平台的竞争力。最后一些常用的支付、登录接口也进行相应的预留。



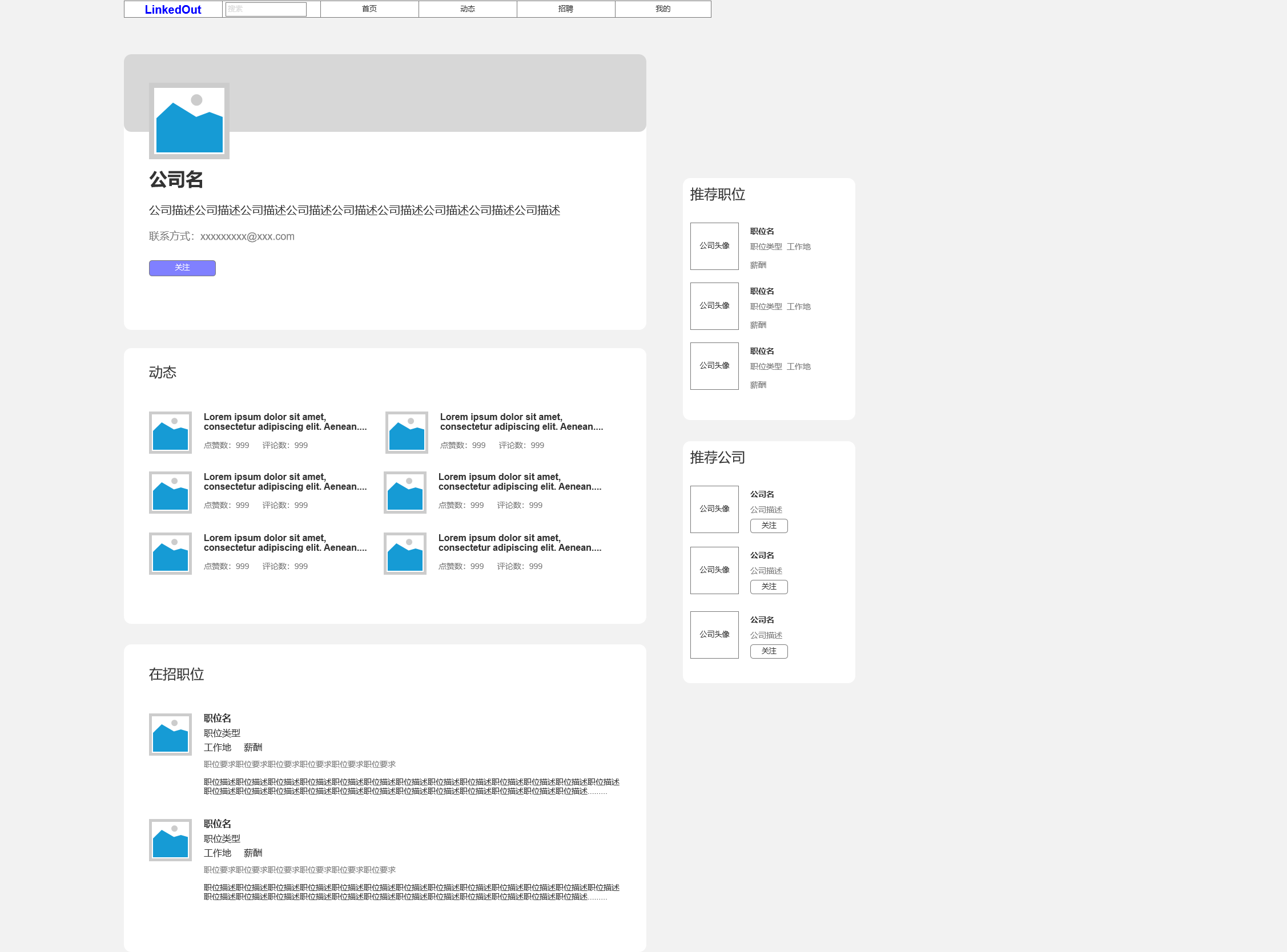
**2.4. 界面设计**

前端我们采用了Axure进行用户界面原型设计，以此来规范前端的开发，减少返工的可能性。我们对几乎每个界面绘制了相应的原型图。

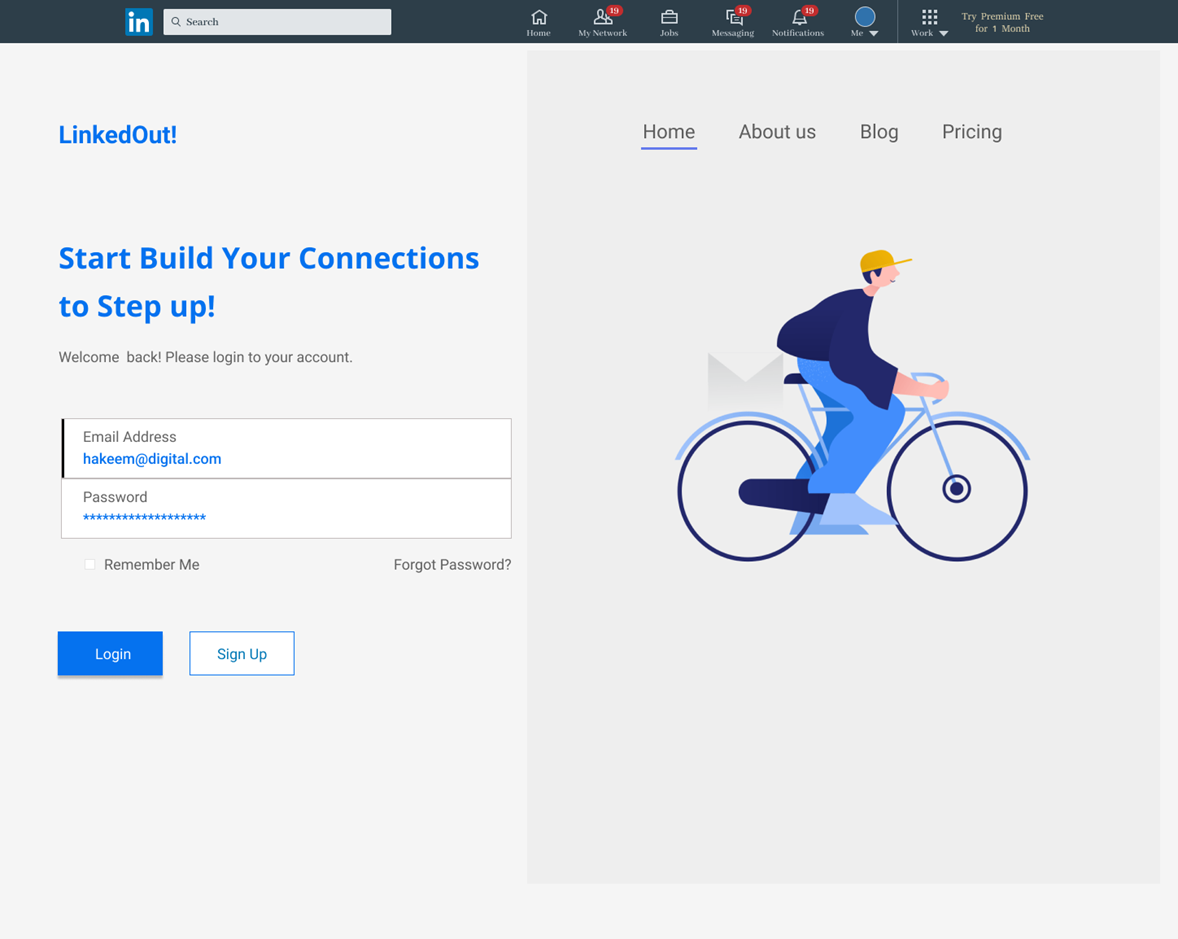
用户个人信息界面展示头像、姓名、动态、工作经历、教育经历等信息。



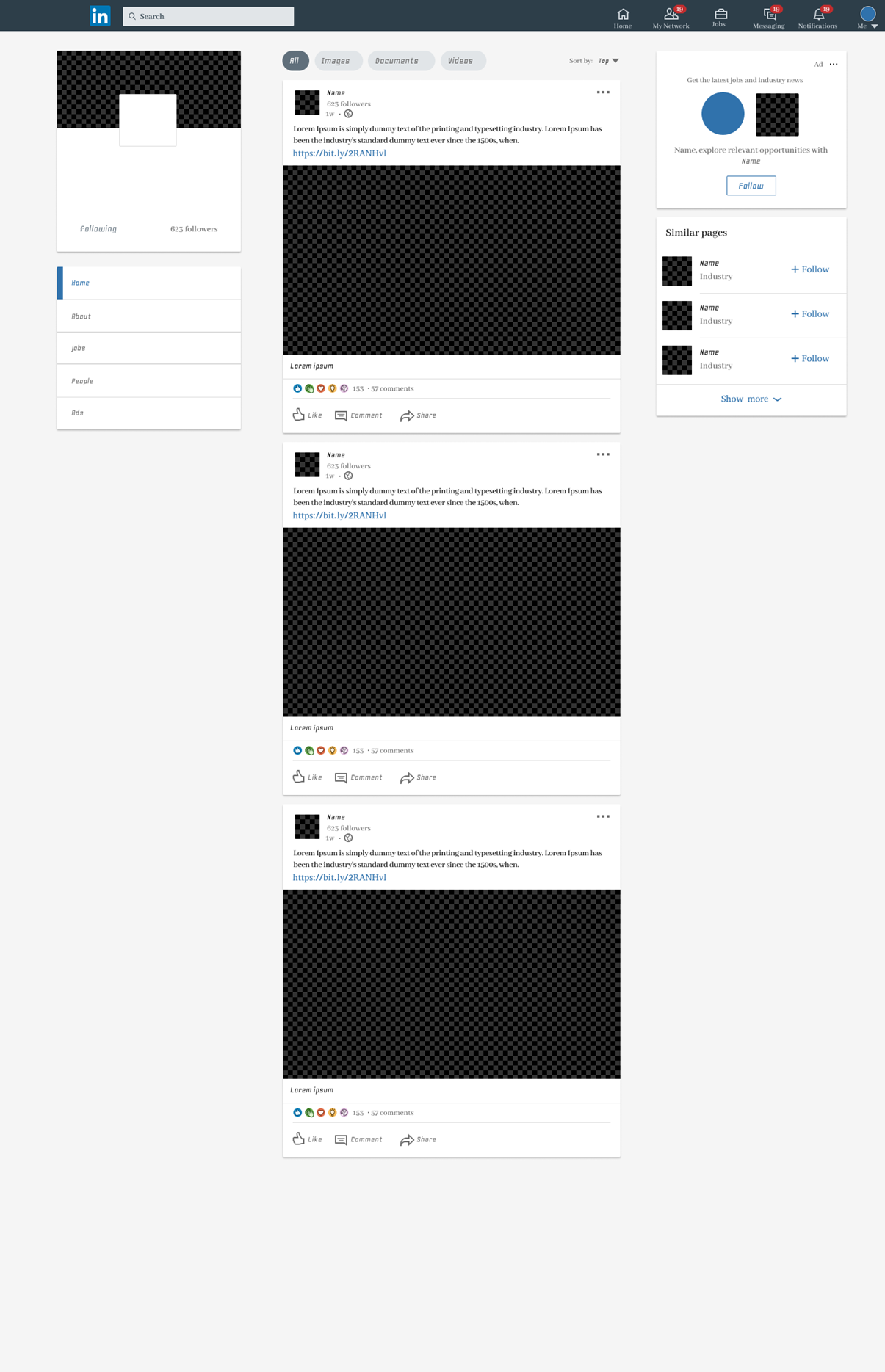
企业用户个人信息界面展示企业用户名、动态、发布的岗位等信息



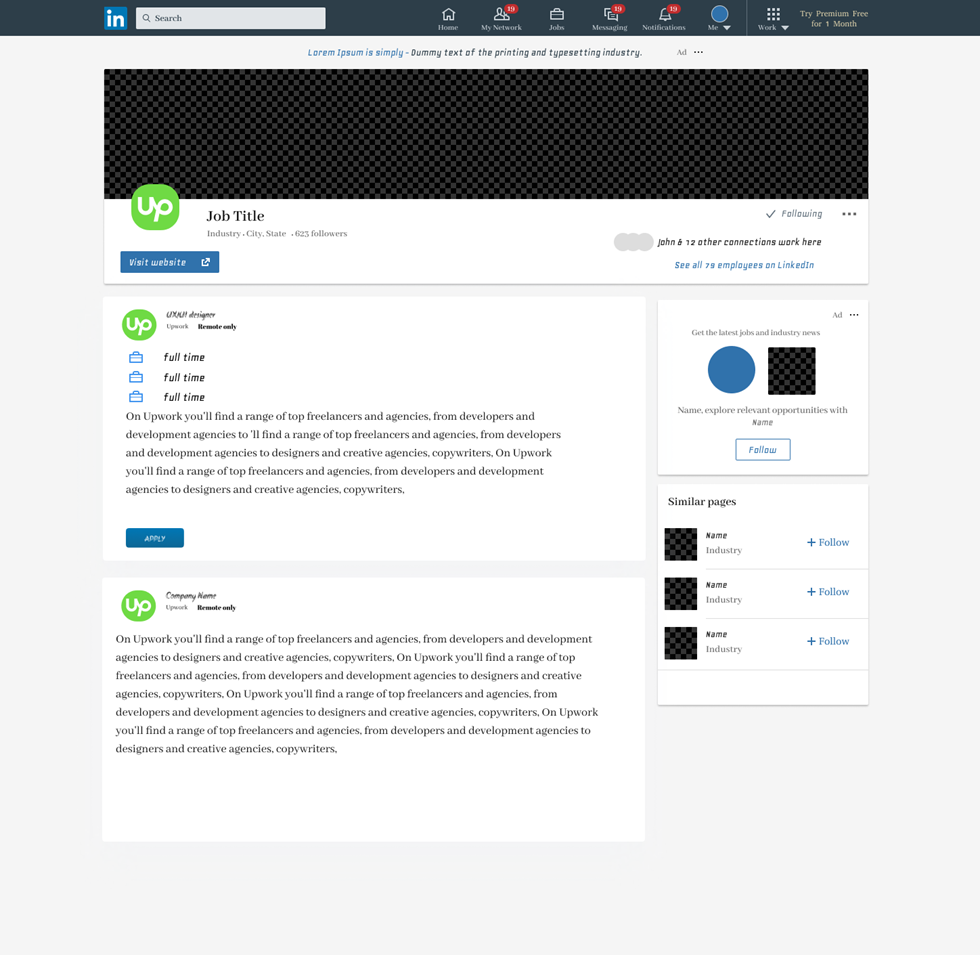
登陆界面与注册界面



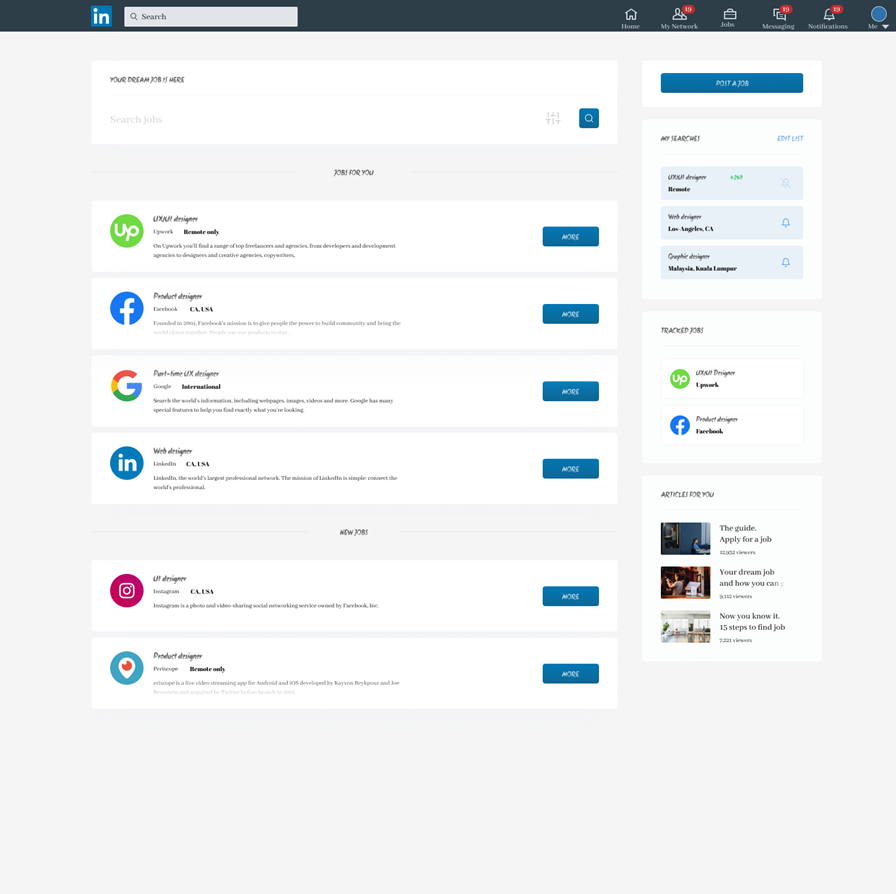
主页展示关注者的动态等最新信息



工作岗位详情界面展示该岗位的各种相关信息

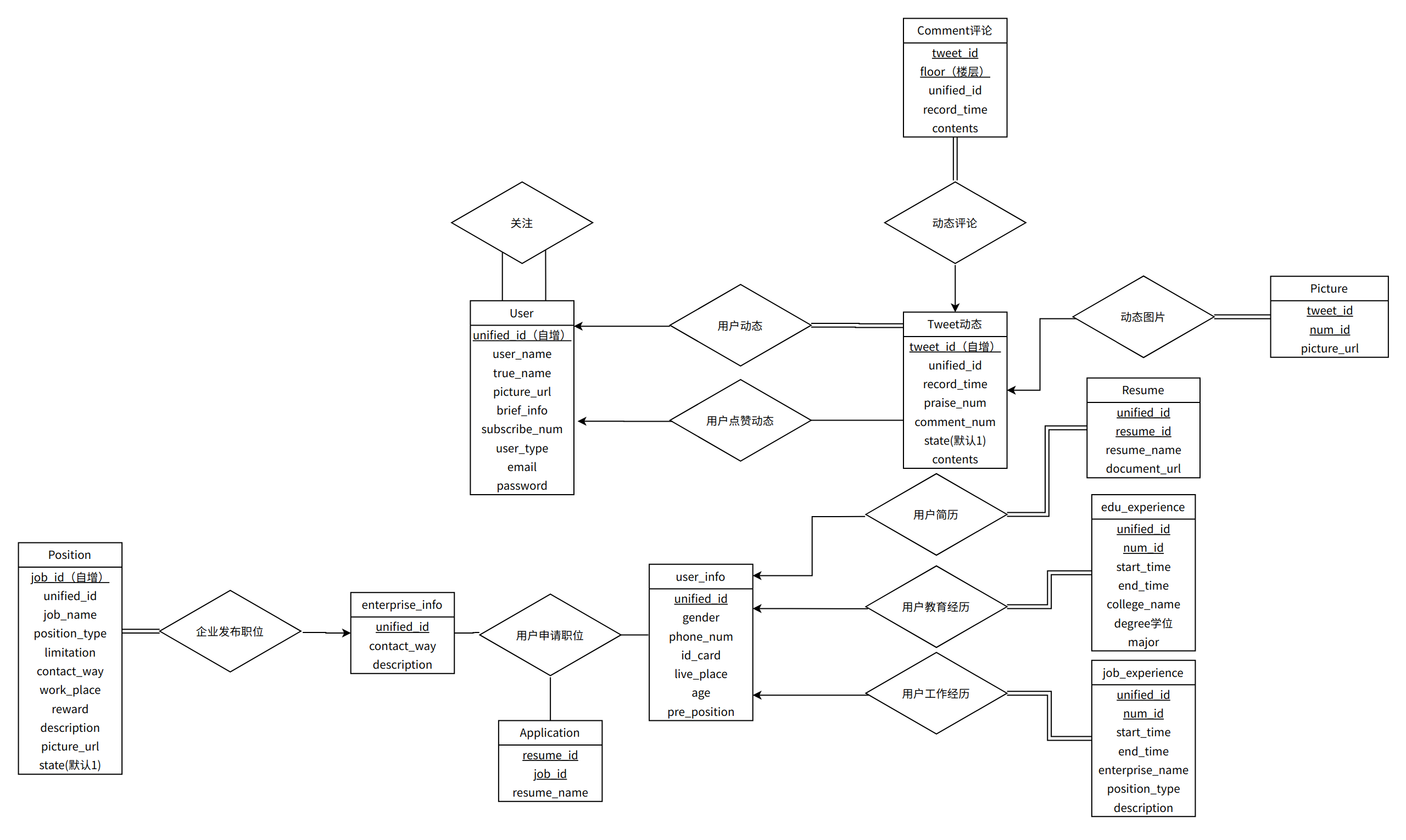


搜索界面展示搜索职位或用户的结果信息



**2.5 数据库设计**

依据“软件需求（分析）规约（说明书）”的数据建模进行数据库逻辑设计和物理设计。



**chat\_record**

职责：存储用户间的聊天信息

内容：聊天内容、消息时间、聊天消息id、用户id1、用户id2

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| unified\_id1 | int | 外码 |
| unified\_id2 | int | 外码 |
| chat\_id | int | 主码 |
| contents | varchar |  |
| record\_time | datetime |  |

**comment**

职责：存储用户动态下的动态评论信息

内容：动态id、楼层数、用户id、评论内容、评论时间

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| tweet\_id | int | 外码、主码 |
| floor | int | 主码 |
| unified\_id | int | 外码 |
| record\_time | datetime |  |
| contents | varchar |  |

**edu\_experience**

职责：存储用户的教育经历

内容：用户id、教育经历id、学校名称、学位、主修专业

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| unified\_id | int | 主码 |
| num\_id | int | 主码 |
| college\_name | varchar |  |
| degree | varchar |  |
| major | varchar |  |
| start\_time | datetime |  |
| end\_time | datetime |  |

**enterprise\_info**

职责：存储企业信息

内容：用户id、联系方式、公司描述

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| unified\_id | int | 主码 |
| contact\_way | varchar |  |
| description | varchar |  |

**favorite**

职责：存储用户收藏的职位信息

内容：用户id、企业id、企业职位id

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| user\_id | int | 主码 |
| job\_name | varchar | 主码 |
| enterprise\_id | int | 主码 |

**job\_experience**

职责：存储用户的职业经历

内容：用户id、职业经历id、经历描述、公司名称、职业类型、开始时间、结束时间

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| unified\_id | int | 主码，外码 |
| num\_id | int | 主码 |
| description | varchar |  |
| position\_type | varchar |  |
| enterprise\_name | varchar |  |
| start\_time | datetime |  |
| end\_time | datetime |  |

**likes**

职责：存储用户的动态点赞记录

内容：用户id、动态id

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| unified\_id | int | 主码，外码 |
| tweet\_id | int | 主码，外码 |

**picture**

职责：存储动态的图片存放地址

内容：图片url、动态id、图片顺序id

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| picture\_url | varchar |  |
| tweet\_id | int | 主码，外码 |
| num\_id | int | 主码 |

**position**

职责：存储招聘职位信息

内容：图片url、职位状态、职位描述、薪水、工作地点、联系方式

职位类型、职位名称、发布者id、职位id

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| picture\_url | varchar |  |
| state | int |  |
| description | varchar |  |
| reward | varchar |  |
| work\_place | varchar |  |
| contact\_way | varchar |  |
| limitation | varchar |  |
| position\_type | varchar |  |
| job\_name | varchar |  |
| unified\_id | int | 外码 |
| job\_id | int | 主码 |

**resume**

职责：存储用户简历

内容：用户id、简历id、文件url、简历名称

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| resume\_name | varchar |  |
| docement\_url | varchar |  |
| resume\_id | int | 主码 |
| unified\_id | int | 主码，外码 |

**subscription**

职责：存储用户间的关注关系

内容：用户id、被关注用户id

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| unified\_id | int | 主码，外码 |
| subscription\_id | int | 主码，外码 |

**tweet**

职责：存储用户发布的动态详细信息

内容：动态id、用户id、发布时间、点赞数量、动态状态、动态内容、评论数量

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| comment\_num | int |  |
| contents | varchar |  |
| state | int |  |
| praise\_num | int |  |
| record\_time | datetime |  |
| unified\_id | int | 外码 |
| tweet\_id | int | 主码 |

**user**

职责：存储用户的详细信息

内容：用户id、用户名、密码、邮箱、用户类型、头像url、简要介绍、真实姓名

关注数、背景

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| unified\_id | int | 主码 |
| user\_name | varchar |  |
| password | varchar |  |
| email | varchar |  |
| user\_type | varchar |  |
| picture\_url | varchar |  |
| brief\_info | varchar |  |
| true\_name | varchar |  |
| subscribe\_num | int |  |
| background | varchar |  |

**user\_info**

职责：存储个人用户的特有信息

内容：用户id、id卡，职位偏好、年龄、居住地址、电话号码、性别

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| unified\_id | int |  |
| id\_card | varchar |  |
| pre\_position | varchar |  |
| age | int |  |
| live\_place | varchar |  |
| phone\_num | varchar |  |
| gender | varchar |  |

**application**

职责：存储用户已经发送的职位申请

内容：职位id、简历id、用户id、企业id、简历名称

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| job\_id | int | 主码，外码 |
| resume\_id | int | 主码，外码 |
| user\_id | int | 主码，外码 |
| enterprise\_id | int | 主码，外码 |
| resume\_name | varchar | 外码 |

**2.6 系统出错处理设计**

**2.6.1 出错信息**

|  |  |  |
| --- | --- | --- |
| 错误类型 | 子项 | 错误原因 |
| 数据库错误 | 连接异常 | 数据库连接断开 |
|  | 数据库异常 | 数据库崩溃 |
| 系统自定义错误 | 输入异常 | 用户名/密码为空 |
|  | 查找异常 | 搜索为空 |
| 链接错误 | 地址异常 | 网页链接不存在 |
|  | 资源异常 | 静态资源加载错误 |
| 服务端错误 | 前后端交互异常 | 前端发送错误参数 |
|  | 后端异常 | 后端代码报错 |

**2.6.2 补救措施**

1. 数据库错误

服务端返回异常信息，并根据主从数据库集群，及时选派一个从数据库成为新的主数据库，通过数据库热备预防数据库异常。

1. 系统自定义错误

前端利用正则表达式对用户输入进行非空判断，并在输入异常和查找异常时及时利用消息弹窗组件给予用户错误提示。

1. 链接错误

* 地址异常：设置404页面，将所有非本系统指定的路由均重定向至404路径。
* 资源异常：对于无法加载的异常图片，设置默认占位符；对于异常文件，拦截用户访问并返回错误提示。

1. 服务端错误

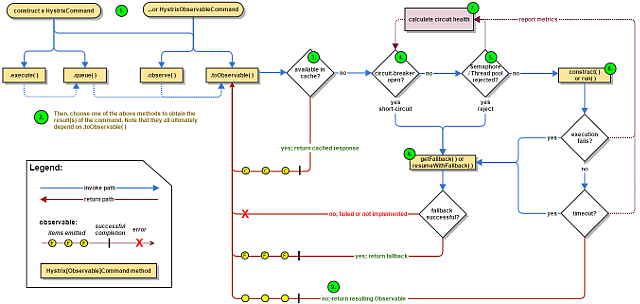
前后端交互异常：设置gateway网关，对所有前端请求做鉴权处理和流量控制，对参数不匹配的请求予以拦截，并返回错误提示。

* 后端异常：利用Hystrix处理雪崩效应，对微服务异常进行服务降级与熔断处理

**2.6.3 系统维护设计**

使用Hystrix中间件，在分布式系统中对服务间的调用进行维护控制，加入一些调用延迟或者依赖故障的容错机制。Hystrix 通过将依赖服务进行资源隔离，进而阻止某个依赖服务出现故障时在整个系统所有的依赖服务调用中进行蔓延；同时Hystrix 还提供故障时的 fallback 降级机制。

* 服务降级：对关键微服务抛出的异常做拦截与服务降级（例如超时异常），返回适当的报错信息
* 服务熔断：在高压环境下对微服务性能进行监测，一旦发现服务异常则启动熔断器机制，截断服务调用，避免雪崩效应，当检测到服务恢复，关闭断路器。同时，依据负载均衡原则，网关将请求转发至未发生服务熔断的服务集群中去。



加入HystrixDashboard子模块作为系统的服务监测组件，对分布式系统中所有的服务状态、请求响应和线程池状态进行实时监控，并及时返回降级与熔断信息：

