|  |  |
| --- | --- |
|  | |
| **《企业大数据征信系统》**  **详细分析与设计**  **V1.0** | |
|  |

目录

[第一部分 引言 3](#_Toc_125)

[第二部分 项目概述 6](#_Toc323127528)

[第三部分 总体设计 6](#_Toc323127529)

[第四部分 界面和业务单设计 10](#_Toc323127539)

[第五部分 单元模块设计 16](#_Toc323127564)

[第六部分 数据库设计 66](#_Toc323128001)

[第七部分 补充设计和说明 69](#_Toc323128036)

# 第一部分 引言

### 一、编写目的

编写本说明书的目的是为了准确阐述项目具体业务需求和需求边界，本说明书的作者是爽划慢填项目组，本说明书的确认者是组长侯博凡，本说明书的读者是项目所有直接干系人。

本说明书是指导项目实施的重要指导性文件，也是用户最后进行验收的依据，说明书中内容一旦确认双方将以此为基础开展工作。如果需要变更说明书内容，必须走变更流程，变更必须得到甲乙双方书面确认，最后变更内容将作为本文的一部分，在项目实施过程中得以体现。

### 二、项目背景

现代市场经济是信用经济，诚信是市场发展的基础。在过去征信系统并不完善，投资者想要了解企业的经营信息与信用状况，需要亲自花费精力查询媒体报道、公司财报、公司借贷记录等信息并进行整合判断，效率低而成本高，且存在敏感信息获取渠道不畅、数据造假等问题。

为了更好地整合企业各方面信息，帮助投资者和政策制定者作出准确判断，提高融资效率，节省社会资源，淘汰劣质企业，促进经济发展，企业大数据征信系统应运而生。在信息技术的基础上，企业大数据征信系统代劳了企业信用信息的收集与分析工作，覆盖的企业范围广，提供了更全面的整合信息，凭借计算机技术为更深层次的数据分析提供了支持，有开发必要。

### 三、定义

#### 1、技术类

从构件化与集成化的角度出发，采用J2EE四层架构的技术方案、构件化的开发模式，应用功能的实现在支撑构件的基础上完成。

1.客户端层（Presentation Layer）： 使用VUE构建前端界面，用于与用户进行交互和展示数据。VUE将用户的操作和请求转化为HTTP请求，发送到后端进行处理。

2. Web层（Web Layer）： Web层充当前后端交互的桥梁，处理来自客户端的请求，并将它们传递给后端进行处理。在这个层次，Spring Boot可以作为你的Web层框架。

控制器层：使用Spring MVC处理HTTP请求，解析请求参数，调用服务层进行业务处理。

服务层：实现业务逻辑，可能涉及到数据处理、数据分析、征信计算等。

数据访问层：与数据库进行交互，执行数据的CRUD操作。

3.业务逻辑层（Business Logic Layer）： 这个层次包含应用程序的核心业务逻辑，将用户的请求转化为具体的业务操作，可能需要与数据库和数据处理组件交互。

服务层：包含业务逻辑，负责协调不同模块的操作，可能涉及到数据处理、业务规则等。

4. 数据访问层（Data Access Layer）：数据访问层负责与数据库进行交互。这个层次涉及到与Hive和Sqoop的交互。

#### 2、业务类

#### **(1)企业征信**

**企业征信**，是指征信机构作为提供信用信息服务的企业，按一定规则合法采集企业、个人的信用信息，有偿提供给经济活动中的贷款方、赊销方、招标方、出租方、保险方等有合法需求的信息使用者，为其了解交易对方的信用状况提供便利。

#### **(2)征信业务**

**征信业务**，是指对企业、事业单位等组织（以下统称企业）的信用信息和个人的信用信息进行采集、整理、保存、加工，并向信息使用者提供的活动。

#### **(3)不良信息**

**不良信息**，是指对信息主体信用状况构成负面影响的下列信息：信息主体在借贷、赊购、担保、租赁、保险、使用信用卡等活动中未按照合同履行义务的信息，对信息主体的行政处罚信息，人民法院判决或者裁定信息主体履行义务以及强制执行的信息，以及国务院征信业监督管理部门规定的其他不良信息。

#### **(4)信用编码**

11315全国企业征信系统为每一家合法注册的企事业单位都匹配独一无二的信用编码，由8位阿拉伯数字组成，终身免费归企业所有，信用编码是企业信用的门牌号码，具有唯一性，不重复，在企业经营，营销推广，赢得市场信赖等方面有广泛的用途。通过信用编码，可以查看包含对应企事业单位的全面信用信息，包括：全国各级政府职能部门公共监管信息；金融机构（包括小额贷款公司、民间借贷等）对企业的信贷评价信息； 行业协会（社团组织）的评价信息；主流媒体对企业发布评价的评价信息；企业管理评估信息及财务信息；市场反馈信息（包括消费者、交易对方、合作伙伴及员工等主体在线实名提交的评价信息）。

#### **(5)信用网址**

根据国务院《征信业管理条例》，利用十多年来积累的大量数据，依法为所有企业建成网上信用档案，每个信用档案均由一组8位数字组成的统一“信用编码”与之唯一对应、永久绑定，“信用编码”+“11315.com”后缀即形成该企业信用档案的“信用网址”，登录该信用网址随时可上网查询其信用档案。

### 四、参考资料

列出有关的参考资料，如：

1. 本项目的经核准的计划任务书或合同、上级机关的批文；

《项目章程》

《里程碑计划》

《开发计划》

《交付物清单》

1. 属于本项目的其他已发表的文件；

《需求说明》

《概要设计》

《API接口文档》

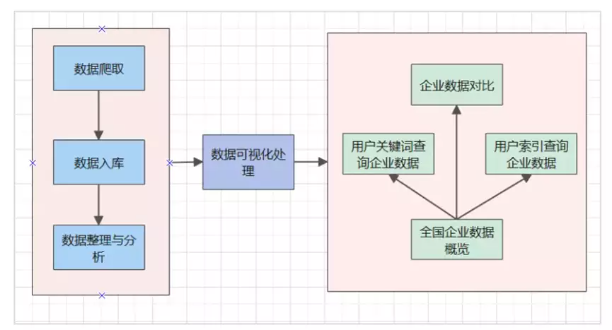
1. 本文件中各处引用到的文件资料，包括所要用到的软件开发标准。

《Java语言编码规范(Java Code Conventions)》

# 第二部分 项目概述

企业大数据征信系统项目旨在建立一个综合性信息平台，通过整合多渠道数据如财务、税务、信用记录等，为金融机构、供应商等提供全面准确的企业信用评估，从而支持决策制定与风险管理。项目包括数据收集、存储管理、分析挖掘等环节，以提升对企业经营状况的洞察力和决策效率。

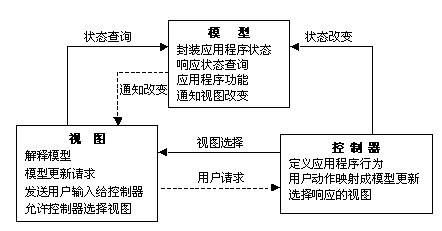
系统从客户实际需求出发，具有一定的灵活性，能适应客户对各种不同种类及经营性质的企业的信用需求，保证网路、硬件、软件和服务体系安全，保证系统数据安全。



# 第三部分 总体设计

#### 1、MVC模型设计

系统实现MVC系统架构，使用SDK4.3标准库，UIView的界面层，UIViewController负责架构结合，NSUserDefaults和CoreData作为模型层进行数据的持久化。



1）Controller层

每个业务功能的主操作场景采用一个核心控制器类，利用SDK对移动终端设备的先进场景切换设计，由核心控制器控制不同业务处理视图的切换。

View层通过事件、或者异步线程模式调用控制器的函数，达到把客户响应汇集到控制器。

Model层通过Delegate托管，回调Controller中实现的协议方法把数据汇集到控制器。

2）View层

View负责数据显示与数据输入。数据的输入通过事件交给Model层处理，数据显示通过Model层的托管实现刷新。

3）Model层

Model层服务数据处理，负责向服务器请求数据，并把数据处理好传递给托管。交给View层完成。

#### 2、Model，Controller，View相互通讯的规则

Controller可以直接和Model通信

Controller也可以直接和View通信

Model和View永远不能直接通信

View和Controller的通信是透明和固定的，主要通过outlet和action实现

View使用Delegate接口和Controller同步信息

View不直接和数据通信，使用dataSource接口从Controller处获取数据

View的delegate和dataSource一般就是Controller

Controller负责为View翻译和格式化Model的数据

Model使用Notification & KVO的方式分发数据更新信息，Controller可以有选择的监听自己感兴趣的信息。

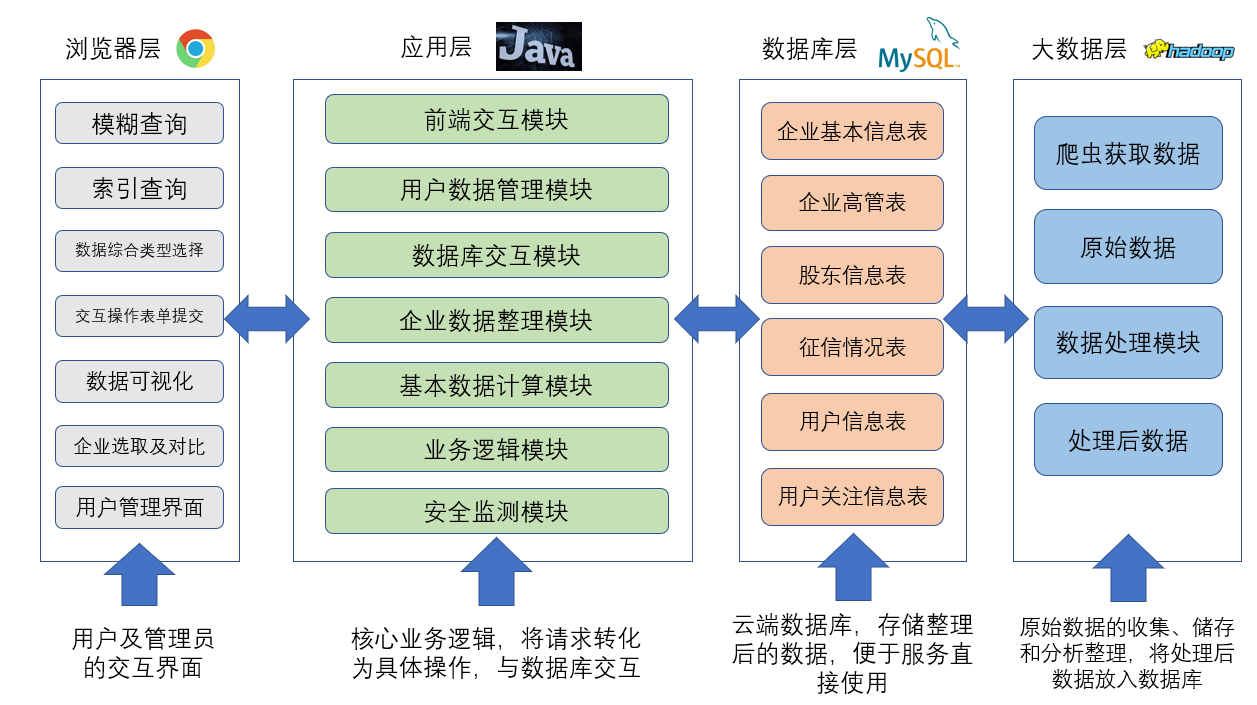
View也可以监听广播信息，但一般不是Model发出的信息

#### 3、Web Service技术

所有数据请求采用Web Service技术，在后台应用设计一个WebService服务提供，通信的数据格式采用JSON。

### 二、核心控制流程

#### 核心控制流程图

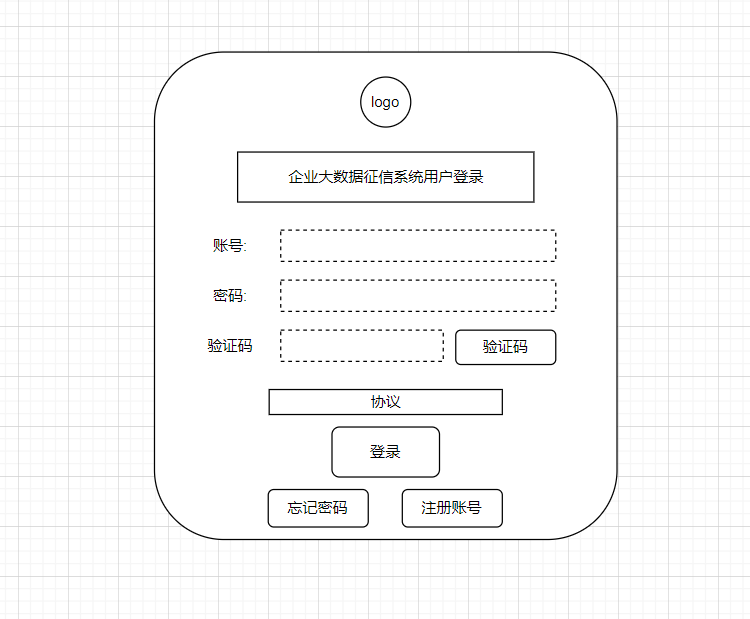


# 第四部分 界面和业务单设计

### 一、界面设计

#### 1、PC界面设计

##### 1.1、登录界面设计



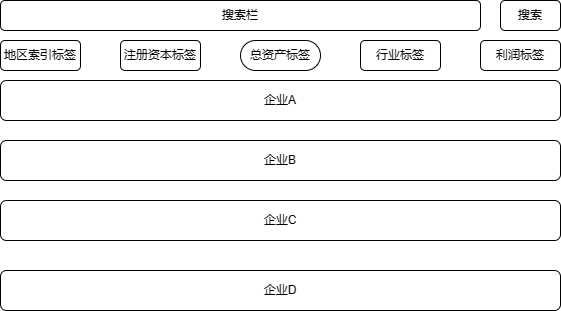


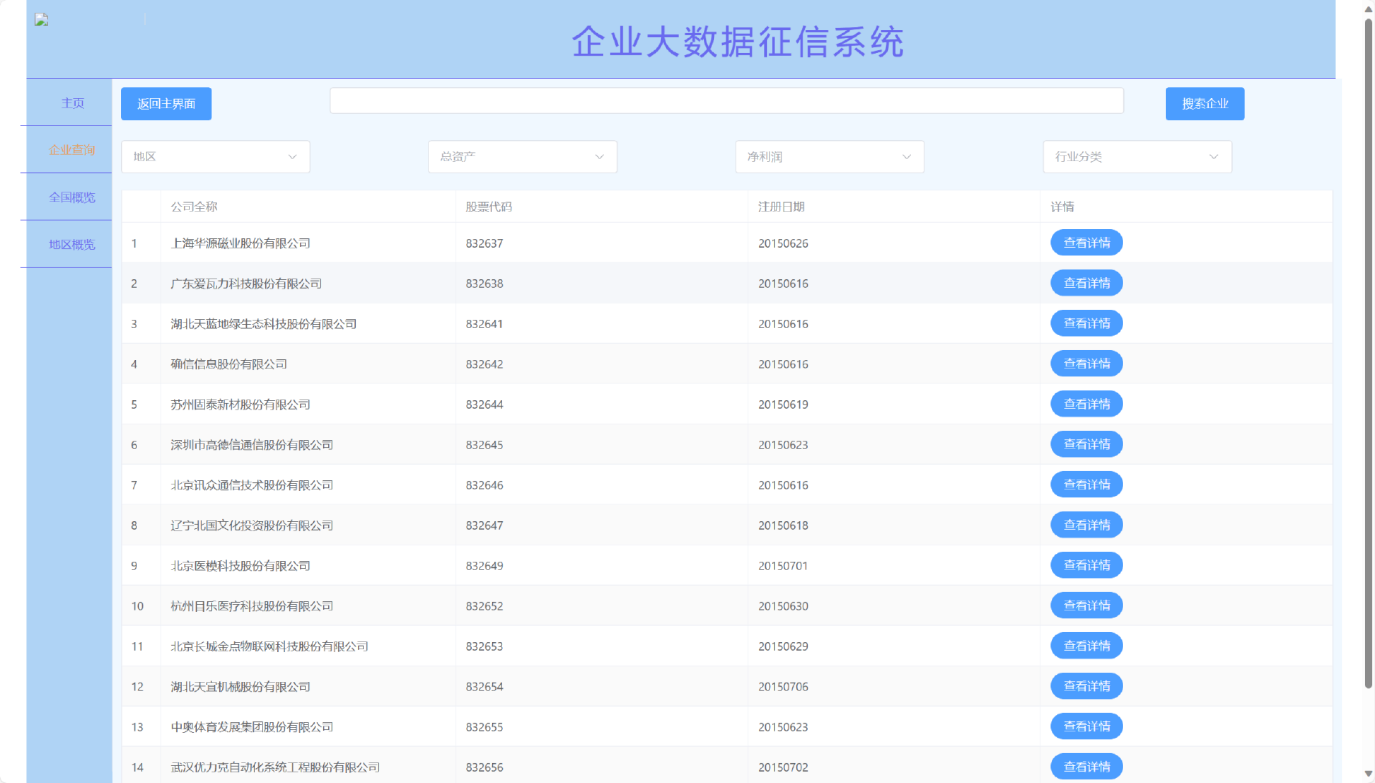
##### 1.2、主体界面设计





##### 1.3、企业查询结果界面设计

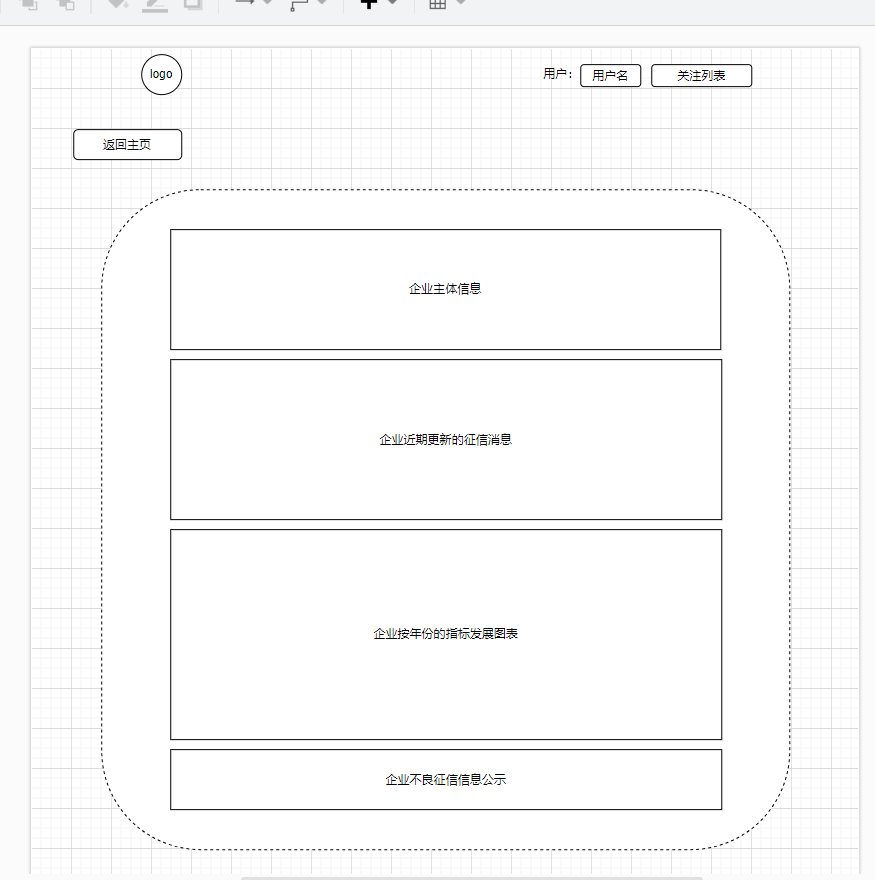


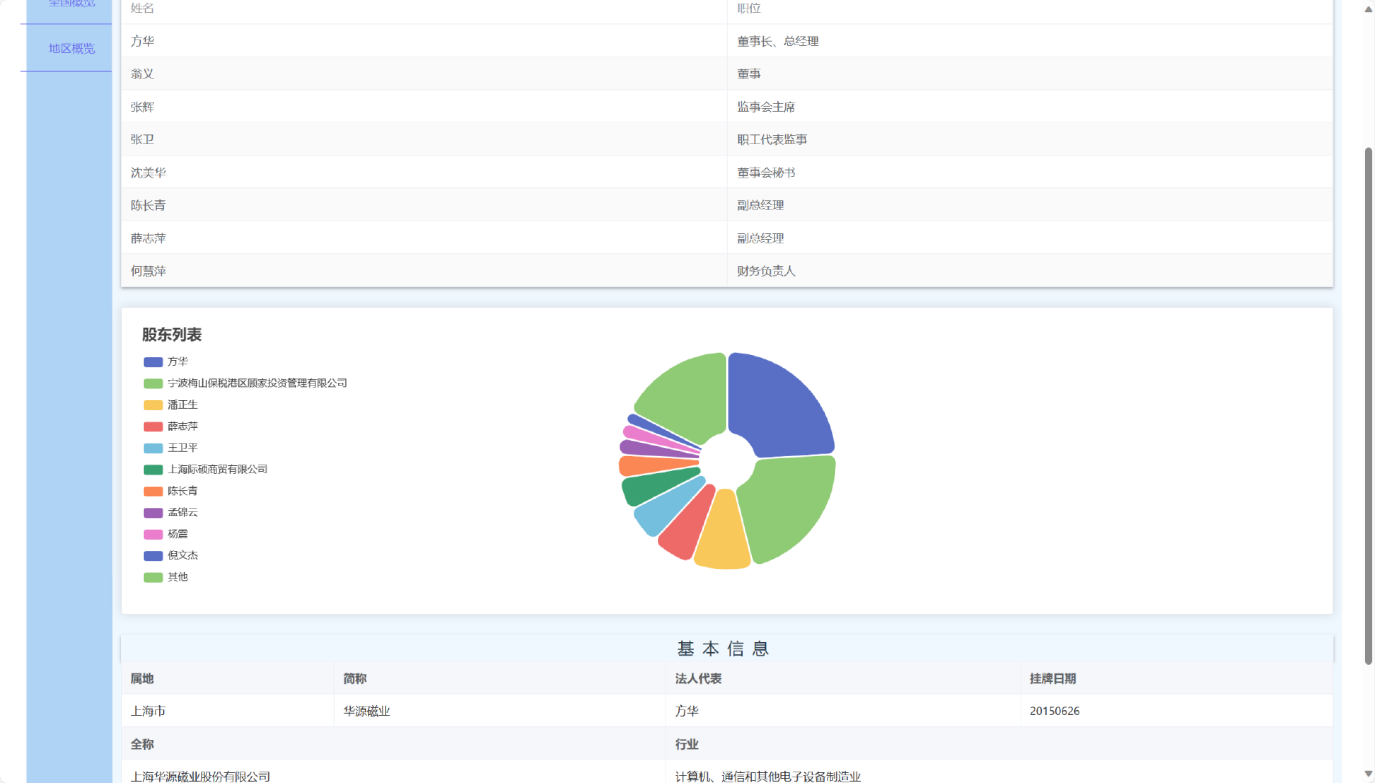


##### 1.4、关注企业界面设计

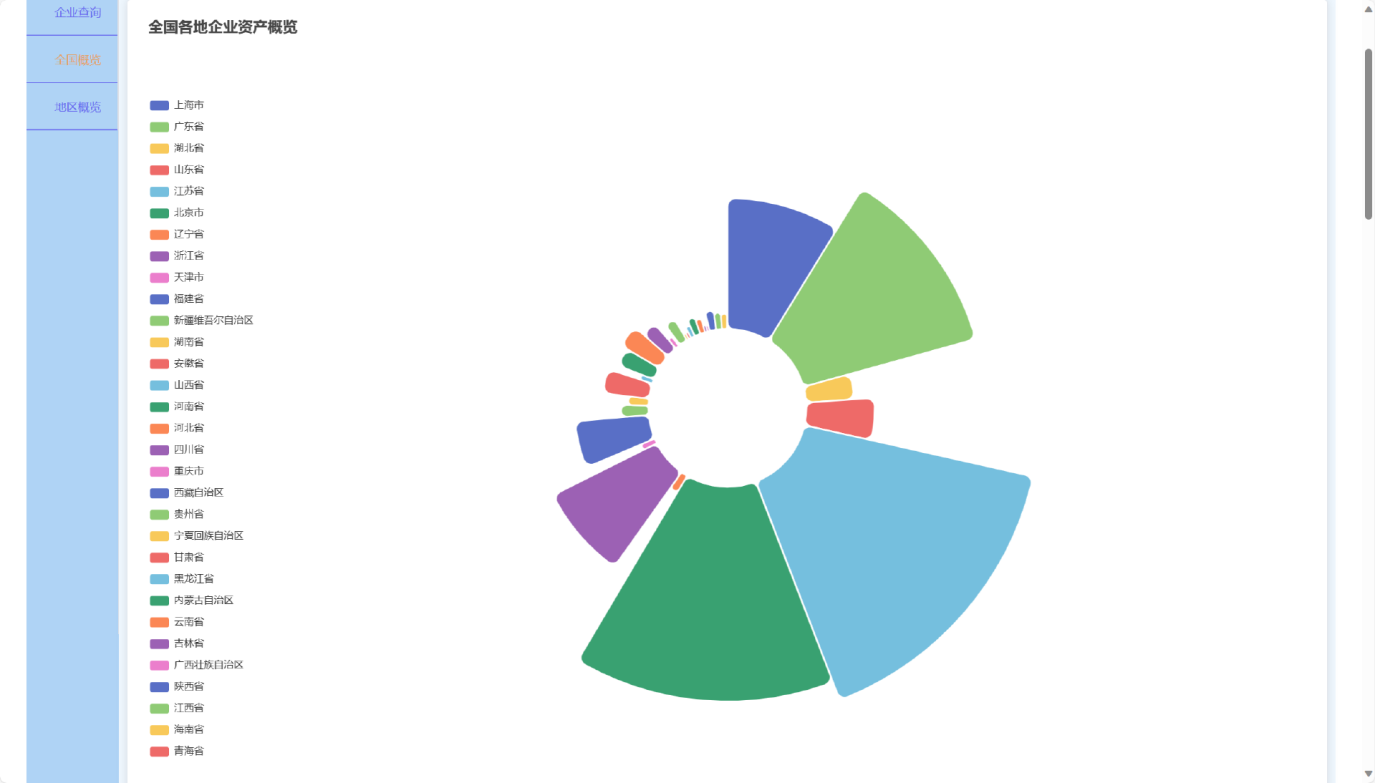
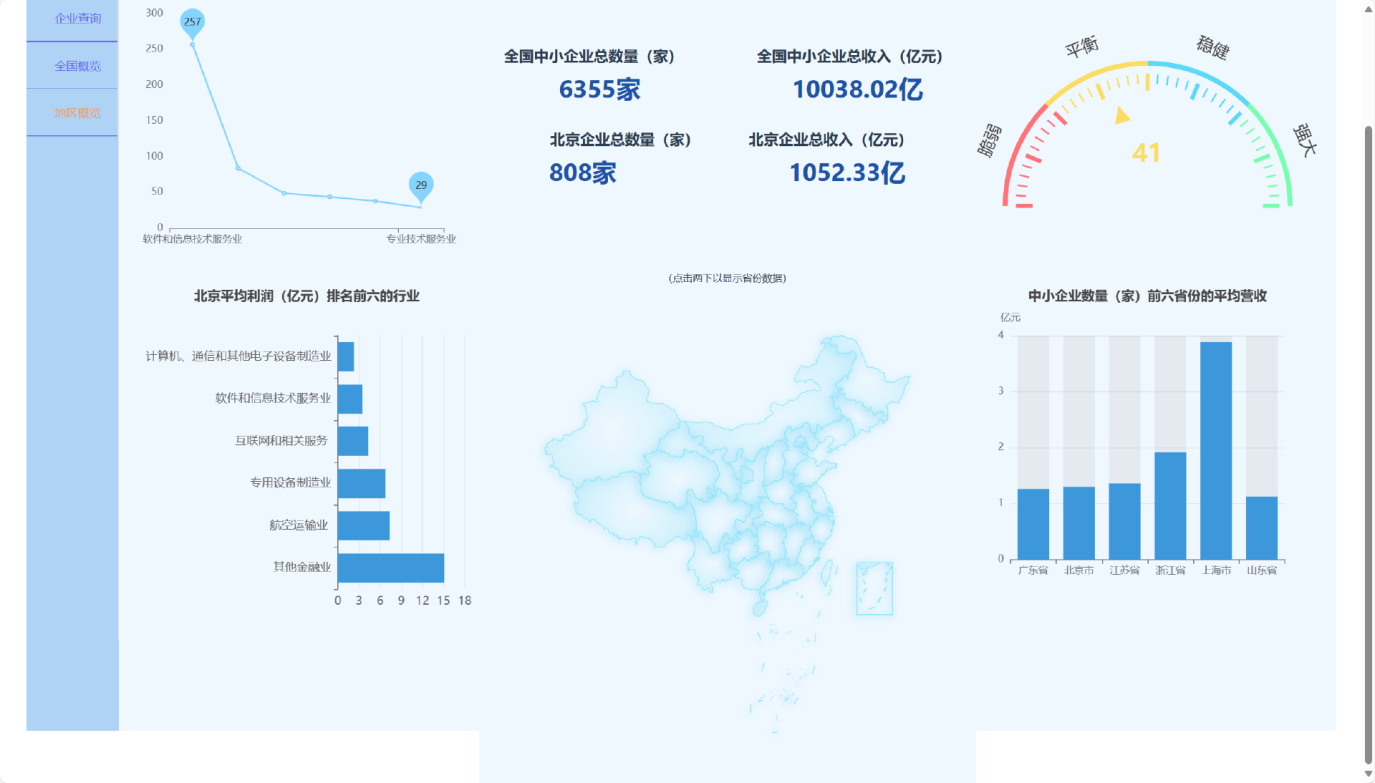
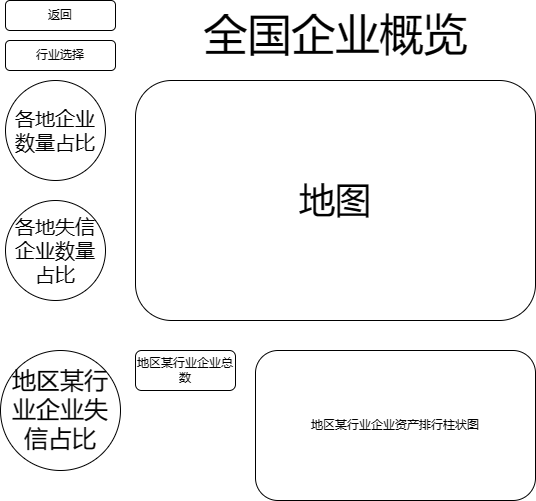


##### 1.5、企业详情界面设计





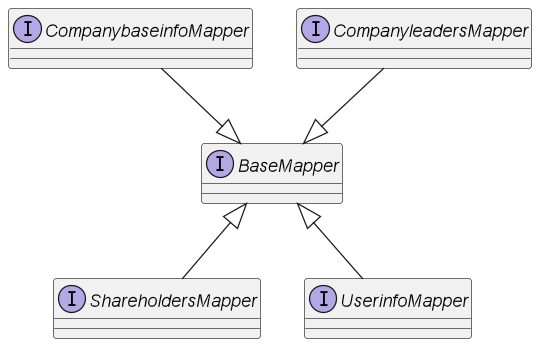
##### 1.6、企业概览界面



# 第五部分 单元模块设计

### 一、数据访问层设计

#### 1、类图设计



#### 2、类的详细设计描述

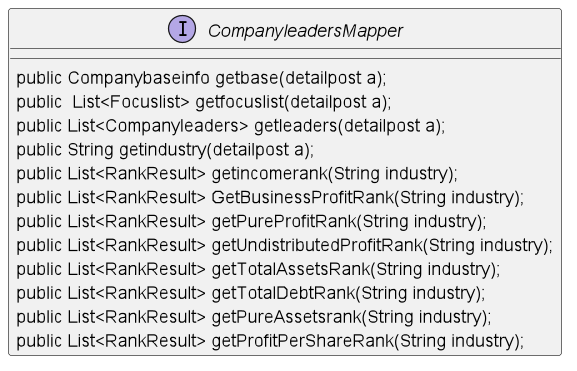
##### 2.1 CompanybaseinfoMapper接口设计



详细描述：

|  |
| --- |
| CompanybaseinfoMapper: 企业基本信息数据库CRUD操作 |
| List<String> getAreaList();  获取所有地区名称 |
| List<String> getIndustryList();  获取所有行业名称 |
| int industry\_companyNumber(String industry);  该行业上市公司总数 |
| List<areaData> industry\_areaCompany(String industry);  该行业内各地区上市公司数量 |
| double industry\_totalAssets(String industry);  行业上市企业总资产 |
| List<areaData> industry\_areaAssets(String industry);  本行业内不同地区的上市企业资产之和(以地区为单位求和，下同) |
| double industry\_totalBusinessIncome(String industry);  本行业上市企业总营业收入 |
| List<areaData> industry\_areaBusinessIncome(String industry);  行业内不同地区的上市企业营业收入之和 |
| double industry\_totalDebt(String industry);  行业上市企业负债之和 |
| List<areaData> industry\_areaDebt(String industry);  行业内不同地区的上市企业负债之和 |
| List<areaData> industry\_areaPureProfit(String industry);  行业内不同地区的上市公司净利润之和 |
| int area\_companyNumber(String area);  该地区上市企业总数 |
| List<industryData> area\_industryCompany(String area);  该地区各行业上市公司数量 |
| double area\_totalAssets(String area);  地区上市企业总资产 |
| List<industryData> area\_industryAssets(String area);  地区不同行业的上市企业资产之和(以行业为单位求和，下同) |
| double area\_totalBusinessIncome(String area);  地区上市企业总营业收入 |
| List<industryData> area\_industryBusinessIncome(String area);  地区不同行业的上市企业营业收入之和 |
| double area\_totalDebt(String area);  地区上市企业负债之和 |
| List<industryData> area\_industryDebt(String area);  地区不同行业的上市企业负债之和 |
| List<industryData> area\_industryPureProfit(String area);  地区不同行业的上市公司净利润之和 |
| List<areaData> selectAreaCompany();  北交所各地区上市企业数量 |
| int selectTotalCompany();  北交所上市企业总数 |
| List<industryData> selectIndustryCompany();  北交所各行业上市企业数量 |
| List<areaData> selectAreaBad();  各地区近2月失信企业数量 |
| int selectTotalBad();  近2月失信企业总数 |
| List<areaData> selectAreaAssets();  各地区上市企业资产总量 |
| List<industryData> selectIndustryAssets();  各行业上市企业资产总量 |
| double selectTotalAssets();  全国所有上市公司资产总量 |
| List<areaData> selectAreaPureAssets();  各地区上市企业净资产总量 |
| List<industryData> selectIndustryPureAssets();  各行业上市企业净资产总量 |
| double selectTotalPureAssets();  全国所有上市公司净资产总量 |
| List<areaData> selectAreaPureProfit();  各地区上市企业净利润总量 |
| List<industryData> selectIndustryPureProfit();  各行业上市企业净利润总量 |
| double selectTotalPureProfit();  全国所有上市公司净利润总量 |
| List<areaData> selectAreaDebt();  各地区上市企业负债总量 |
| List<industryData> selectIndustryDebt();  各行业上市企业负债总量 |
| double selectTotalDebt();  全国所有上市公司负债总量 |

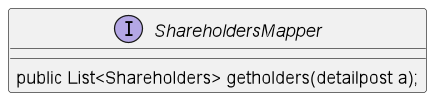
##### 2.2 CompanyleadersMapper接口设计



详细描述：

|  |
| --- |
| CompanyleadersMapper: 企业高管信息数据库操作 |
| public Companybaseinfo getbase(detailpost a);  查询公司基本信息 |
| public List<Companyleaders> getleaders(detailpost a);  查询公司高管 |
| public String getindustry(detailpost a);  得到该企业所属行业 |
| public List<RankResult> getincomerank(String industry);  根据营业收入得到企业排名 |
| public List<RankResult> GetBusinessProfitRank(String industry);  根据营业利润得到企业排名 |
| public List<RankResult> getPureProfitRank(String industry);  根据净利润得到企业排名 |
| public List<RankResult> getUndistributedProfitRank(String industry);  根据未分配利润得到企业排名 |
| public List<RankResult> getTotalAssetsRank(String industry);  根据总资产得到企业排名 |
| public List<RankResult> getTotalDebtRank(String industry);  根据总负债得到企业排名 |
| public List<RankResult> getPureAssetsrank(String industry);  根据净资产得到企业排名 |
| public List<RankResult> getProfitPerShareRank(String industry);  根据每股收益得到企业排名 |

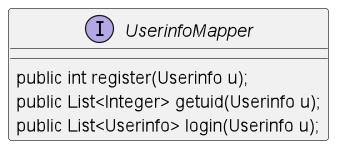
##### 2.3 ShareholdersMapper接口设计



详细描述：

|  |
| --- |
| ShareholdersMapper: 企业股东信息数据库操作 |
| public List<Shareholders> getholders(detailpost a);  获取股东信息列表 |

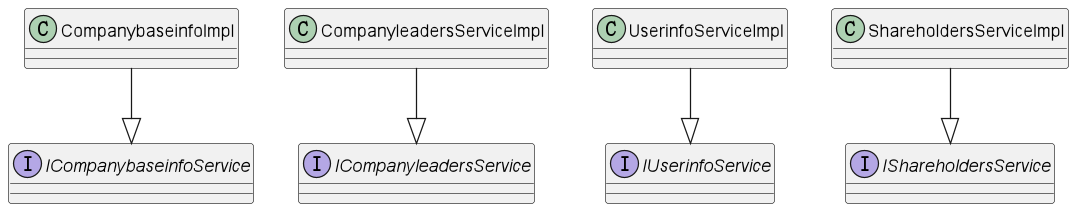
##### 2.4 UserInfoMapper接口设计



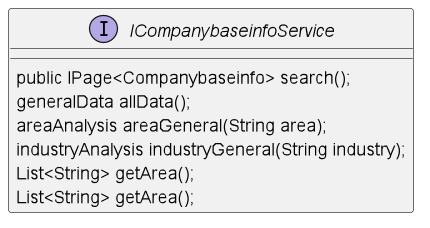
|  |
| --- |
| UserInfoMapper: 用户登录注册数据库操作 |
| public int register(Userinfo u);  根据用户名和密码将数据插入数据库 |
| public List<Integer> getuid(Userinfo u);  根据用户名和密码查询用户UID |
| public List<Userinfo> login(Userinfo u);  根据用户名和密码在数据库中进行查询 |

### 二、业务逻辑层设计

#### 1.类的详细设计描述



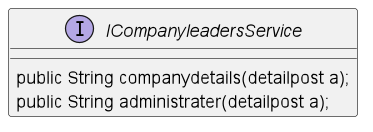
##### 1.1. ICompanybaseinfoService接口设计



详细描述：

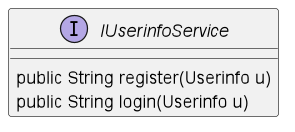
|  |
| --- |
| ICompanybaseinfoService企业基本信息类 |
| public IPage<Companybaseinfo> search(companysearch cbi, int page, int pagesize);  根据关键词模糊查询、索引查询等方式搜索企业及其基本信息 |
| generalData allData();  显示全国企业数据概览 |
| areaAnalysis areaGeneral(String area);  查询指定地区的企业数据概览 |
| industryAnalysis industryGeneral(String industry);  查询指定行业的企业数据概览 |
| List<String> getArea();  显示全部地区列表 |
| List<String> getIndustry();  显示全部行业列表 |

##### 1.2. ICompanyleadersService接口设计



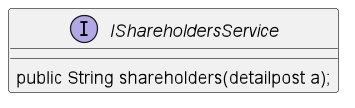
|  |
| --- |
| ICompanyleadersService企业高管类 |
| public String companydetails(detailpost a);  根据全名搜索到指定企业后，显示其详细信息 |
| public String administrater(detailpost a);  根据全名搜索到指定企业后，显示其公司高管信息 |

##### **1.3. IUserinfoService接口设计**



|  |
| --- |
| IUserinfoService: 用户信息类 |
| public String register(Userinfo u)  用户注册 |
| public String login(Userinfo u)  用户登录 |

##### **1.4. IShareholdersService接口设计**



|  |
| --- |
| IShareholdersService持股人类 |
| public String shareholders(detailpost a);  显示指定企业的持股人信息 |

### 三、全国企业概览

#### 1、全国数据分析

查看全国企业的统计信息，以了解相关的行业信息

|  |  |
| --- | --- |
| 功能 | 描述 |
| 全国各省企业资产概览 | 详细了解各省的企业的资产信息 |
| 全国各省上市企业失信概览 | 查看近两个月内各地的失信企业数量 |
| 全国各行业净利润概览 | 查看近一年内各行业的净利润 |

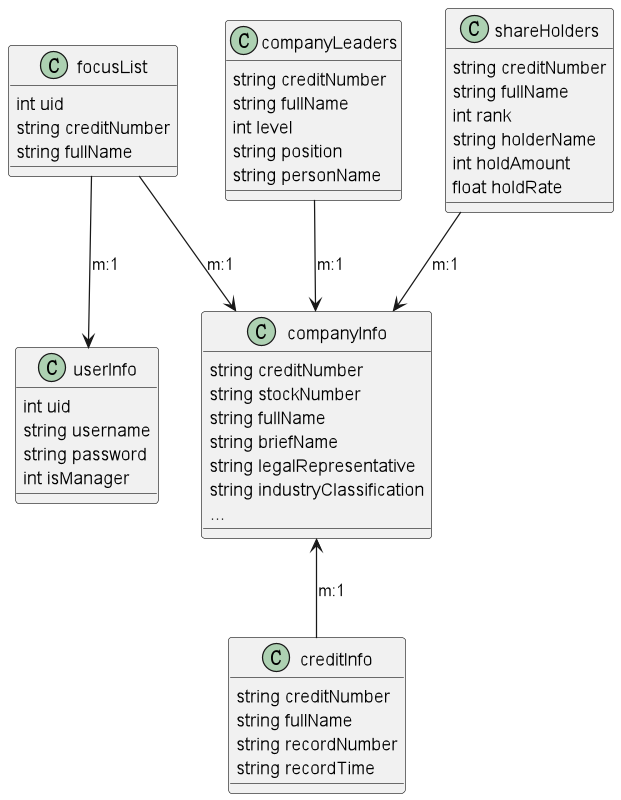
#### 2、地区数据分析

根据地图选取地区，以查看各地区的企业情况

|  |  |
| --- | --- |
| 功能 | 描述 |
| 地区数量排名前六行业 | 折线图显示地区数量排名前六行业 |
| 地区利润排名前六行业 | 柱状图显示地区利润排名前六行业 |
| 地区经济健康状况 | 轮盘图给出地区企业的经济得分 |

# 第六部分 数据库设计

### 一、数据库整体结构图



数据表总览

|  |  |  |
| --- | --- | --- |
| 序号 | 表名 | 注释 |
| 1 | **companyInfo** | 记录所有企业的基本信息 |
| 2 | **companyLeaders** | 记录企业的各位领导人信息 |
| 3 | **shareHolders** | 记录企业的股东信息 |
| 4 | **creditInfo** | 记录被纳入黑名单的企业信息 |
| 5 | **userInfo** | 记录注册的用户信息 |
| 6 | **focusList** | 记录用户的关注列表信息 |
| 7 | **updateTime** | 记录数据库的最近更新时间 |

### 二、系统管理

1. companyInfo实体描述

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | creditNumber | 信用代码 | String |
| 2 | stockNumber | 股票代码 | String |
| 3 | fullName | 公司全称 | String |
| 4 | briefName | 公司简称 | String |
| 5 | legalRepresentative | 法人代表 | String |
| 6 | industryClassification | 行业分类 | String |
| 7 | listingDate | 挂牌日期 | String |
| 8 | area | 地区 | String |
| 9 | registrationAddress | 注册地址 | String |
| 10 | registrationDate | 注册日期 | String |
| 11 | registeredAssets | 注册资本 | String |
| 12 | mainProducts | 主营产品 | String |
| 13 | businessIncome | 营业收入 | Double |
| 14 | businessProfit | 营业利润 | Double |
| 15 | pureProfit | 净利润 | Double |
| 16 | undistributedProfit | 未分配利润 | Double |
| 17 | totalAssets | 总资产 | Double |
| 18 | totalDebt | 总负债 | Double |
| 19 | pureAssets | 净资产 | Double |
| 20 | profitPerShare | 每股收益 | Double |

2. companyLeaders数据表设计

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | stockNumber | 股票代码 | String |
| 2 | fullName | 公司全称 | String |
| 3 | level | 人员级别 | Int |
| 4 | position | 人员职位 | String |
| 5 | personName | 人员姓名 | String |

3.shareholders表结构

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | stockNumber | 股票代码 | String |
| 2 | fullName | 公司全称 | String |
| 3 | rank | 股东排名 | Int |
| 4 | holderName | 股东名称 | String |
| 5 | holdAmount | 持股数 | Int |
| 6 | holdRate | 持股比例 | Double |

4、creditInfo实体描述

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | creditNumber | 信用代码 | String |
| 2 | fullName | 公司全称 | String |
| 3 | recordNumber | 执行案号 | String |
| 4 | attention | 关注人数 | int |
| 5 | recordTime | 立案时间 | String |

5、userInfo实体描述

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | uid | 用户编号 | Int |
| 2 | username | 用户名称 | String |
| 3 | password | 用户密码 | String |
| 4 | isManager | 是否为管理员账户 | Int |

6、focusList实体描述

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | uid | 用户编号 | Int |
| 2 | creditNumber | 信用代码 | String |
| 3 | fullName | 公司全称 | String |

7、updateTime实体描述

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | latestUpdateTime | 最近一次更新时间 | String |

# 第七部分 补充设计和说明

### 一、运行环境

#### 1、服务器环境

1.Web服务器：为了托管前端应用和处理HTTP请求，使用Web服务器。

2.应用服务器：部署Spring Boot应用程序，提供后端服务。使用Tomcat容器。

3.数据库服务器：部署Hive以及与之相关的Hadoop集群来存储和处理大数据。确保硬件和配置能 够满足大数据存储和处理的需求。

#### 2、客户机器环境

1.操作系统： 用户的客户机器可以使用不同的操作系统，例如Windows、macOS、Linux。

2.浏览器兼容性： 前端使用VUE，前端界面在各种主流的Web浏览器中能正常显示和运行，包括Chrome、Firefox、Safari等。

#### **3、开发环境要求**

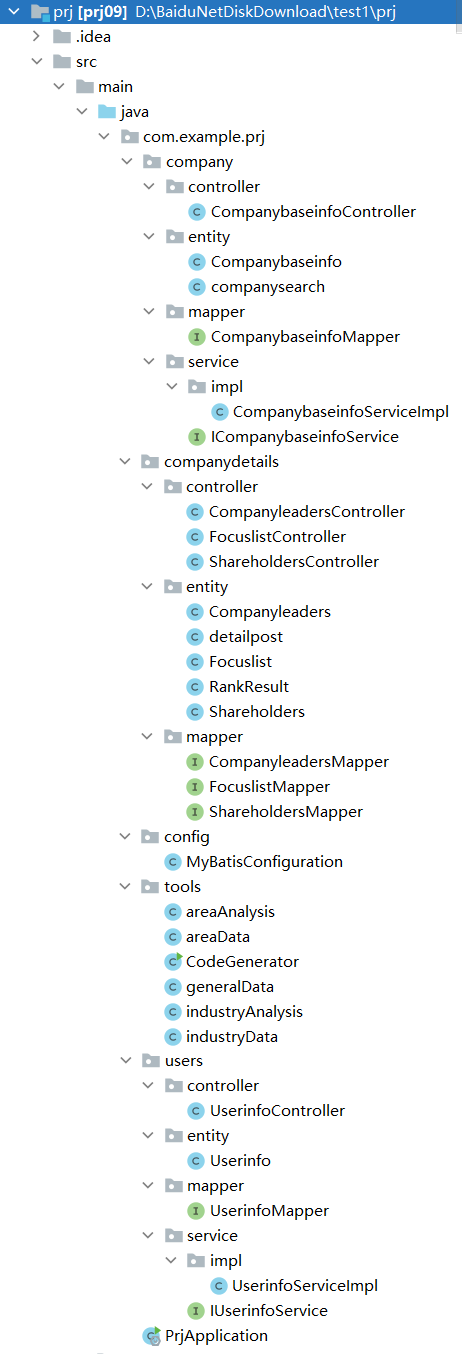
|  |  |  |
| --- | --- | --- |
| 项目 | 名称 | 版本 |
| 开发平台 | Window | 10 |
| 开发工具 | VSCode+IDEA | — |
| 代码管理工具 | gitee | — |
| 开发环境 | JDK | 8 |

### 二、系统性能要求

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| # | 项目 | 模块 | 级别 | 技术参数 |
| 1 | 设计实现技术指标 | 系统架构 | A | 采用C/S模式三层架构。 |
| 2 | 面向对象开发语言与框架 | A | 采用Object-C2.0开发语言和CoreData,Quarz2D等技术。采用Coreplot开源图表绘制框架 |
| 3 | 注释和文档 | A | 符合CMMI软件开发过程标准文档（至少提供：需求、概要、详细设计、测试报告、部署和环境、用户手册），代码注释量>=30%。 |
| 4 | 模块化和适合实训 | A | SOA设计、模块化，保证系统各模块单元较强的独立性适合实训教学。 |
| 5 | 测试覆盖率 | A | 功能覆盖率>=100%，业务覆盖率>=100%，语言覆盖率>=100%，逻辑覆盖率>=80%。 |
| 6 | 资源利用率要求 | CPU占用率 | B | <=50%利用率（附近标准配置） |
| 7 | 内存使用率 | B | <=75%利用率（附近标准配置） |
| 8 | 响应时间要求 | 服务器 | B | <=100ms（附近标准配置） |
| 9 | 网络 | B | <=100ms（附近标准配置） |
| 10 | 客户端 | B | <=5s（附近标准配置） |
| 11 | 系统稳定性要求 | 成熟性 | A | 真实的用户，成功使用本系统 |
| 12 | 稳定性 | B | 无故障运行时间>=365天，系统恢复时间<=2小时。 |
| 13 | 先进性 | A | 采用目前体验最好、最流行的iPhone与iPad移动终端。 |
| 14 | 典型意义 | A | 案例项目要有典型意义，有推广价值。 |
| 15 | Web服务接口要求 | WebService服务器端 | A | 支持协议定制，支持安全过滤，支持消息队列，多种模式重发，支持成功检测。消息延迟<=2S，丢包<=0.001%。 |
| 16 | WebService移动终端 | A | 支持协议定制，支持安全过滤，支持消息队列，多种模式重发，支持成功检测。消息延迟<=2S，丢包<=0.001%。 |
| 17 | MAP平台端 | A | 消息延迟<=2S，丢包<=0.001%。 |
| 18 | 集成部署环境 | 服务器 | A | Microsoft Windows XP/2003 Server |
| 19 | 数据库 | A | Oracle DB 10g 及以上 |
| 20 | 手持机 | A | IPhone 4/iOS 4.0以上 |
| 21 | PC电脑 | A | Microsoft Windows XP 及以上 |
|  |  |  |  |  |
| 说明：级别（A:表示非常重要必须达到的技术性能要求,B:表示重要推荐达到的技术性能要求,C：表示非重要可以弱化的技术性能要求.） | | | | |

### 三、包路径与WEB目录结构设计

#### 1、包路径设计



#### 2、WEB目录结构设计

#### 