都市商圈灵活用工

软件架构文档

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 12/07/2019 | 1.0 | 首次修订完成 | 陈奕君，黄思诚 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 4

3. 逻辑视图 4

3.1 概述 4

3.2 在构架方面具有重要意义的设计包 4

4. 进程视图 4

5. 部署视图 4

6. 实现视图 5

7. 数据视图（可选） 5

8. 核心算法设计（可选） 5

软件架构文档 （简化版）

# 简介

## 目的

本文档将从构架方面对都市商圈灵活用工系统进行综合概述。文档会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

### 本文档旨在为项目的开发提供基础和规范，并提供可重用的资产。架构设计的过程中已将一些公共部分抽象提取出来，形成公共类和工具类，以达到重用的目的。

### 利用软件架构提供的框架或重用组件，缩短项目开发的周期。

### 通过软件架构来提高产品的质量，尽可能满足用户的需求

该文档的读者包括：开发人员、项目管理者和需要阅读本报告的高层管理人员。

## 参考资料

API文档：

* *Auth.md* 鉴权相关接口定义
* *Merchant.md* 商家端接口定义
* *Wechat.md* 微信端接口定义

数据库模型图：

* *CDM.pdf* 数据库概念视图
* *PDM.pdf* 数据库物理视图

迭代计划：

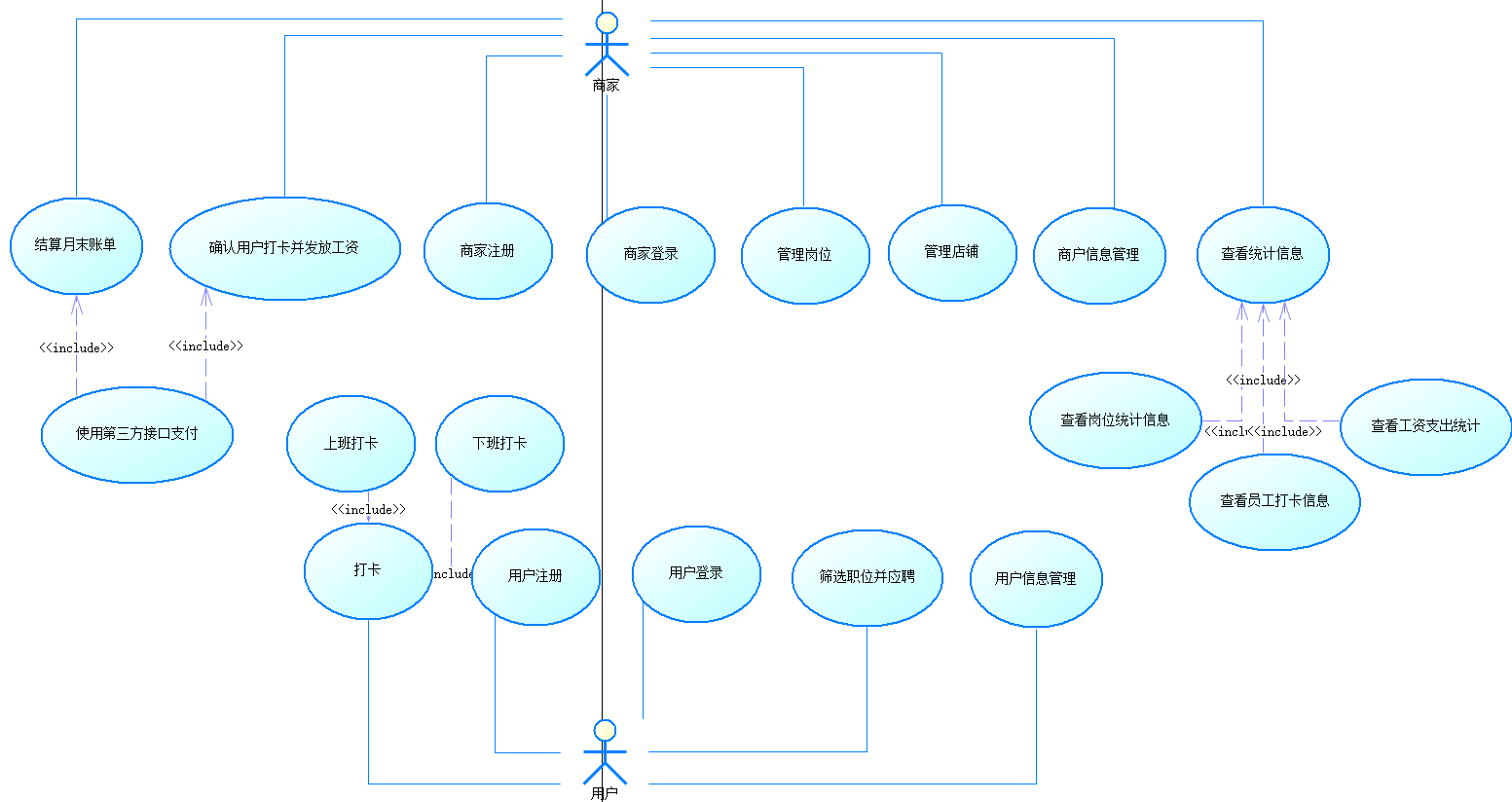
* *plan\_r1.md* 第一次迭代计划

其他资料：

* *proposal.pdf* 项目立项书
* *SRS.pdf* 软件需求规约

# 用例视图

[本节列出用例模型中的一些用例或场景，这些用例或场景应体现最终系统中重要的、核心的功能；或是在构架方面涉及范围很广（使用了许多构架元素）；或强调或阐明了构架的某一具体的细微之处。]



## 商家注册

该用例允许商家提出注册申请，输入合适的用户名与密码，从而成为系统的商户用户。该用例的主要参与者为待注册的商户。

## 商家登录

该用例允许已注册的商家用户登入系统。该用例的主要参与者是已注册的商户。

## 商户管理店铺

该用例允许商家用户管理自己下属的店铺，包括添加删除店铺与修改店铺信息。该用例的主要参与者为已注册的商户。

## 商户信息管理

该用例允许商家用户管理自己的相关信息。改用例的主要参与者为已注册的商户。

## 商户管理岗位

该用例允许商家用户管理下属店铺所拥有的岗位，包括添加岗位、修改岗位信息、删除岗位、管理岗位应聘情况等。

## 使用第三方接口支付

该用例主要描述用户使用第三方接口结算与转账汇款。该用例的主要执行者为商户。

## 结算月末账单

该用例主要描述商家每月与平台结算月末账单。该用例包含商户使用第三方平台支付的子流。该用例的主要执行者为商户。

## 确认用户打卡并发放工资

该用例主要描述用户工作完成后商户确认用户的打卡信息并同意发放工资。该用例包含商户使用第三方平台支付的子流。该用例的主要执行者为商户。

## 商家查看统计信息

该用例主要描述商户查看系统中相关统计信息的过程，包含岗位统计信息、员工打卡信息与工资支出统计。该用例的主要执行者为商户。

## 用户注册

该用例从微信端进入的用户输入合适的个人信息从而成为系统的普通用户。该用例的主要参与者为待注册的微信端用户。

## 用户登录

该用例允许已注册的微信端用户登入系统。该用例的主要参与者是已注册的微信端用户。

## 用户上下班打卡

该用例描述用户上下班打卡的过程。该用例的主要参与者是已应聘的微信端用户。

## 用户筛选岗位并应聘

该用例描述用户从岗位列表筛选合适的岗位，并应聘的功能。该用例的主要参与者是微信端用户。

## 用户信息管理

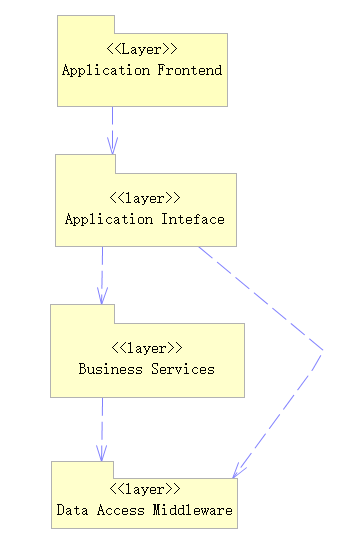
该用例描述用户管理自己的个人信息的过程。该用例的主要参与者是微信端用户。

# 逻辑视图

[本节说明设计模型在构架方面具有重要意义的部分，例如设计模型被分解为多个子系统和包。而每个重要的包又被分解为多个类和类实用程序。您应该介绍那些在构架方面具有重要意义的类，并说明它们的职责，以及几项非常重要的关系、操作和属性。]

## 概述

系统逻辑架构如下图所示。



系统的逻辑视图主要由四层组成，分别是Application Frontend层、Application层、Bussiness Service层与Data Access Middleware层。

Application Frontend层主要实现了系统的人机交互界面，负责接收用户的请求并向下层的Application Interface发送数据请求并呈现结果。

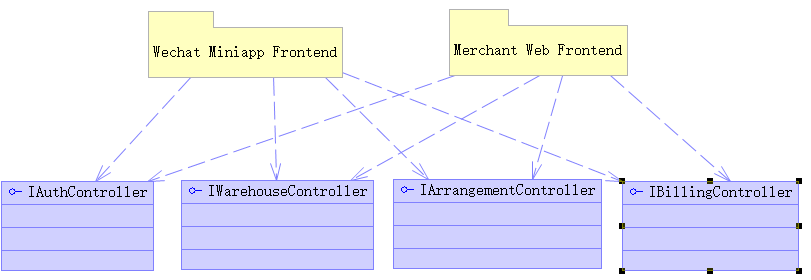
Application层主要由响应各种用户界面请求的控制器类完成，它响应来自前端的REST请求并调用Bussiness Service层中的函数进行业务逻辑处理，并将相应的结果返回给用户。

Bussiness Service层主要实现业务逻辑，包含实际进行业务处理的Service类与访问数据的DAO类、访问数据库的Repository类与Entity实体类。

Data Access Middleware主要包含进行数据库访问的各类中间件，主要包含Spring framework中的函数与类库。

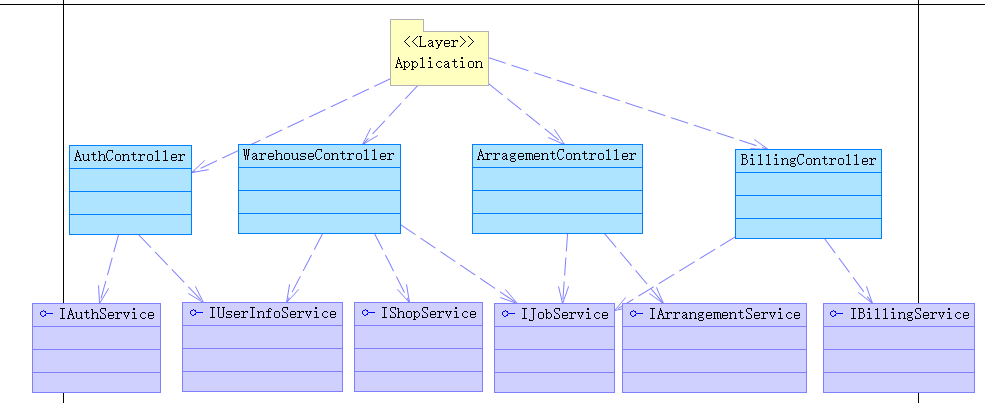
## 在构架方面具有重要意义的设计包

### Application Frontend层



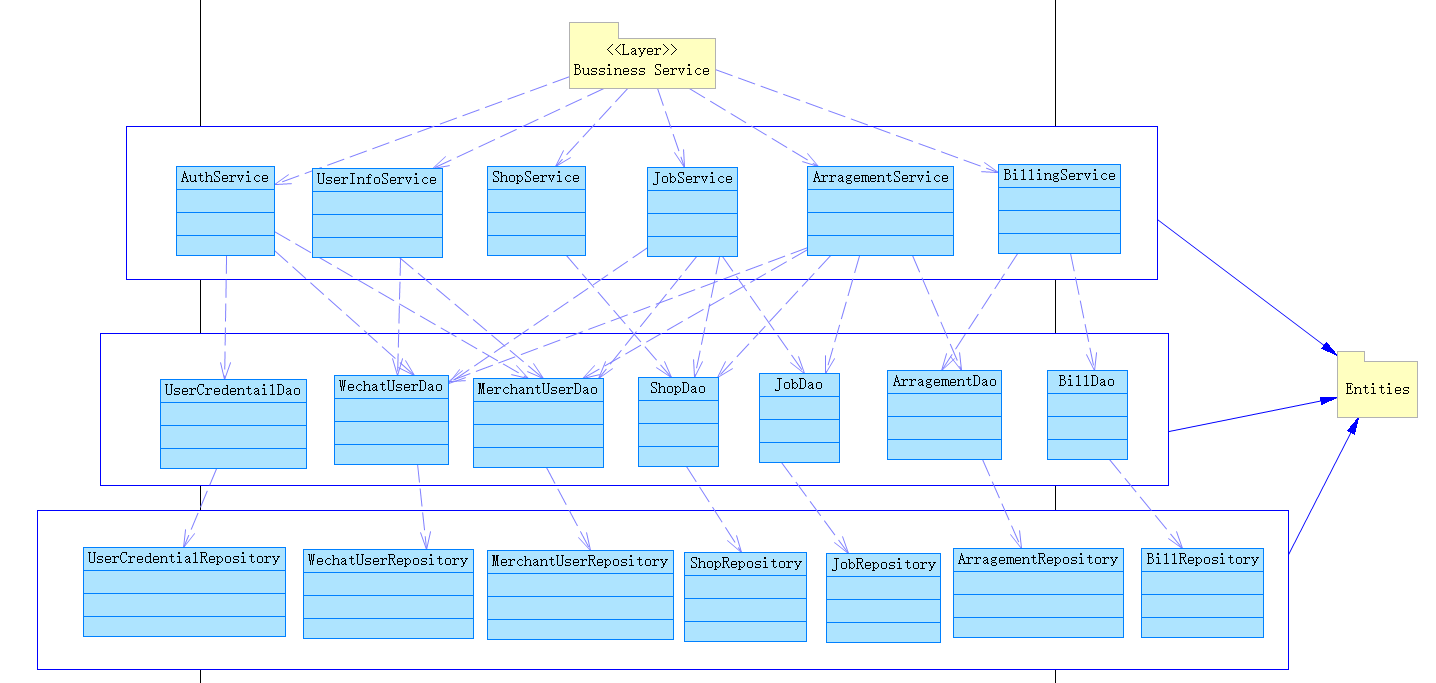
Application FrontEnd层主要由两个子包构成：微信小程序端包库与商家Web端包库。它们通过RESTful接口与Application层进行通信。

### Application层



Application层主要包括4个子包：AuthController实现鉴权接口，WarehouseController实现信息管理接口，ArragementController实现排班相关接口；BillingController实现财务管理相关接口。

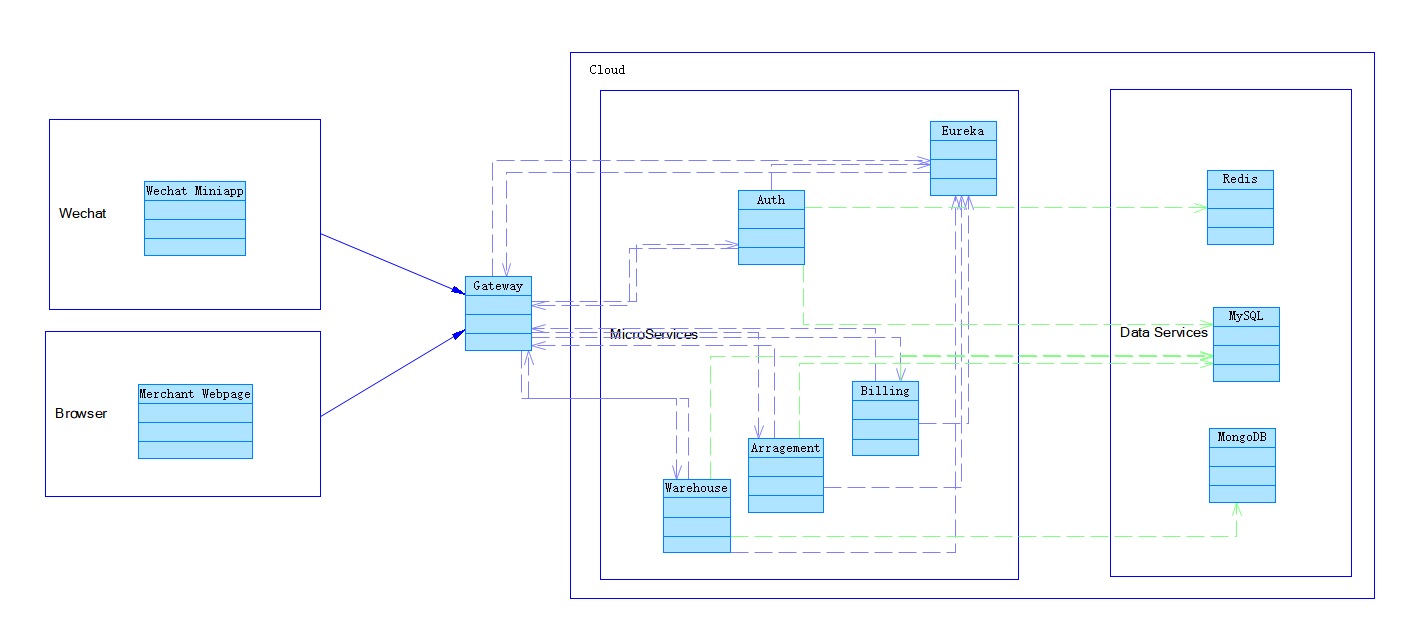
### Bussiness Service层



Bussiness Service层主要实现业务逻辑与数据访问，包括Service层、Dao层与Repository层。Service层实现具体业务逻辑，Dao层进行数据访问控制，Repository层与下层的中间件进行通信，进行OR映射与实际数据访问。所有层级共同依赖实体类entity，详细定义可参考CDM与PDM。

# 进程视图

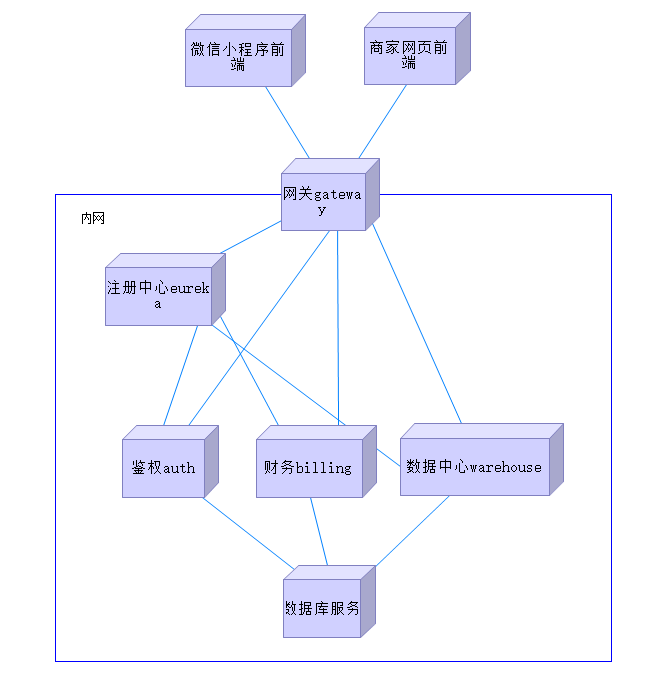
进程视图如下所示：



用户分别通过web端（商家）与微信端（用户）与后端网关交互。系统由数据库和若干微服务副本构成，每个数据库和微服务副本部署在一个Docker容器中，运行一个进程。其中包括：关系型数据库MySQL，非关系型数据库MongoDB，KV数据库Redis；Eureka Server注册中心微服务,Gateway网关微服务，Auth鉴权微服务，Warehouse数据中心微服务，Billing支付微服务与Arrangement排班微服务。

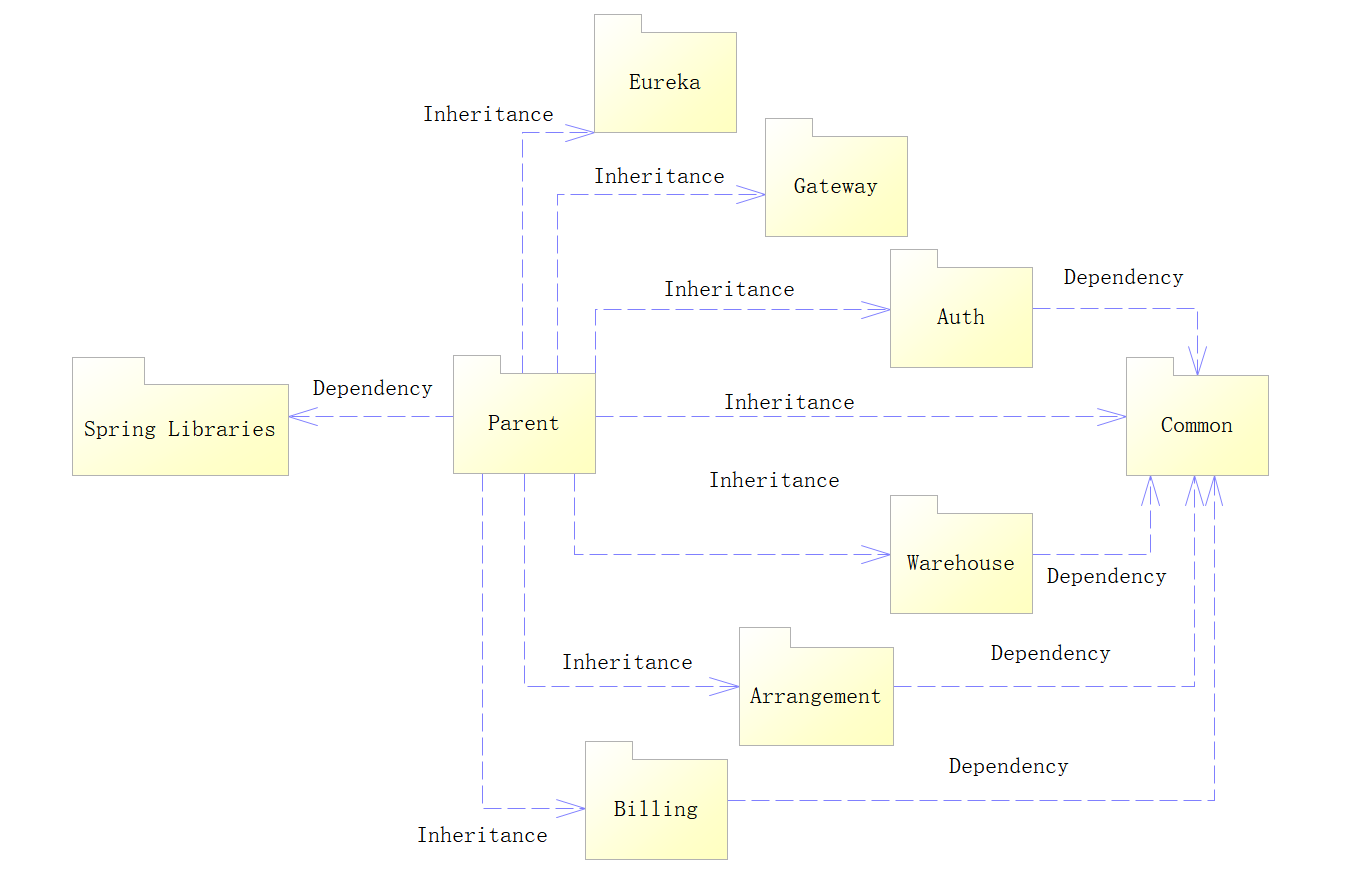
# 部署视图

整个系统分布式部署于多台云服务器上，并位于同一个子网内。服务运行于Docker内直接共享宿主环境的端口。各数据库和服务通过内网IP地址进行互相通信。仅网关带有公网IP地址，并向用户提供服务。各台云服务器需要至少4核CPU、8GB内存，并提供至少100Mbps的内网带宽，网关需要提供1Gbps的公网带宽。

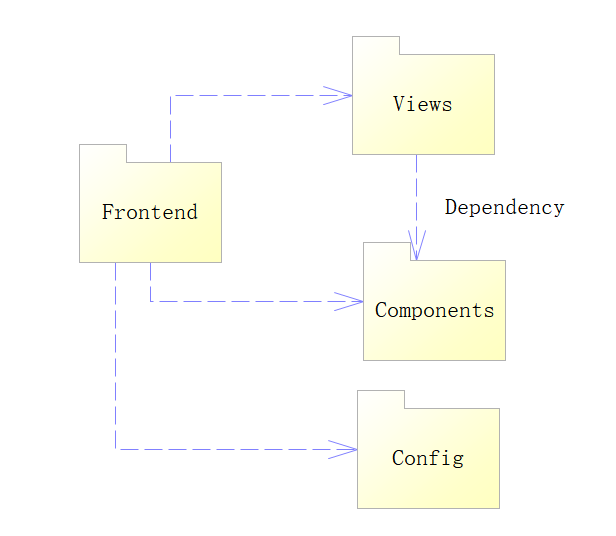
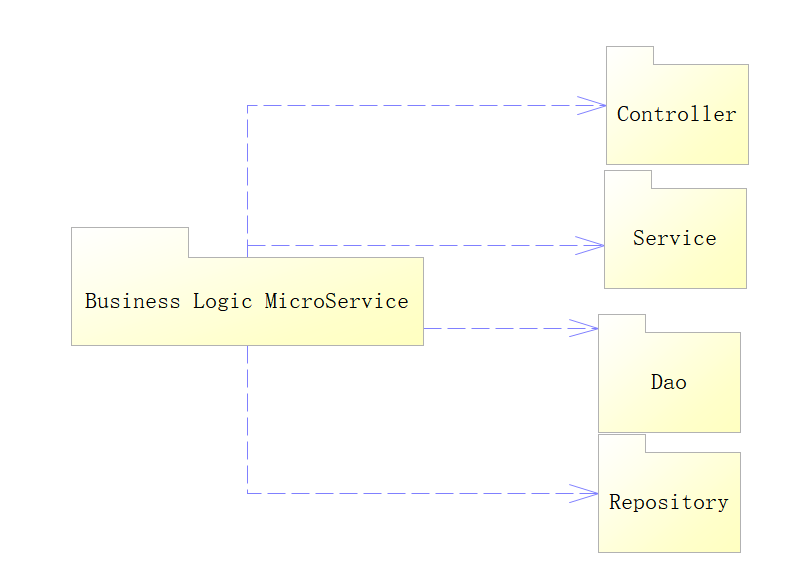


# 实现视图

后端工程代码结构如图所示：



所有微服务子模块均继承自Parent模块。Parent模块负责统一管理各子模块的依赖，保持依赖版本一致。所有的公用Util类（如Controller的ResponseBuilder模块）与所有的实体类均由Common子工程统一管理，保持实体类在不同微服务的一致性。



后端微服务模块的代码结构如左图所示，分为Controller、Service、Dao与Repository层。前端工程主要由View，Components与Config构成，且基于Vue框架的web端与基于小程序框架的微信端结构类似。

# 数据视图（可选）

详细可见对象设计的概念模型*CDM.pdf*与物理模型*PDM.pdf*。

# 核心算法设计（可选）

核心排班算法主要基于两个方面，一是基于时间、地点、薪资的匹配；二是基于用户感兴趣标签的匹配。时间、地点的匹配可直接基于数据排序；标签匹配需要从数据中学习标签的兴趣向量，并对兴趣向量余弦相似度算法计算距离进行排序。