1 概述

1.1 系统简述

本系统是基于HTML, CSS, JavaScript和Python Fastapi框架开发的Web应用程序。其中，前端的浏览器在客户端供用户使用，前端发送的HTTP请求被后端接收后，先由视图层处理再转发至模型层，通过ORM调用数据库中的数据，再反馈给前端。

系统包含以下两个子系统：

* + 用户信息管理
  + 宿舍信息管理

1.2 软件设计目标

1.2.1 使学生更快捷地申请宿舍业务

1.2.2 使宿舍管理员更方便地处理宿舍业务

1.2.3 支持数据的批量处理

1.3 参考资料

1.4修订版本记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **描述** | **作者** |
| <9/6/2021> | <1.0> | 关于宿舍管理系统的初步详细设计 | 小组所有成员 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

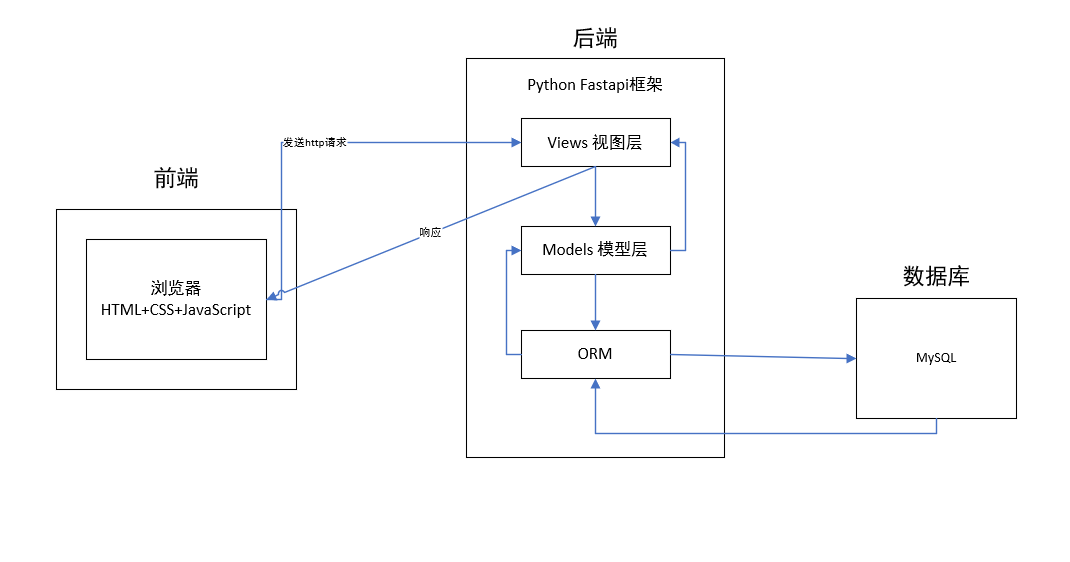
2 术语表

请参考需求分析文档。

3 设计概述

3.1 系统的复用计划：库、框架、模式、构件等方面的复用（如果没有就不写了）

Python Fastapi框架



3.2系统接口设计(根据概要设计进一步细化设计)

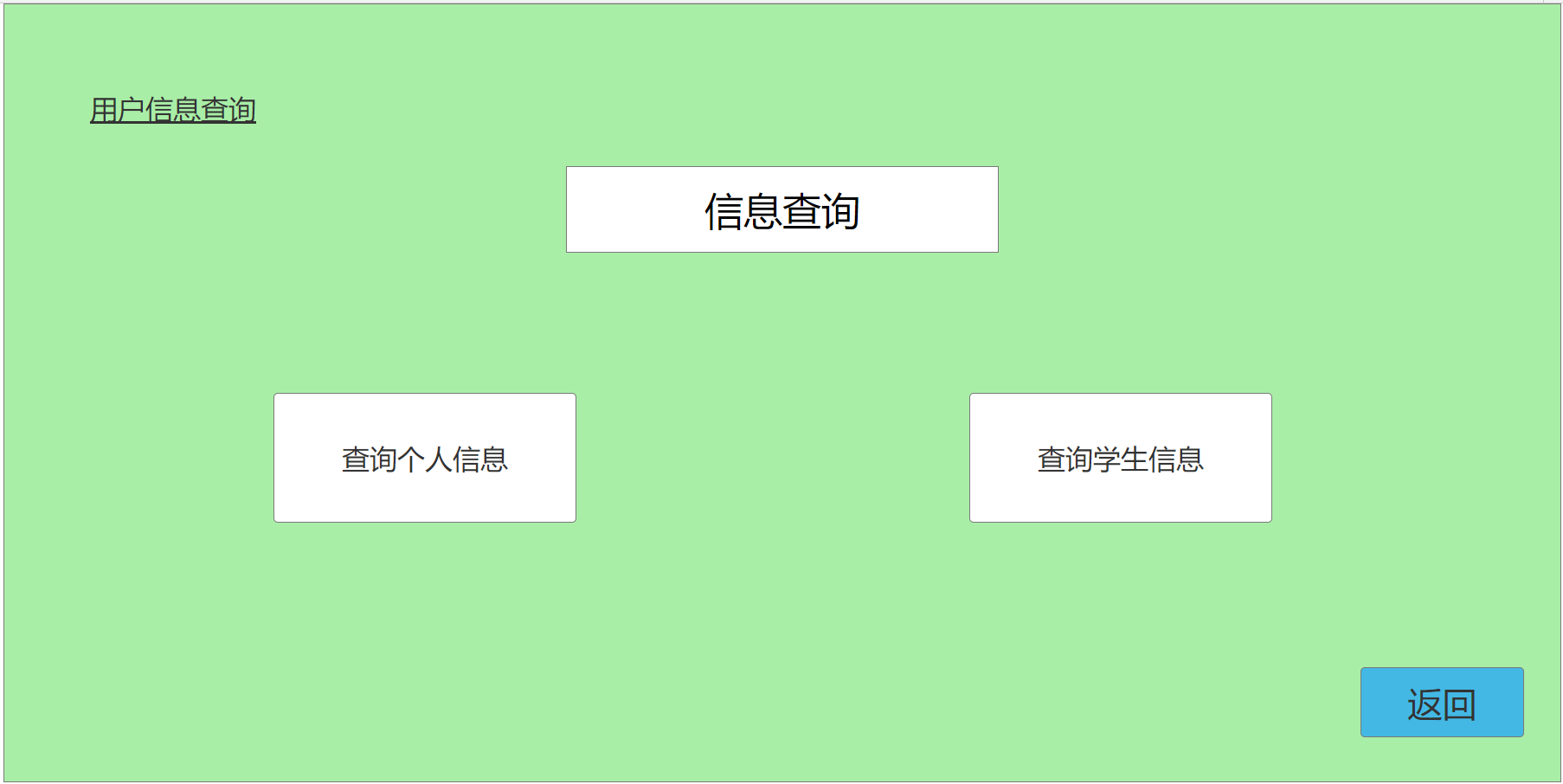
• 各种提供给用户的界面

**1．宿舍管理员界面**

1.1 宿管管理的总界面：普通宿管可以用鼠标点击查询宿舍信息、查询用户信息、处理宿舍业务按钮来进入对应界面；点击返回按钮返回至**登录界面**；总宿管能额外有处理离校申请、处理入住申请按钮。



1.2 查询用户信息界面：宿管点击查询个人信息或查询学生信息来进入对应界面；点击返回按钮返回至**宿管管理总界面**。



1.3 查询个人信息界面：宿管可以点击返回按钮返回至**查询用户信息界面。**



1.4 查询学生信息界面：用户可以用鼠标选择年级、学院、性别、班级或用键盘输入学号、姓名；然后点击查询按钮，系统将返回查询的学生信息在下方。

点击返回按钮可以返回至**查询用户信息界面**。



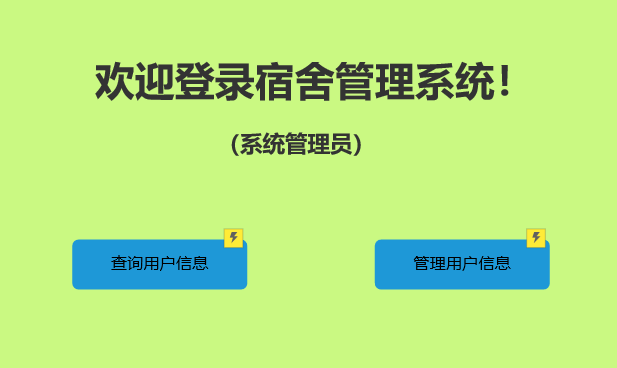
1.5 处理入住申请界面：宿管点击确认分配按钮可以为单个学生分配宿舍，系统将弹出分配成功或者分配失败的信息；点击一键分配可以为所有申请学生分配宿舍；点击返回按钮将返回至**宿舍管理总界面**。



1.6 处理离校申请界面：宿管可以点击拒绝按钮，然后在弹窗中用键盘输入拒绝原因；点击返回按钮可以返回至**宿舍管理总界面**。



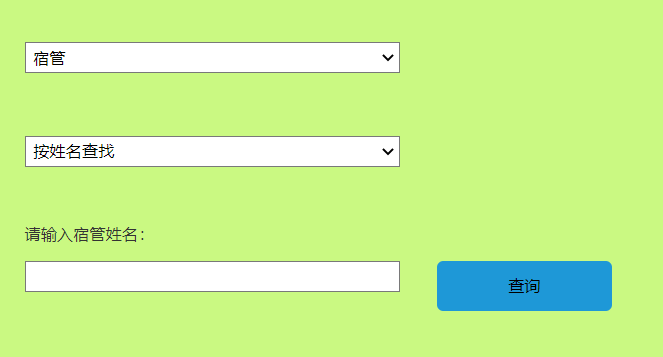
**2.系统管理员界面**



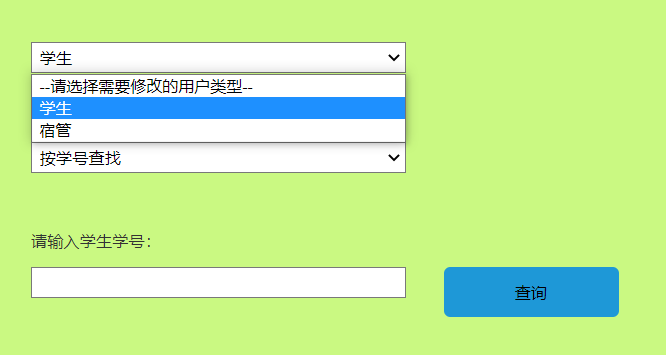
2.1 系统管理员总界面：系统管理员登录后，系统跳转至此界面，可以执行查询用户信息和管理用户信息操作。

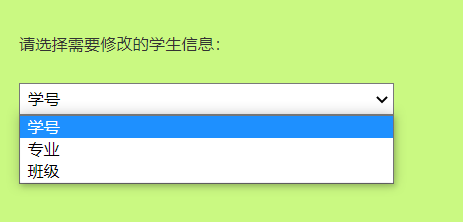
2.2 查询用户信息界面：系统管理员可以选择查询的用户类型，并根据用户类型的具体属性进行查询。





2.3 修改用户信息界面：系统管理员先查找到需要修改的用户，再选择具体属性进行修改。



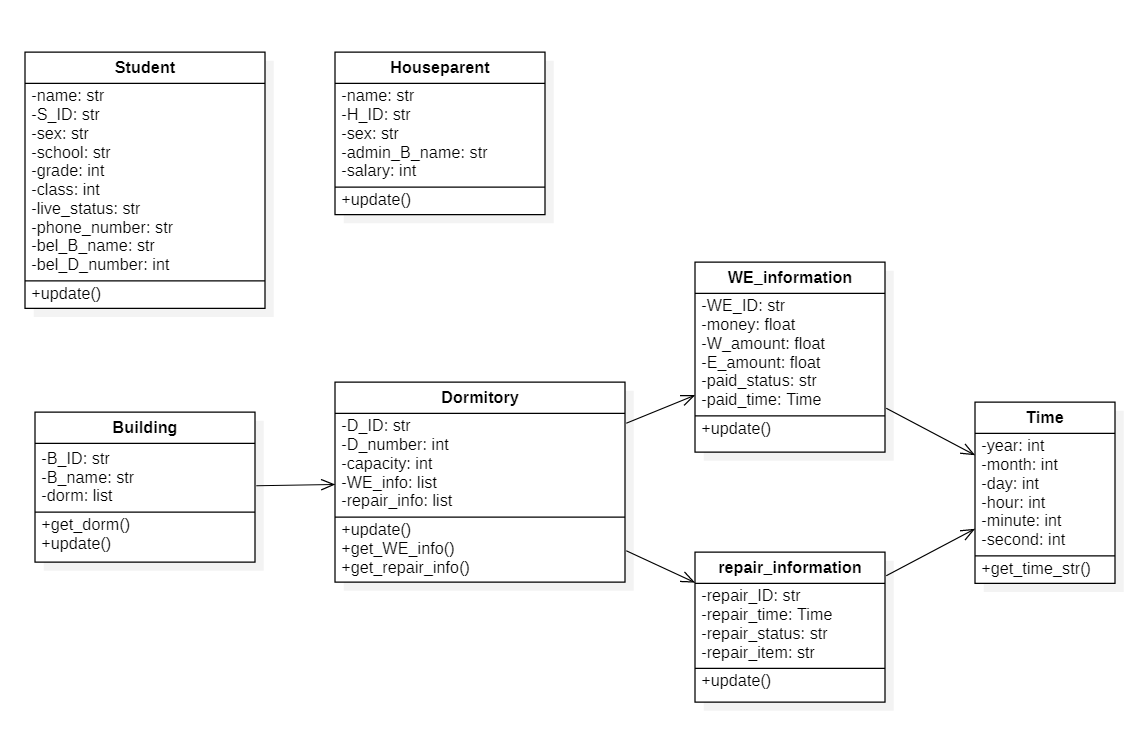


• 系统外部接口设计：与外部系统的交互设计

3.3 对象模型设计

• 对象模型（类图或对象图）：提供整个系统的对象模型，在其中应该包含

所有的系统对象。所有对象之间的关联必须被确定并且必须指明联系的基数。

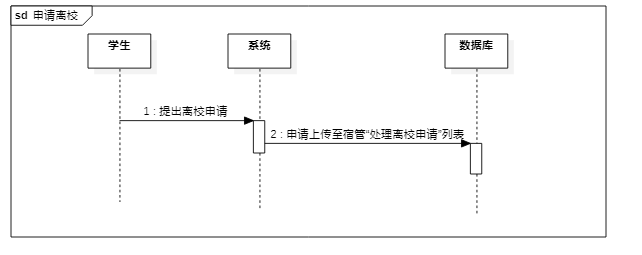


• 对象描述: 在这个部分叙述每个对象的细节，它的属性、它的方法。对每个对象的每个属性详细说明：名字、类型; 对每个对象的每个方法详细说明：方法名，返回类型，返回值，参数，用途以及使用的算法的简要说明。

3.4系统用例实现详细设计

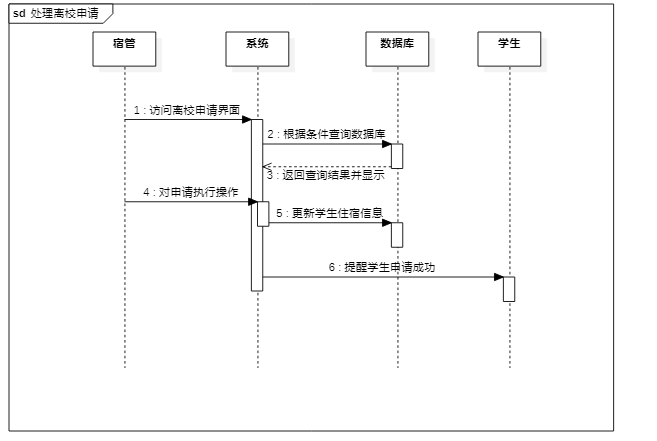
3.4.1 申请离校：

– 顺序图：



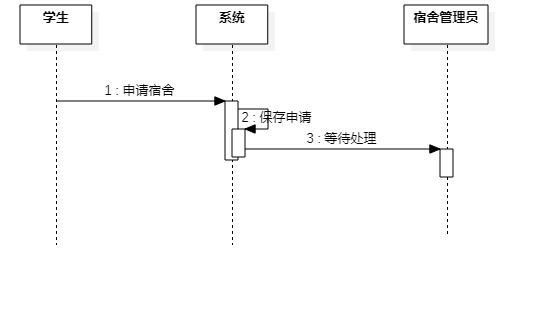
3.4.2 处理离校申请：

– 顺序图：



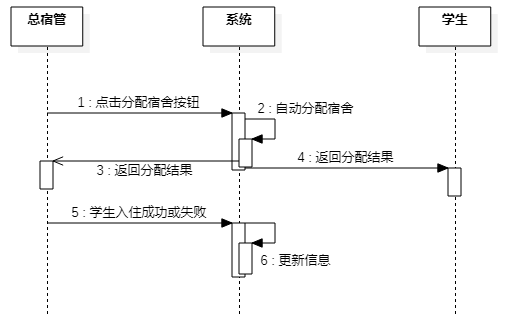
3.4.3 申请入住

– 顺序图：



3.4.4处理入住申请

– 顺序图：



3.5系统非功能设计

3.5.1满足系统性能需求：

3.4.1.1使用连接池。连接池技术能够重用到数据库的连接，而不是每次请求都建立新的TCP连接，新连接仅在于连接池中得不到连接时才建立。当连接被关闭时，它被返回到连接池中。

3.4.1.2使用缓存技术。

3.5.2满足系统可靠性需求：

3.4.2.1在软件需求分析阶段，制定详细的需求分析计划和需求分析方法，保证需求分析彻底、完整、准确、可靠。

3.4.2.2设计过程中保证需求分析和说明定义的一致性、完整性、准确性和无歧义性，进而保证开发过程中对需求容易辨识。

3.4.2.3进行详细的数据组织规划和逻辑设计，保证软件设计过程中的数据组织的准确性和逻辑的可靠性，进而防止软件开发设计过程中出现过多的逻辑错误和数据描述错误。

3.4.2.4选择可靠有效的建模语言，系统设计整个过程中保证先建模后验证再编码，有效保证每个编码的质量。

3.5.3满足系统易用性需求：

3.4.3.1在系统开发过程中遵循迭代优化式开发模式，通过详细设计与代码开发是作用与反作用的关系来提高系统易用性。

3.4.3.2采取界面内容可见即可用的设计原则，在详细设计时，将系统各项操作与权限角色进行关联，无权操作内容不应出现在用户界面上，且不应出现用户需根据个人角色信息选择不同按键的操作（用户有双重角色除外）。