软件技术架构：

cors

springboot

Hibernate

cors

Echarts

React

Material UI

&

Bootstrap

Sqlalchemy

Flask

Sqlalchemy

sklearn

Frontend

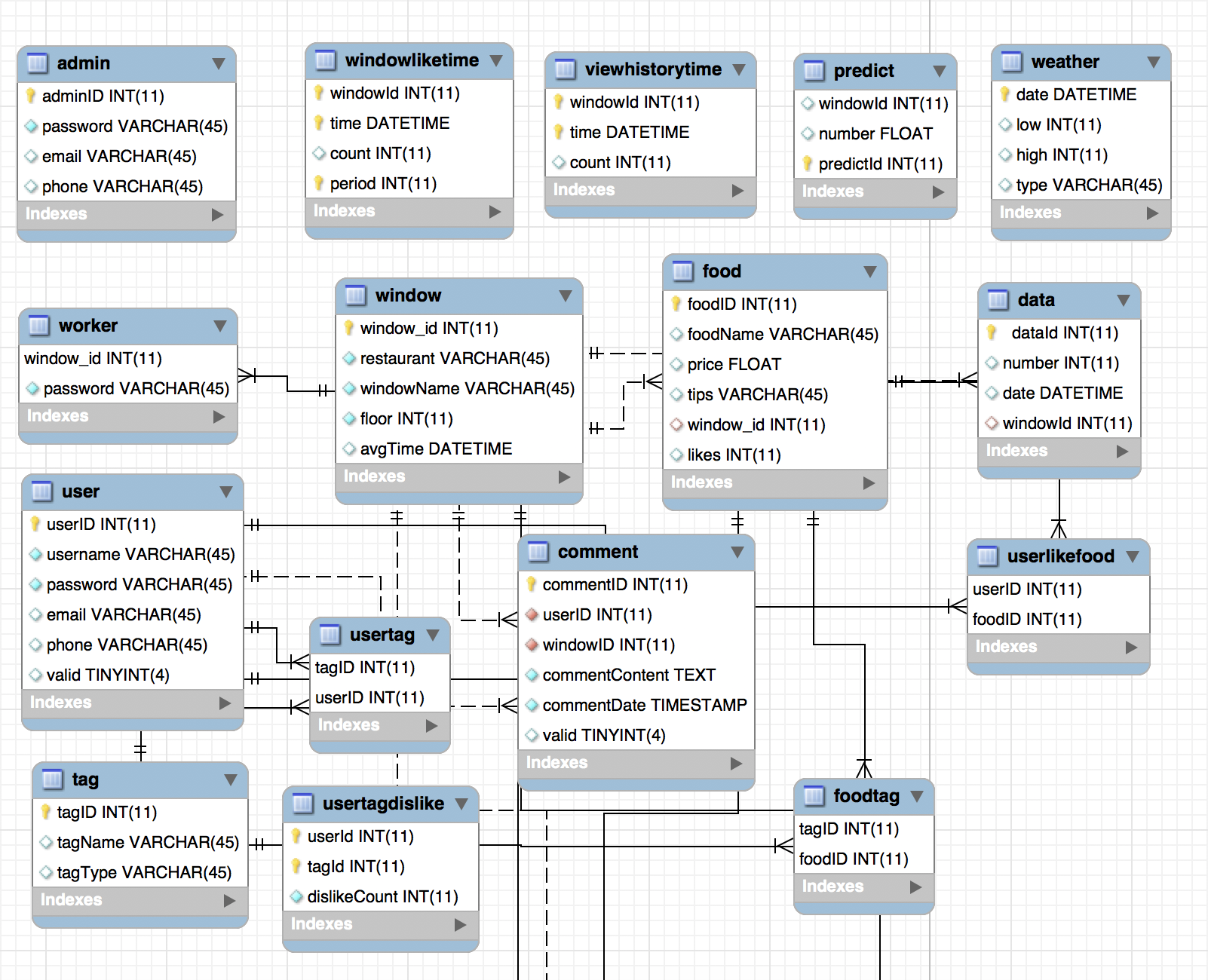
timed task

Backend

MySQL

软件架构

数据库架构：



排队人数模拟、统计、预测与实时可视化设计详情：

* 排队人数模拟：

采用正态分布加随机数并控制上下限的方式生成实时数据。数据间隔为2分钟一个，在软件实际运行时获取数据时即2分钟统计一次餐厅刷卡计数器的计数。

* app.py 用于实时生成模拟数据。
* 排队人数统计：

目前假设早餐时间为7-10点，午餐时间为11-14点，饭餐时间为17-20点， 以2分钟为间隔计数一次，即每个窗口一天会有3 \* 90个数据点， 它们将实时被加入数据库中。

* fetch.py 系统每天在一个时间段，将今日天气情况、气温，今日窗口总点赞数统计，今日窗口总浏览量统计（都为累加值）统一记录在数据库中。
* generate.py 用于生成大量历史排队人数数据。
* generateWeather.py 用于生成大量模拟天气数据。
* generateLike.py 用户生成大量模拟窗口点赞数数据。
* generateHistory.py 用户生成大量模拟窗口浏览数数据。
* 排队人数预测：

对某一时段（早、午、晚）窗口排队人数进行预测。预测算法将前8周对应的该礼拜数、时间段数的90 \* 8个排队人数数据点、天气类型、气温、窗口点赞数、窗口排队人数作为参数，进行岭回归拟合，并考虑到容纳峰值，进行预测。在每个时段结束后对下一时段进行预测。

* ridgePredict.py 用岭回归进行排队人数预测，删除数据库中对于上一时段的预测，并将其计入数据库。
* 时间预测：

给每个窗口设置一个窗口平均出菜时间，结合实时计数算出等待时间。

* 实时可视化
* fetchPredict.py 后端提供3个排队人数数据接口，分别为”data/now”用于获取实时数据，”data/yesterday”用于获取昨日数据，”data/predict”，用于获取预测数据。
* 图表具体实现

1. **窗口图表：**

前端每两分钟向后端请求实时数据刷新一次数据。可视化图表同时提供窗口实时数据的折线以及历史数据的折线。图表分为实时模式与累计模式，历史模式分为昨日模式和预测模式。

1. **总图表：**

提供全部的餐厅、窗口选项，通过手动选择餐厅、窗口获取该窗口的可视化图表。其余交互信息及接口与窗口图表一致。

1. **对比图表：**

可视化图表同时提供两个窗口实时数据的两条对比折线，不提供动态刷新功能。图表分为实时模式和累计模式。

全局（导航栏）菜品搜索设计详情：

对于键入的关键词在实时时段的菜品的名称、提示以及其所有标签中搜索，列出所有符合条件的菜品。对于每条搜索结果，提供菜品的基本信息（名称、价钱、提示、标签等），并分别提供其所属的餐厅、窗口的跳转链接。

用户身份设计详情：

用户身份分为普通用户、管理员与食堂工作人员。为其提供不同的登录入口，仅为普通用户提供注册接口。对存入数据库的用户密码进行加密。普通用户、管理员与食堂工作人员分别有不同的个人中心，并都可进入其余的菜品浏览等页面。