



Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет имени
Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Робототехники и комплексной автоматизации»
КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА по дисциплине «Базы данных»

| | |
|----------|---|
| Студент: | Новокшанов Евгений Андреевич |
| Группа: | РК6-56Б |
| Тема: | Разработка информационной системы ресторана |

Студент

подпись, дата

Новокшанов Е.А.
Фамилия, И.О.

Преподаватель

подпись, дата

Фамилия, И.О.

Москва, 2022

1. Разработка инфологической модели предметной области;
2. Разработка логической модели базы данных;
3. Разработка запросов;
4. Разработка процедур для создания отчетности;
5. Разработка приложения конечного пользователя.

Содержание

| | |
|---|----------|
| Разработка информационной системы ресторана | 5 |
| Техническое задание | 5 |
| Этап проектирования | 5 |
| Этап проектирования | 5 |
| Описание предметной области | 6 |
| Конечные пользователи | 7 |
| UML-диаграмма вариантов использования | 7 |
| 1 Описание работы с запросами | 8 |
| Карточка варианта использования | 8 |
| Сценарий конечного пользователя(менеджер): | 8 |
| Исключения: | 8 |
| UML-диаграмма сценария работы с запросами: | 9 |
| Файловая структура работы с запросами: | 10 |
| Требование к шаблону: | 11 |
| 2 Описание авторизации | 12 |
| Карточка варианта использования | 12 |
| Сценарий конечного пользователя(менеджер): | 12 |
| Исключения: | 12 |
| UML-диаграмма сценария авторизации: | 13 |
| Файловая структура авторизации: | 14 |
| Требование к шаблону: | 15 |
| 3 Описание работы с отчетами | 16 |
| Карточка варианта использования | 16 |
| Сценарий конечного пользователя(менеджер)(создание): | 16 |
| Сценарий конечного пользователя(менеджер)(просмотр): | 16 |
| Сценарий конечного пользователя(менеджер)(обновление): | 16 |
| Исключения: | 17 |
| UML-диаграмма сценария работы с отчетами: | 17 |
| Файловая структура работы с отчетами: | 18 |
| Требование к шаблону: | 19 |
| 4 Описание работы с заказом | 20 |
| Карточка варианта использования | 20 |
| Сценарий конечного пользователя(менеджер)(заполнение): | 20 |
| Сценарий конечного пользователя(официант)(очистка): | 20 |
| Сценарий конечного пользователя(официант)(оформление заказа): | 20 |
| Исключения: | 21 |
| UML-диаграмма сценария работы с заказами: | 21 |
| Файловая структура работы с отчетами: | 22 |
| Требование к шаблону: | 23 |
| 5 Файловая структура | 24 |
| 6 Инфологическая модель работы ресторана | 25 |

| | |
|-----------------------------|----|
| Логическая модель | 25 |
| Заключение | 26 |

Разработка информационной системы ресторана

Техническое задание

Этап проектирования

1. Определить конечных пользователей будущей системы.
2. Составить UML-диаграмму вариантов использования.
3. Выделить основной вариант использования информационной системы (основной бизнес-процесс в предметной области).
4. Разработать систему авторизации пользователей ИС.
5. Разработать системную архитектуру ИС.
6. Для всех вариантов использования разработать главные успешные сценарии и расширения к ним.
7. Разработать системные UML-диаграммы последовательности для всех сценариев с использованием MVC-паттерна.
8. Разработать требования ко всем шаблонам для каждого варианта использования.
9. Разработать инфологическую модель предметной области в форме UML-диаграммы классов.
10. Разработать логическую модель будущей базы данных.

Этап проектирования

1. Реализовать разработанную на этапе проектирования информационную систему на языке Python в среде фреймворка Flask.
2. Каждый вариант использования оформить, как блюпринт.
3. Доступ конечных и внешних пользователей к вариантам использования реализовать с помощью декораторов.

Описание предметной области

В ресторане официанты принимают заказы от посетителей. Об официантах в БД хранятся следующие данные: уникальный номер, паспортные данные, дата приема на работу. Предусмотрена также дата увольнения, которая для работающих официантов не заполняется. Пространство ресторана поделено на зоны (количество столиков, vip, для курящих/нет, со специальными детскими местами и пр.). В каждой зоне располагаются столы. Посетителям за столами предлагается меню. В меню указывается шифр блюда, название блюда, цена, вес в граммах. Любой свободный официант может принять заказ. В заказе указывается номер стола, дата и время заказа. Сохраняется также общая стоимость заказа, которая вычисляется при окончательном расчете с посетителем. Заказ состоит из строк заказа. Каждая строка заказа ссылается на один из пунктов меню и показывает количество заказанных блюд.

Конечные пользователи

1. Официант;
2. Директор;
3. Менеджер;
4. Шеф-повар;
5. Повар.

UML-диаграмма вариантов использования

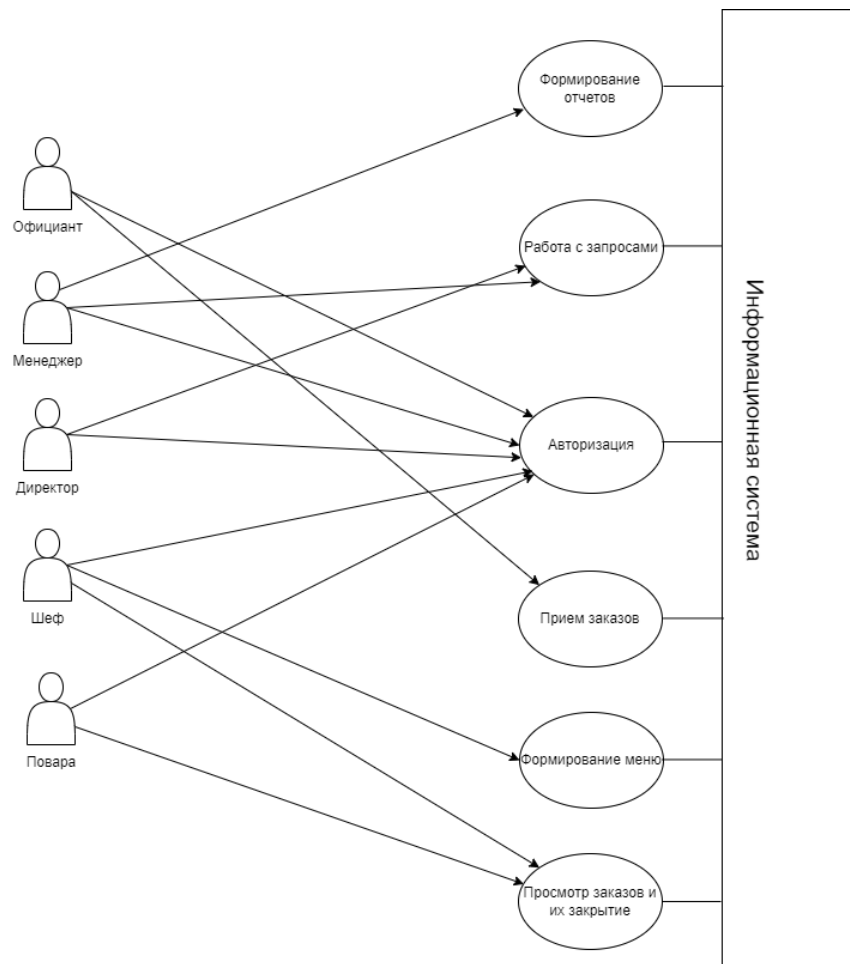


Рис. 1. UML-диаграмма вариантов использования

1 Описание работы с запросами

Карточка варианта использования

Название: Выполнение запросов.

Предусловие: Пользователь успешно вошел в систему и авторизовался.

Гарантия: получение результата выполненного запроса.

Минимальная гарантия: получение сообщения об ошибке. База данных осталась в согласованном состоянии.

Сценарий конечного пользователя(менеджер):

1. Пользователь выбирает запрос из меню запросов
2. Пользователь начинает работу с выбранным запросом
3. Пользователь вводит параметры и отправляет в систему
4. Система выполняет запрос и отправляет результат

Исключения:

1. Пользователь задает неверный тип данных.
2. Пользователь отправляет пустой запрос.
3. Пользователь вводит неверную дату.

UML-диаграмма сценария работы с запросами:

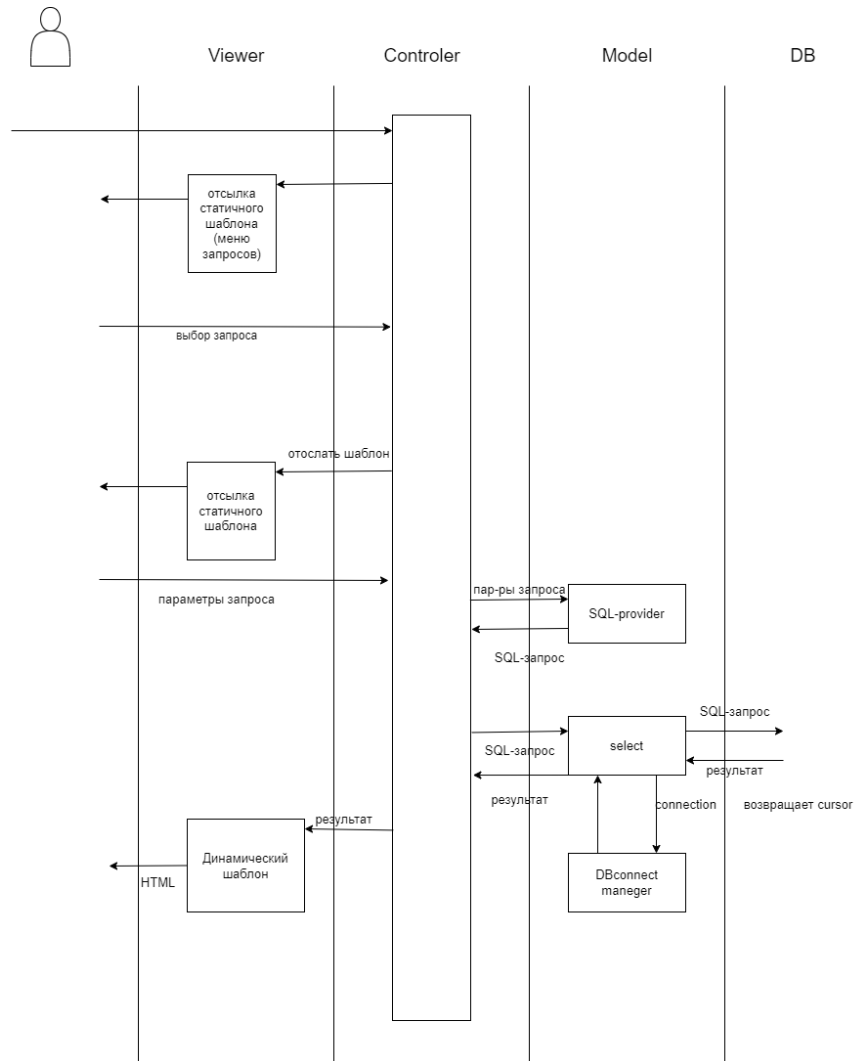


Рис. 2. UML-диаграмма варианта использования: работы с запросами

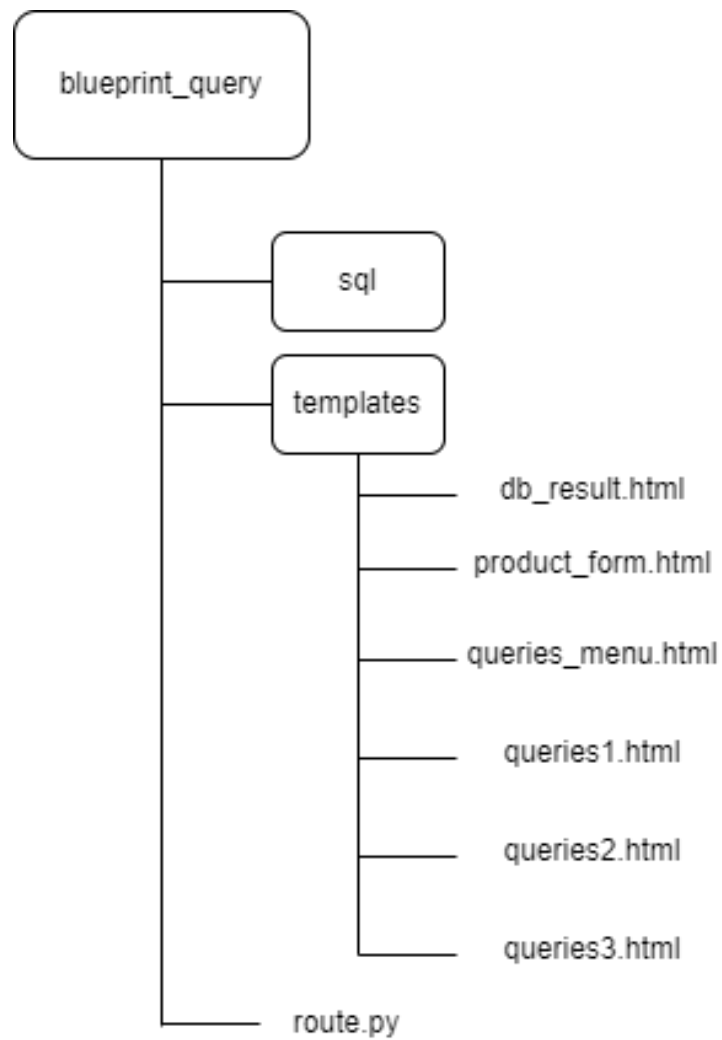
Файловая структура работы с запросами:

Рис. 3. Файловая структура варианта использования "Работа с запросами"

Требование к шаблону:

- Меню запросов:

Данная страница представляет из себя список ссылок к следующим запросам:

- Просмотр меню ресторана (ссылка должна формироваться с помощью функции `url_for('bp_query.queries1')` для перехода к с следующей странице);
- Поиск официантов (ссылка должна формироваться с помощью функции `url_for('bp_query.queries2')` для перехода к с следующей странице);
- Сумма заказов за месяц (ссылка должна формироваться с помощью функции `url_for('bp_query.queries2')` для перехода к с следующей странице).

Должен быть предусмотрен выход в главное меню адрес которого формируется с помощью следующей функции `url_for('menu_choice')`.

- Запрос на просмотр меню ресторана:

Входные данные: список с названиями блюд, тип: `list` и `string` соответственно.

Должен формироваться выпадающий список из соответствующих элементов списка.

Выходные данные: сервер с помощью функции `request.form.get` принимает название блюда по ключу `"product_name"`;

- Запрос на получение информации о официантах:

На странице должен быть предусмотрено поле ввода фамилии или знака `%` (для получения информации о всех официантах).

На вход система получает строку с фамилией официанта или знак `%`, для дальнейшего формирования `SELECT`-запроса, в который подставляется строка. Полученный список с наименованиями колонок и кортеж с результатом выполнения запроса передается в шаблон с результатом выполнения запроса.

- Запрос на получение информации о сумме заказов за определенный месяц:

На странице должен быть предусмотрен ввод месяца и года в виде календаря. Выходные данные: строка с датой в формате `уууу-мм`. Далее формируется `SELECT`-запрос, в который подставляется строка. Полученный список с наименованиями колонок и кортеж с результатом выполнения запроса передается в динамический шаблон с результатом выполнения запроса.

- Результат выполнения запроса: Входные данные: список(тип `list`) с наименованиями колонок и кортеж(тип `tuple`) из списков с соответствующими данными. На странице должна сформироваться таблица, для визуализации работы запроса и полученного результата.

2 Описание авторизации

Карточка варианта использования

Название: Авторизация

Пользователь находится на странице для ввода логина и пароля.

Гарантия: пользователь успешно авторизовался и получил доступ к каталогу действий доступных ему.

Минимальная гарантия: получение сообщения об ошибке.База данных осталась в согласованном состоянии.

Сценарий конечного пользователя(менеджер):

1. Пользователь авторизуется: вводит логин и пароль для соответствующей группы пользователей.
2. Система проверяет наличие данного пользователя в базе данных
3. Система перенаправляет пользователя на следующую страницу с каталогом возможных действий для его группы пользователей.

Исключения:

1. Пользователь входит под неправильным логином или паролем.
2. Пользователь передает пустые данные.

UML-диаграмма сценария авторизации:

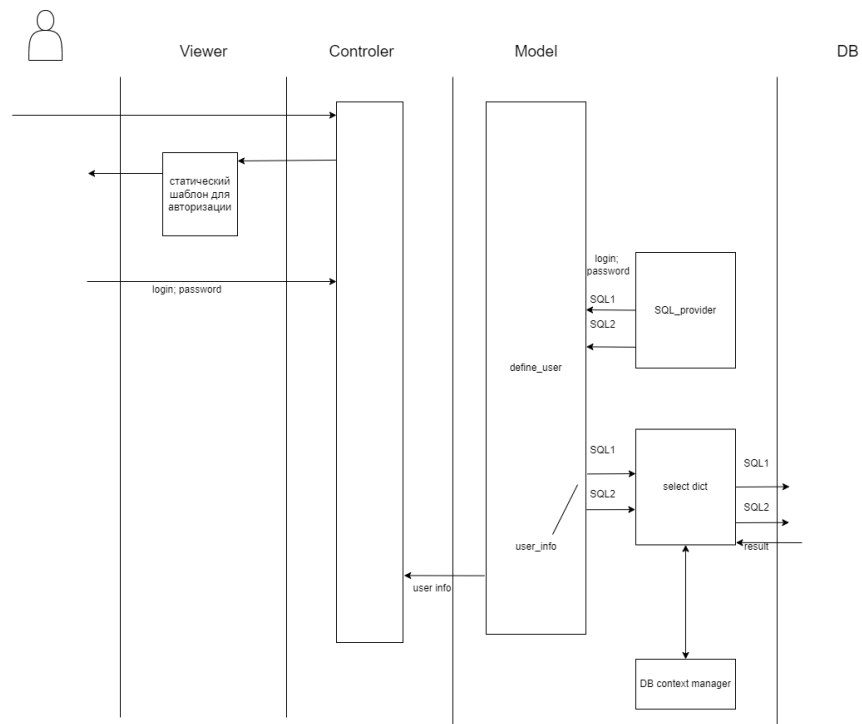


Рис. 4. UML-диаграмма варианта использования: авторизация

Файловая структура авторизации:

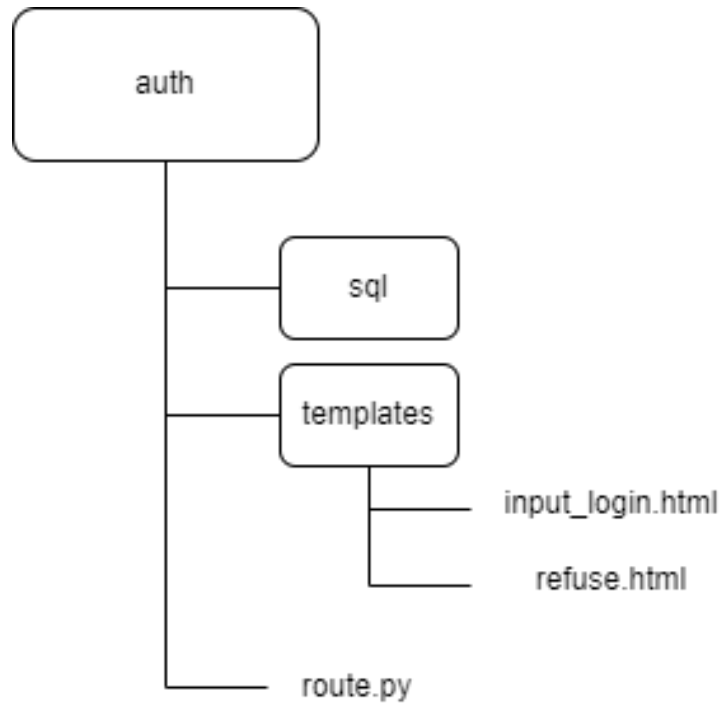


Рис. 5. Файловая структура варианта использования "Авторизация"

Требование к шаблону:

1. Ввода логина и пароля:

На странице располагается 2 поля ввода для логина и пароля. Выходные данные: Логин и пароль в виде строк(тип string) передаются в SELECT-запрос, по результатам которого система определяет к какой группе относится пользователь и отправляет ему шаблон с каталогом возможных действий, иначе пользователь получает сообщение с просьбой повторного ввода логина и пароля.

2. Каталог действий:

В зависимости от того, авторизовался ли пользователь, формируется шаблон со ссылками сформированными функцией `url_for()` на соответствующие обработчики:

- `'bp_query.querie'`;
- `'bp_report.start_report'`;
- `'bp_order.waiter_list'`.

А так же ссылка на шаблон ввода логина и пароля: `'blueprint_auth.start_auth'`.

3 Описание работы с отчетами

Карточка варианта использования

Название: Работа с отчетами

Предусловие:

1. в БД создается табличка для каждого типа отчетов;
2. Создана процедура для создания отчета.

Гарантия: при введении нового отчетного периода создаются записи в таблице отчетов.

Минимальная гарантия: получение сообщения об ошибке и есть возможность продолжить работу. База данных осталась в согласованном состоянии.

Сценарий конечного пользователя(менеджер)(создание):

1. Пользователь начинает работу. Система выводит страничку со всеми отчетами, есть кнопки "создать "просмотреть" и "обновить".
2. Пользователь нажимает кнопку "создать". Система присылает форму для ввода периода.
3. Система отвечает сообщением об успешном создании отчета и предлагает вернуться в меню отчетов.

Сценарий конечного пользователя(менеджер)(просмотр):

1. Пользователь начинает работу. Система выводит страничку со всеми отчетами, есть кнопки "создать "просмотреть" и "обновить".
2. Пользователь нажимает кнопку "просмотреть". Система присылает форму для ввода периода.
3. Система отправляет шаблон с результатом выполнения запроса по просмотру отчета за данный период.

Сценарий конечного пользователя(менеджер)(обновление):

1. Пользователь начинает работу. Система выводит страничку со всеми отчетами, есть кнопки "создать "просмотреть" и "обновить".
2. Пользователь нажимает кнопку "обновить". Система присылает форму для ввода периода.
3. Система отвечает сообщением об успешном обновлении отчета и предлагает вернуться в меню отчетов.

UML-диаграмма сценария работы с отчетами:

1. Пользователь обновляет еще не созданный отчет.
2. Пользователь создает отчет с пустой датой.
3. Пользователь вызывает просмотр не созданного отчета.

UML-диаграмма сценария работы с отчетами:

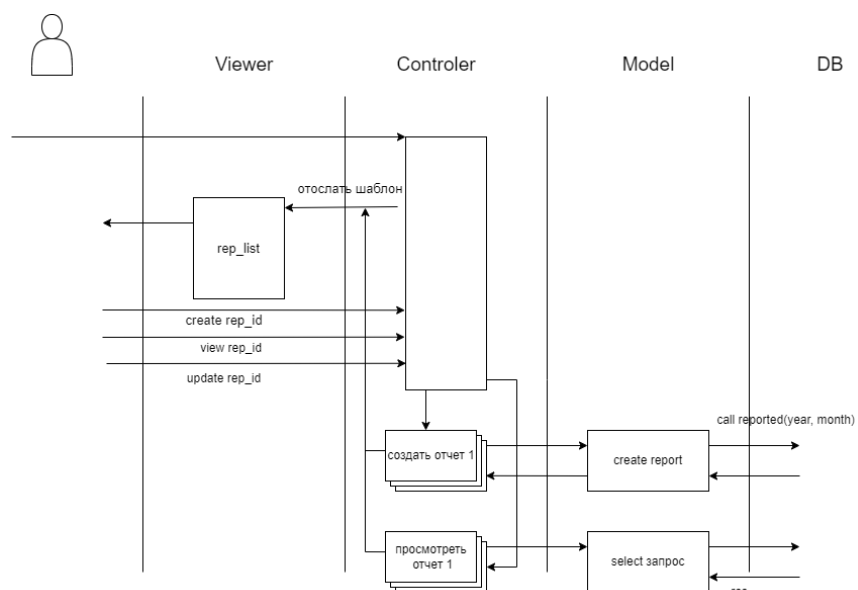


Рис. 6

Файловая структура работы с отчетами:

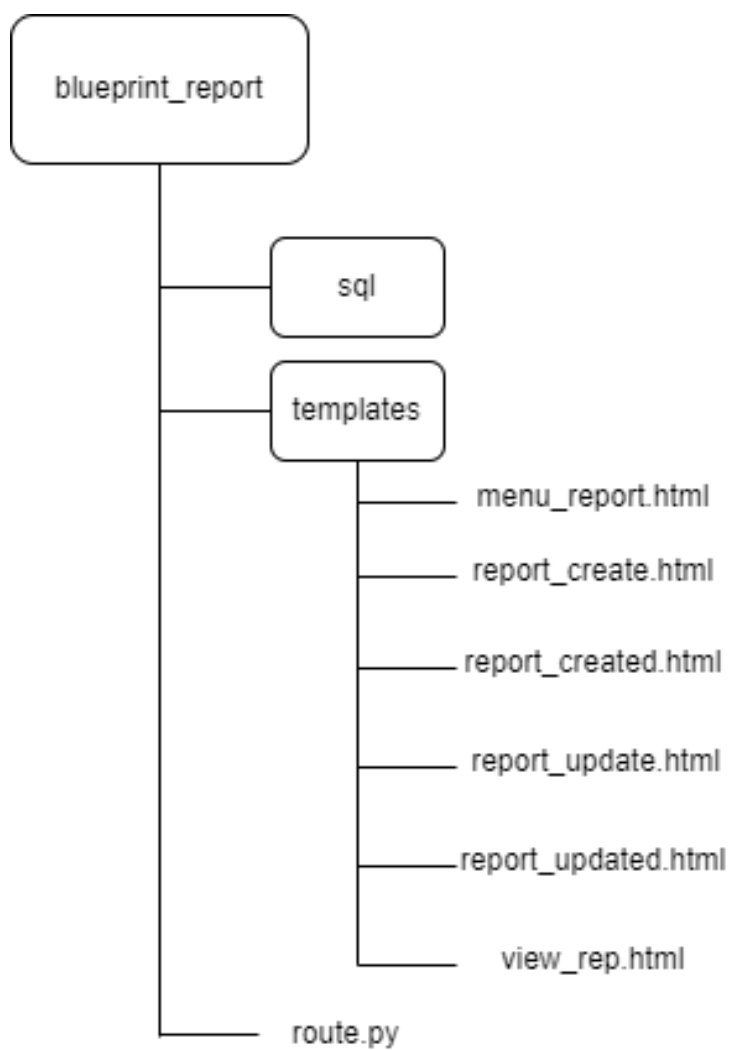


Рис. 7. Файловая структура варианта использования "Работа с отчетами"

Требование к шаблону:

1. Меню запросов:

Входные данные:

Передается список(тип list) словарей(тип dict): Ключи в словаре:

- "rep_name"
- "rep_id"

Далее должен быть сформирован список отчетов с соответствующими именами и номерами.

Выходные данные:

Сервер получает POST запрос и с помощью функции `request.form.get` принимает ключ для формирования адреса обработчика с помощью `url_for`. Ключ имеет следующий вид:

- `create_rep` для кнопки создания
- `view_rep` для кнопки просмотра
- `update_rep` для кнопки обновления

Должен быть предусмотрен выход в главное меню адрес которого формируется с помощью следующей функции `url_for('menu_choice')`.

2. Шаблон создания отчета, обновления и просмотра имеют одинаковую структуру, входные и выходные данные:

Структура состоит из 2-х полей ввода года и месяца. Выходные данные:

При получении POST запроса сервер с помощью функции `request.form.get` считывает год с ключом `input_year` и месяц с ключом `input_month`, которые необходимы для формирования и занесения отчета за этот месяц в базу данных.

3. Шаблон просмотра результата:

Входные данные: список(тип list) с наименованиями колонок и кортеж(тип tuple) из списков с соответствующими данными. На странице должна сформироваться таблица, для визуализации работы запроса и полученного результата.

4 Описание работы с заказом

Карточка варианта использования

Название: Работа с заказом

Предусловие:

1. в базе данных есть таблица с заказами и строками заказов;
2. пользователь успешно авторизовался под ролью официанта.

Гарантия: при заполнении корзины и нажатии кнопки "Оформить заказ" система заполняет соответствующие таблицы и выдает пользователю сообщение об успешно созданном заказе.

Минимальная гарантия: получение сообщения об ошибке и есть возможность продолжить работу. База данных осталась в согласованном состоянии.

Сценарий конечного пользователя(менеджер)(заполнение):

1. Пользователь начинает работу с системой.
2. Система отправляет шаблон с заказом.
3. Пользователь нажимает кнопку "добавить".
4. Система присылает тот же шаблон, но с содержимым заказа.
5. Пользователь нажимает кнопку "оформить заказ". Система перенаправляет пользователя на новую страницу с сообщением об успешном создании заказа.

Сценарий конечного пользователя(официант)(очистка):

1. Пользователь начинает работу с системой.
2. Система отправляет шаблон с заказом.
3. Пользователь нажимает кнопку "очистить".
4. Система присылает тот же шаблон, но с пустым содержимым заказа.

Сценарий конечного пользователя(официант)(оформление заказа):

1. Пользователь начинает работу с системой.
2. Система отправляет шаблон с заказом.
3. Пользователь нажимает кнопку "оформить заказ".
4. Система перенаправляет пользователя на страницу с сообщением об успешно созданном заказе.

Исключения:

1. Пользователь пытается оформить пустой заказ.
2. Пользователь пытается ввести id столика или официанта, которых нет.
3. Пользователь пытается выходит из заполнения заказа, не оформив его.

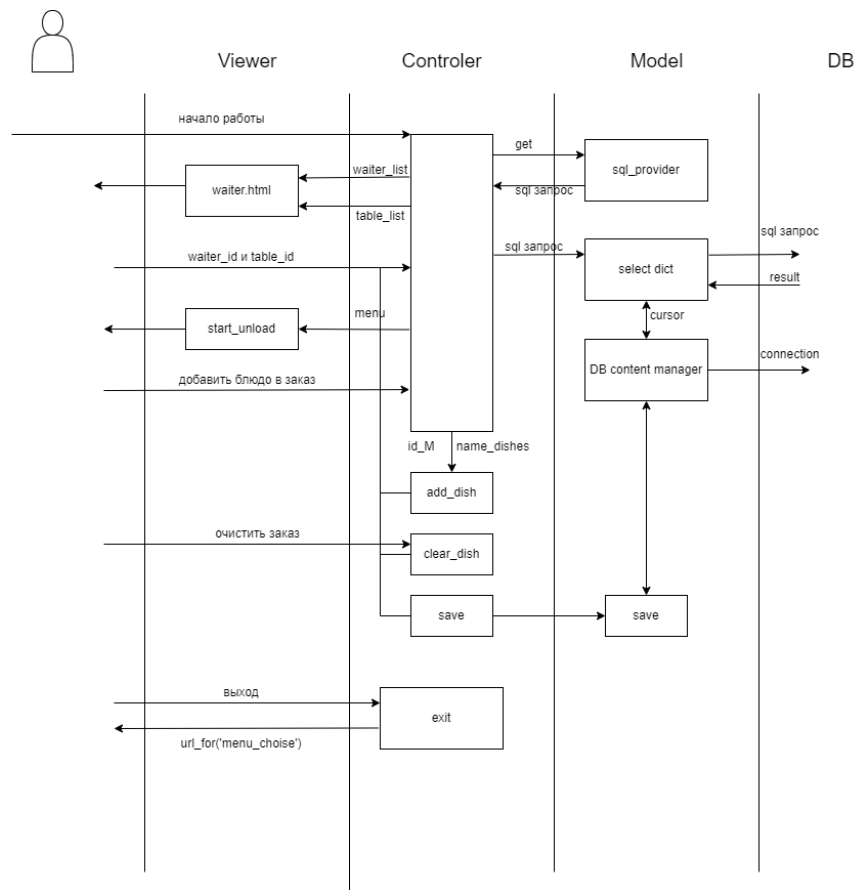
UML-диаграмма сценария работы с заказами:

Рис. 8. UML-диаграмма варианта использования: работа с заказами

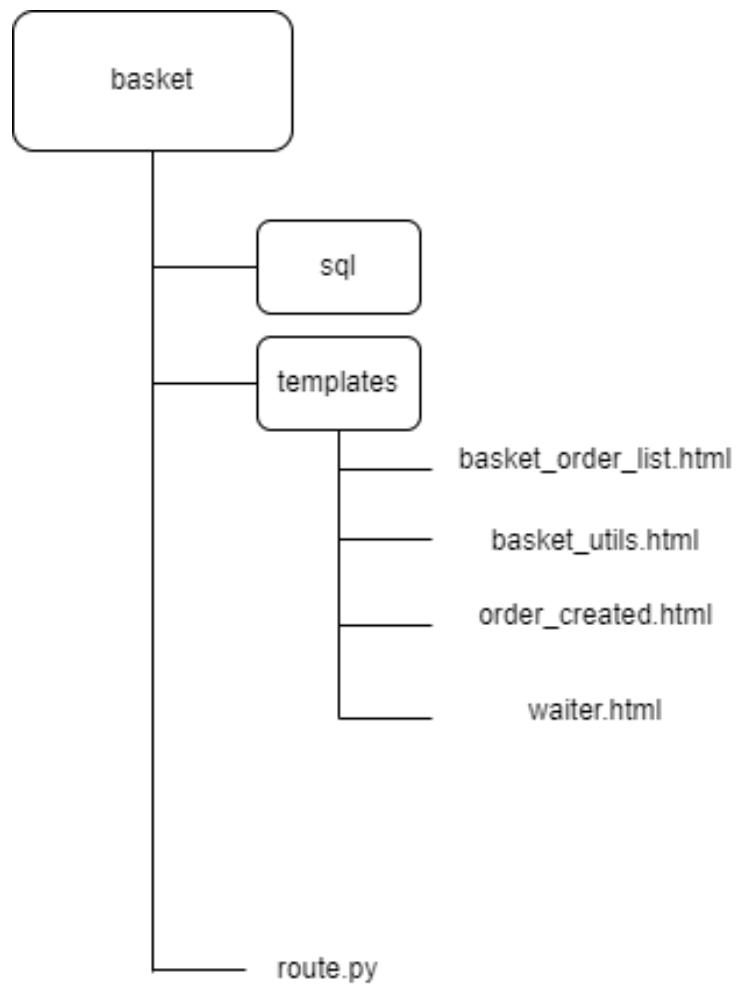
Файловая структура работы с отчетами:

Рис. 9. Файловая структура варианта использования "Работа с заказами"

Требование к шаблону:

1. Шаблон заказа

- Список блюд:

Входные данные: словарь с именем `items` с ключами `'name_dishes'`, `'price'`, `'amount'` и `'id_M'`. Должен формироваться список блюд с ценой. Выходные данные: Сервер обрабатывает страницу методом POST. С помощью `request.form.get` считывается `items['id_M']` с ключем `'prod_id'`.

- Список блюд добавленных в заказ:

Входные данные: словарь с именем `items` с ключами `'name_dishes'`, `'price'`, `'amount'` и `'id_M'`. Должен формироваться список блюд с ценой и количеством.

Так же необходимо предусмотреть выход в главное меню.

2. Идентификация официанта:

Входные данные: список(тип `list`) с именем `'items'`. В шаблоне выпадающий список должны храниться с ключем `'waiter_id'`.

список(тип `list`) с именем `'items_T'`. В шаблоне выпадающий список должны храниться с ключем `'table_id'`. Выходные данные: При обработке методом POST сервер получает номер официанта и номер столика с помощью `request.form.get` с ключами `'waiter_id'` и `'table_id'` соответственно.

5 Файловая структура

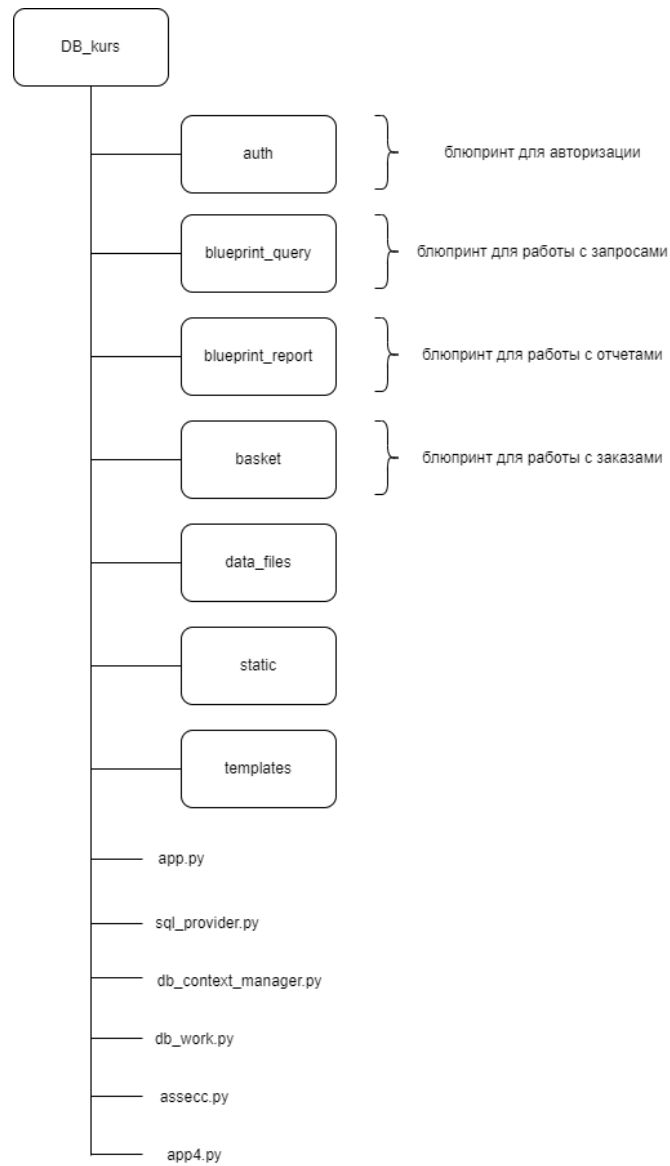


Рис. 10. Файловая структура проекта

6 Инфологическая модель работы ресторана

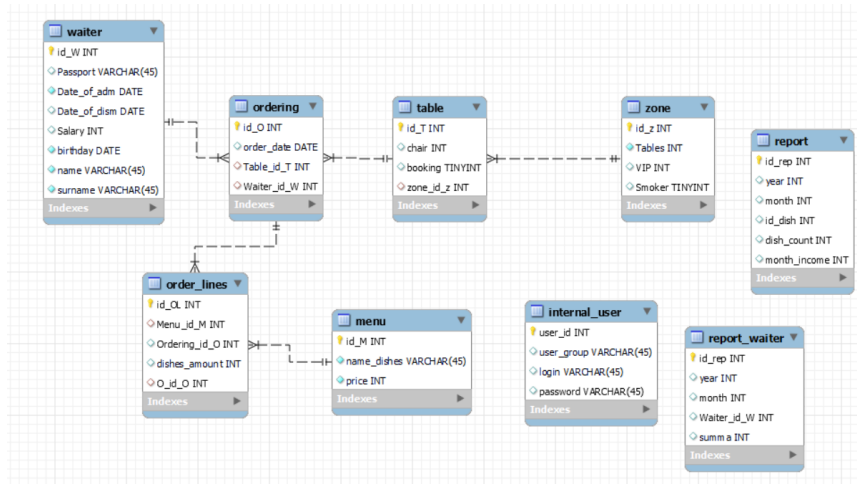


Рис. 11. UML-диаграмма(инфологическая модель)

Логическая модель

| | | | | | |
|----|-------------|-----------------------|-------------------------------|-------------------------|---------------------------|
| 1 | Ordering | | | | |
| 2 | id_o | Order amount | Order_date | Table_id_T(Table, id_t) | Waiter_id_W(Waiter, id_w) |
| 3 | PK | | | FK | FK |
| 4 | | | | | |
| 5 | | | | | |
| 6 | | | | | |
| 7 | Table | | | | |
| 8 | | | | | |
| 9 | id_t | chair | booking | Zone_id_z(Zone, id_z) | |
| 10 | PK | | | FK | |
| 11 | | | | | |
| 12 | | | | | |
| 13 | | | | | |
| 14 | | | | | |
| 15 | | | | | |
| 16 | Waiter | | | | |
| 17 | id_W | Passport | Date_of_adm | Date_of_dism | Salary |
| 18 | PK | | | | |
| 19 | | | | | |
| 20 | | | | | |
| 21 | | | | | |
| 22 | | | | | |
| 23 | | | | | |
| 24 | Order_lines | | | | |
| 25 | id_O | Menu_id_M(Menu, id_M) | Ordering_id_O(Ordering, id_o) | | |
| 26 | PK | FK | FK | | |
| 27 | | | | | |
| 28 | | | | | |
| 29 | | | | | |
| 30 | | | | | |
| 31 | Menu | | | | |
| 32 | id_M | name_dishes | price | | |
| 33 | PK | | | | |
| 34 | | | | | |
| 35 | | | | | |
| 36 | | | | | |
| 37 | | | | | |
| 38 | Zone | | | | |
| 39 | id_z | Tables | VIP | Smoker | |
| 40 | PK | | | | |
| 41 | | | | | |
| 42 | | | | | |



Рис. 12. Логическая модель работы БД

Заключение

В результате выполнения курсовой работы была разработана информационная система (далее ИС) для базы данных ресторана, рассчитанная на конечных пользователей системы. Для этого была создана структура базы данных на сервере MySQL, а так же разработано веб-приложение на языке Python 3.9 и HTML5 с использованием фреймворка Flask, использующий набор инструментов Werkzeug, а также шаблонизатор Jinja2.

При разработке ИС были разработаны модели предметной области и написана документация. Основной бизнес-процесс курсовой работы включает в себя составление заказов посетителей ресторана. Кроме этого была реализована авторизация внутренних пользователей, а также их возможность работы с отчетами и запросами.

Выходные данные

Постановка:  преподаватель кафедры РК6, Пивоварова Н.В.
 Решение и вёрстка:  студент группы РК6-56Б, Новокшанов Е.А.

2022, осенний семестр