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

ФАКУЛЬТЕТ «Робототехника и комплексная автоматизация»

КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

«Разработка информационной системы для заказов банкетов»

2024 г.

Оглавление

Техническое задание

Техническое задание на этап проектирования

Техническое задание на этап проектирования состоит из следующих пунктов:

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.

Техническое задание на этап реализации

Техническое задание на этап реализации состоит из следующих задач:

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

Описание предметной области.

Требуется создать информационную систему для заказов банкета: реализовать выполнение запросов и создание отчетов о работе сотрудников, разработать систему пользователя, программную архитектуру приложения.

В ресторане менеджеры принимают предварительные заказы на организацию банкетов.

О менеджерах в БД хранятся следующие данные: уникальный номер, паспортные данные, дата приема на работу. Предусмотрена также дата увольнения, которая для работающих менеджеров не заполняется.

Для организации банкетов ресторан располагает несколькими залами.

Каждый зал имеет уникальный номер и название. Известно также максимальное количество посадочных мест в зале.

В ресторане имеется меню. В меню указывается шифр блюда, название блюда, цена, вес в граммах.

При оформлении заказа указывается менеджер, обслуживающий банкет, дата и время банкета, предполагаемое количество участников. Выбирается один из доступных залов.

К каждому заказу прилагается предварительный список блюд, выбранных из меню и их количество.

После составления списка блюд подсчитывается предварительная стоимость и вносится аванс, размер которого сохраняется в заказе. После проведения банкета подсчитывается реальная стоимость, которая также сохраняется в заказе.

В информационной системе реализована авторизация для внешних и внутренних пользователей, на основе которой каждому пользователю присваивается роль, соответствующая его группе доступа.

Реализована работа с параметризованными запросами, где пользователь может выполнять запросы для введенных параметров и видеть результаты этих запросов, сформированные на основе данных из БД. Также в системе есть и запросы, параметры для которых вводить не нужно.

Также реализована работа с отчетами, где пользователь получает доступ к созданию отчетов и просмотру уже существующих отчетов. Созданные отчеты хранятся в БД, у пользователя есть доступ к вводу параметров, для создания или просмотра конкретного отчета.

Реализован основной бизнес-процесс – составление заказа, который фактически происходит в несколько этапов. Поэтому, логичнее всего, обозначить его, как несколько отдельных действий, произведенных разными (по роли) пользователями. Выполнение данных действий происходит строго в последовательности, указанной ниже:

1. Резервирование зала (пользователь переходит к созданию нового заказа и начинает с того, что резервирует на выбранную дату определенный зал: исходя из выбранного ожидаемого количества человек на банкет)
2. Обработка заказа (пользователь выбирает для конкретного заказа свободного на зарезервированную дату сотрудника (менеджера))
3. Составление списка блюд (фактически работа с корзиной: пользователь составляет предварительный список блюд для банкета, выбирая позиции из предложенного в меню ресторана перечня)
4. Оплата (сразу после составления списка блюд система переводит пользователя на оплату текущего заказа (или он может перейти туда впоследствии сам); тем самым пользователь вносит аванс за заказ)

Далее по хронологии происходит сам банкет

Реализован дополнительный бизнес-процесс – завершение заказа (его действия происходят после проведения банкета, и указаны также, в хронологическом порядке):

1. Редактирование чека заказа (уже после банкета пользователь заходит в систему и добавляет позиции в уже составленный чек заказа (которые не входили в предварительный список); система запоминает реальную стоимость заказа, суммируя новые позиции с суммой внесенного аванса)
   **в случае если за время самого банкета новых позиций не было выбрано, пользователь информирует об этом систему; бизнес-процесс полностью завершается.*
2. Оплата (пользователь оплачивает позиции, сформированные в предыдущем пункте, бизнес-процесс полностью завершается)

**Данные действия распределены между ролями согласно UML-диаграмме.*

Проектирование информационной системы.

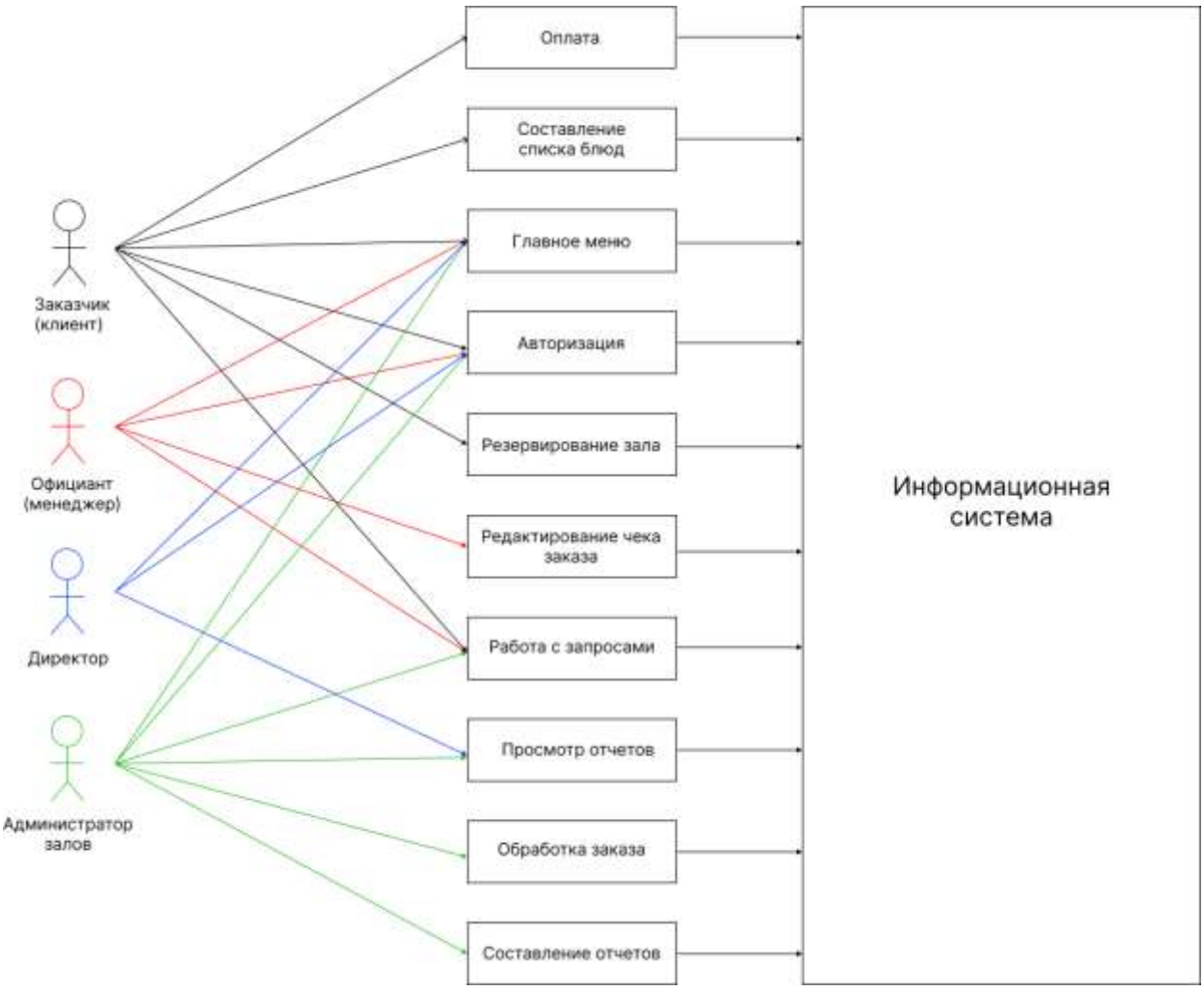
Определение конечных пользователей.

В данной информационной системе в качестве конечных пользователей выделим следующих актеров, приняв менеджера за банкетного менеджера (далее – официанта, для большей наглядности) для большей наглядности его роли:

- Заказчик (клиент): «*client*»
- Банкетный менеджер (официант): «*manager*»
- Директор: «*director*»
- Администратор залов: «*hall_admin*»

Официант, директор и администратор залов являются внутренними пользователями. Заказчик является внешним пользователем. Иерархия внутренних пользователей (в порядке возрастания): официант – директор – администратор залов.

UML-диаграмма вариантов использования



Требования к базовым шаблонам

Для всех html-шаблонов в проекте реализован единый шаблон base.html, от которого впоследствии они наследуются

Требования к base.html:

1. **Функциональные требования:**
   - Должен служить базовым шаблоном для всех страниц
   - Должен поддерживать динамическую смену заголовка страницы
   - Должен подключать специальный CSS файл, который содержит дизайн всех различных форм и атрибутов (кнопок, блоков и др.), которые будут использованы шаблонами-наследниками (main_style.css)
   - Должен отображать навигационное меню
   - Должен поддерживать систему навигационных цепочек (хлебных крошек)
   - Должен отображать flash-сообщения
   - Должен показывать информацию о пользователе и кнопку выхода
2. **Структурные блоки:**
   - title - для заголовка страницы
   - extra_css - для дополнительных стилей
   - make_order_css - для стилей заказов
   - nav_items - для пунктов навигации
   - breadcrumbs - для навигационных цепочек (хлебных крошек)
   - Множество блоков контента (content, exit, base_report и т.д.)
   - footer - для нижней части страницы
   - extra_js - для дополнительных скриптов
3. **Требования к оформлению:**
   - Тёмная навигационная панель (navbar-dark bg-dark)
   - Фиксированное положение навигации (sticky-top)
   - Адаптивный дизайн с использованием Bootstrap
   - Отступы между основными блоками (mt-3, mt-4, mt-5)
   - Темный футер с белым текстом

Подобный дизайн позволяет пользователю легко ориентироваться внутри веб-приложения. На темном футере расположены «никнейм», состоящий из user+уникальный номер, и роль пользователя, а также кнопка с возможностью выйти из системы. Также пользователь всегда может всегда выйти в главное меню, используя переходник «Банкетный зал» внутри футера или «Домой» на навигационной цепочке. Элементы navbar будут варьироваться в зависимости от роли пользователя.

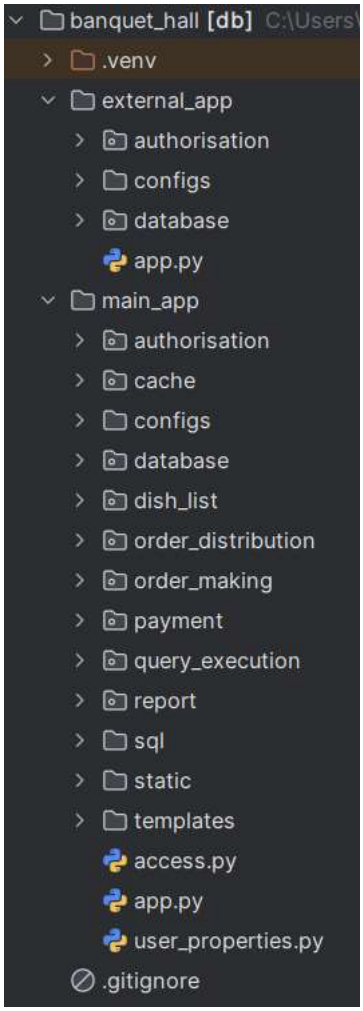
Второй базовый шаблон наследуется от base.html и является так называемым предком для html-шаблонов, отображающих меню для пользователей, у которых на этом меню должны отображаться заказы, относящиеся к конкретному пользователю.

Требования к orders_base.html:

1. **Функциональные требования:**
   - Должен наследоваться от base.html
   - Должен предоставлять структуру для страниц, связанных с заказами
   - Должен поддерживать различные типы меню для разных пользователей
2. **Структурные блоки:**
   - Должен расширять блок content из базового шаблона
   - Должен содержать три подблока для разных типов пользователей:
   - internal_user_menu_for_manager (наследник)
   - internal_user_menu_for_hall_admin (наследник)
   - external_user_menu (наследник)
   - Должен подключать дополнительные стили через блок extra_css
3. **Требования к оформлению:**
   - Должен включать Font Awesome для иконок
   - Должен подключать специальный CSS файл для заказов (orders.css)
   - Должен поддерживать все стили из базового шаблона
4. **Дополнительные требования:**
   - Все меню должны быть стилистически согласованы
   - Должна быть обеспечена совместимость с системой уведомлений из базового шаблона
   - Должен корректно отображаться на всех устройствах

Следует также отметить два вспомогательных шаблона: **access_error.html** и **exit.html.** Оба наследники base.html и имеют только одну кнопку с переадресацией на главное меню. Первый служит для отображения сообщения, которое информирует пользователя об отказе в доступе (если тот захотел зайти в вариант использования, недоступный для своей роли). Второй служит для отображения сообщения, которое информирует о выходе из системы, если пользователь осуществил выход.

Файловая архитектура проекта



Вариант использования: «Авторизация»

Предусловия: пользователь должен быть зарегистрирован в ИС. При попытке входа в систему неавторизованный пользователь будет перенаправлен на данный вариант использования.

Гарантия: пользователь введет свои логин и пароль, перейдет на главное меню и сможет пользоваться системой в соответствии со своей ролью.

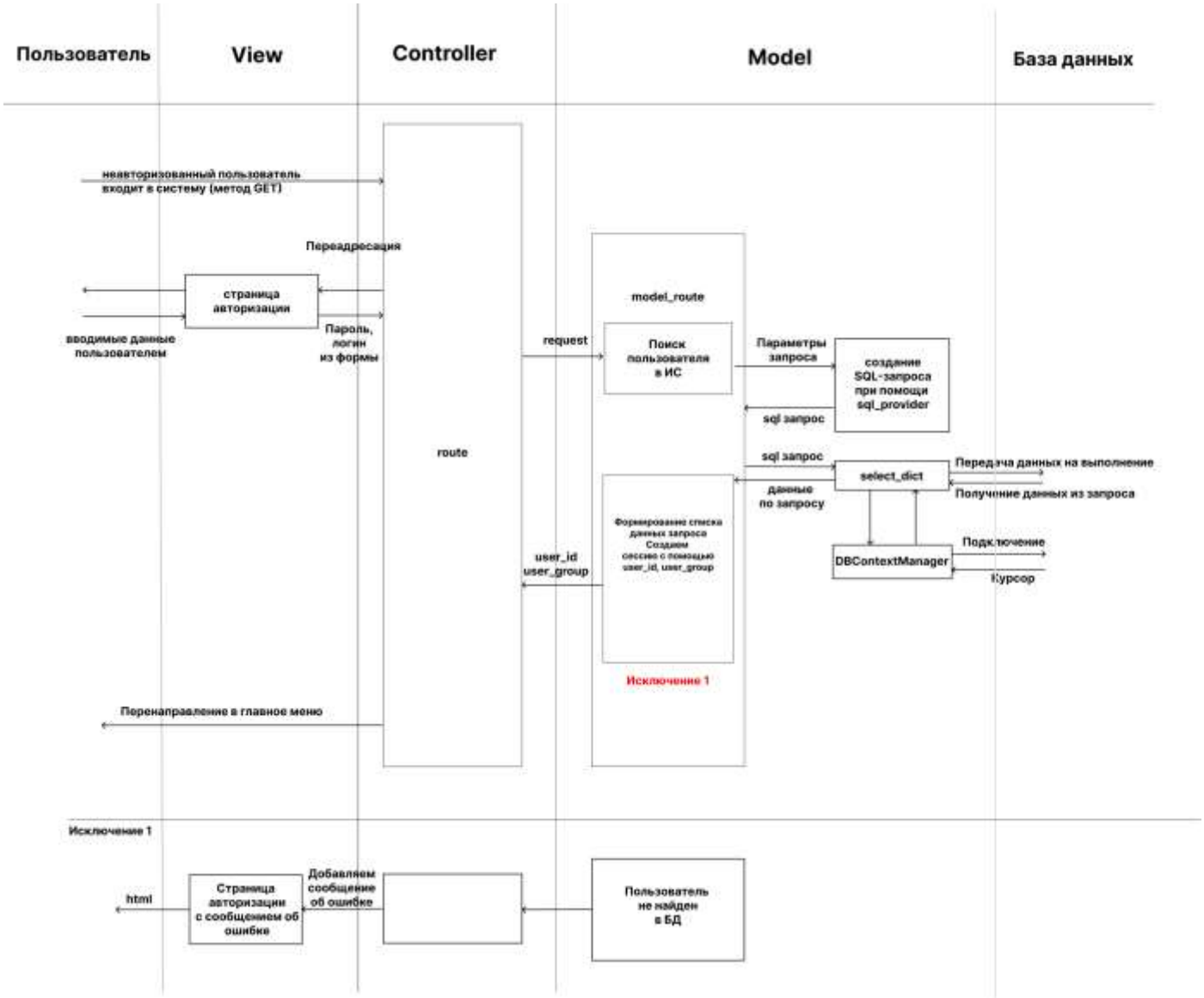
Минимальная гарантия: пользователь не войдет в систему (увидит соответствующую ошибку и сможет продолжить работу).

Успешный сценарий работы авторизации:

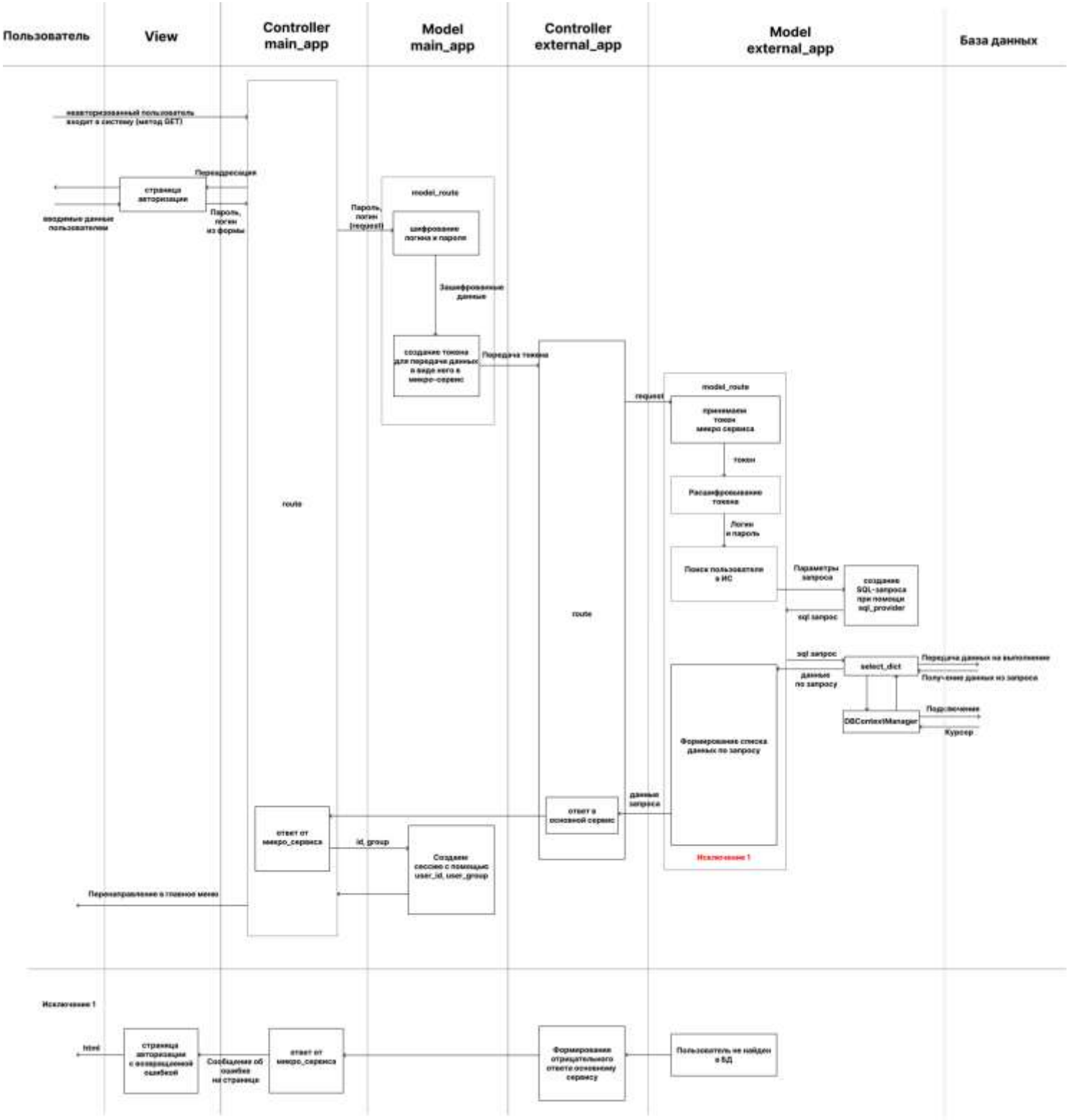
1. Пользователь входит в систему.
2. Появляется форма для авторизации.
3. Пользователь вводит логин и пароль.
4. Система перенаправляет пользователя на Главное меню.

Исключения:

1. Пользователя с такими данными нет в БД – система сообщает об ошибке и возвращает ко 2-му пункту.

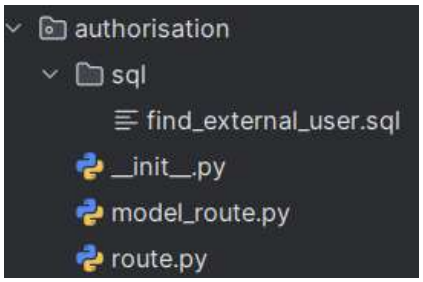


Sequence Diagram для MVC авторизации внутренних пользователей

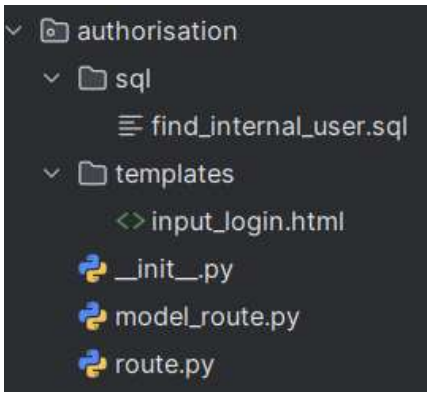


Sequence Diagram для MVC авторизации внешних пользователей

Файловая архитектура реализации варианта использования



Авторизация для внешних пользователей



Авторизация для внутренних пользователей

Требования к шаблону input_login.html

Назначение: Страница авторизации пользователей.
Наследование: Наследуется от шаблона base.html
Должен содержать:

1. breadcrumbs:
   - Текст "Авторизация"
2. Основной контент (в блоке base_auth):
   - Заголовок "Авторизация" (выровненный по центру)
   - Блок для отображения сообщений об ошибках:
     - ✓ Переменная message для вывода текста ошибки
3. Форма авторизации (выровненная по центру):
   - Метод отправки: POST
   - Поля формы:
     - ✓ Текстовое поле "login" с меткой "Логин"
     - ✓ Поле пароля "password" с меткой "Пароль"
     - ✓ Чекбокс "is_internal" с меткой "Сотрудник"
     - ✓ Кнопка отправки формы с текстом "Отправить"

Особые требования к элементам:

- Использование кастомных CSS-классов:
  - ✓ form-control-custom для полей ввода
  - ✓ checkbox-custom для стилизации чекбокса
  - ✓ btn-custom и btn-primary-custom для кнопки отправки
- Все поля формы должны иметь соответствующие id, совпадающие с их name-атрибутами

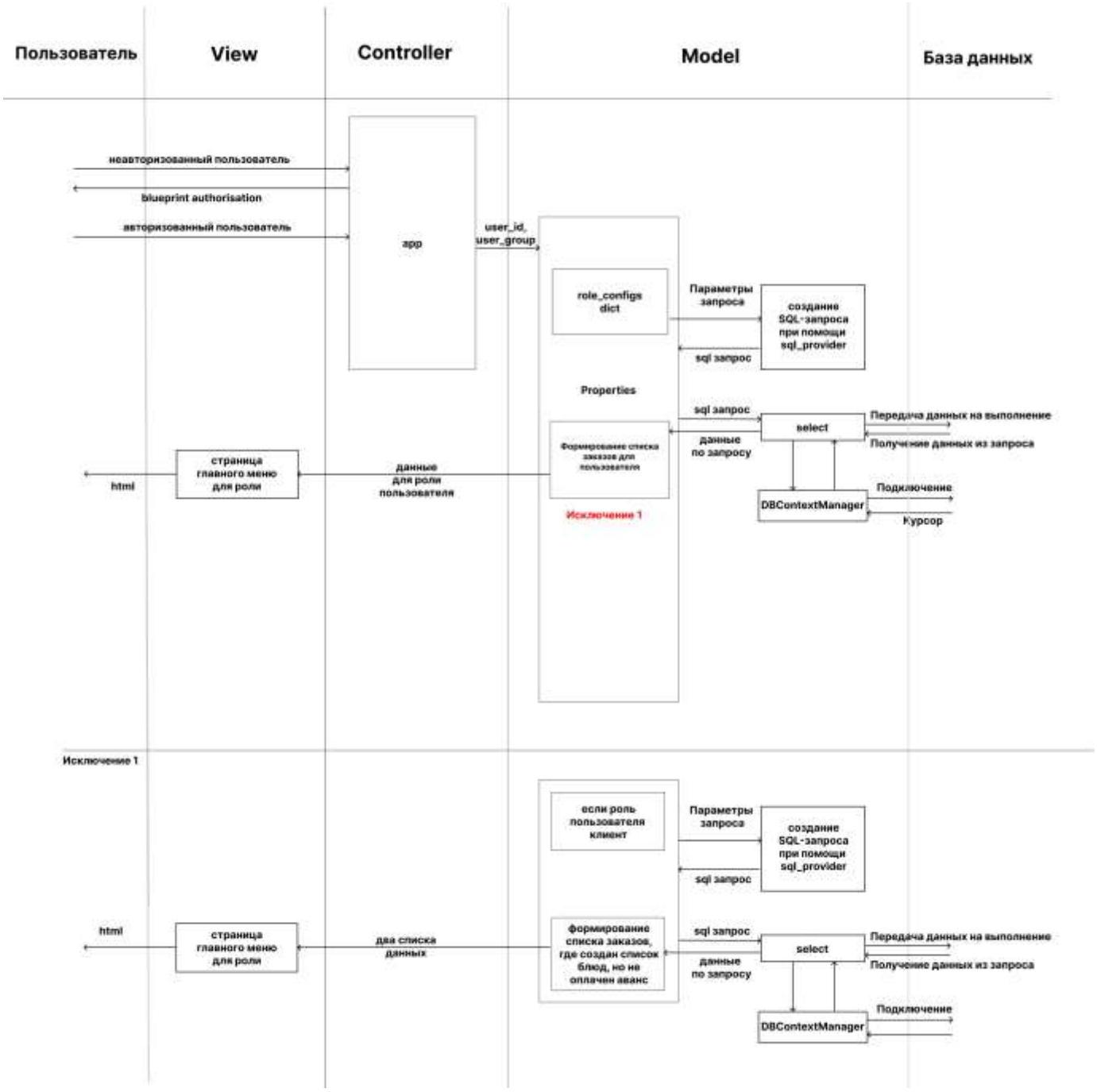
Вариант использования: «Главное меню»

Предусловия: пользователь должен быть авторизован в ИС.

Гарантия: пользователь сможет увидеть интерфейс главного меню, соответствующий его роли.

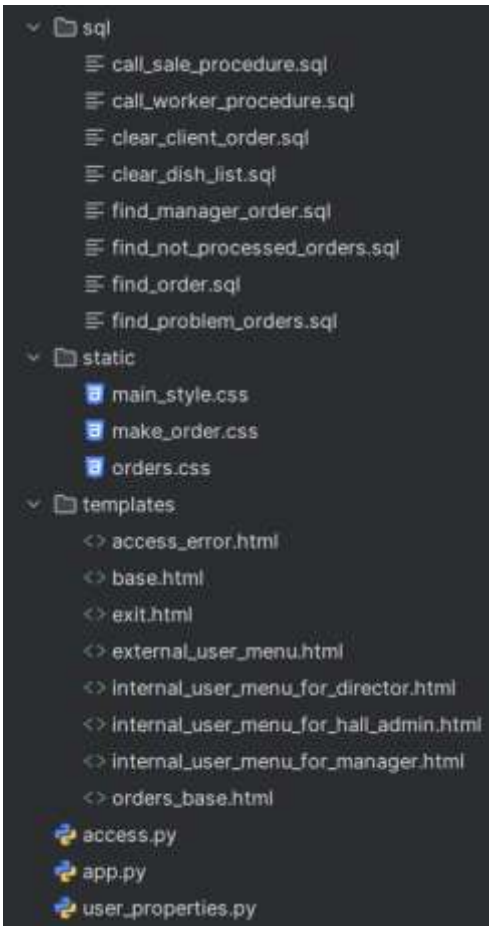
Успешный сценарий работы главного меню:

1. Пользователь начинает сценарий.
2. Система выдает Главное меню в соответствии с ролью пользователя.
3. Для ролей: клиент, официант, администратор залов на главном меню будет также присутствовать раздел с информацией о статусах произведенных заказов (для клиента) и о заказах, которые необходимо будет обслужить, обработать и т.п. (для официанта и администратора залов).



Sequence Diagram для MVC главного меню

Файловая архитектура реализации варианта использования



Требования к шаблонам главного меню пользователей

Шаблон external_user_menu.html (для внешних пользователей)
Назначение: Меню клиента с его заказами
Наследование: Наследуется от orders_base.html
Должен содержать:

1. Навигационное меню:
   - Ссылка "Оформить новый заказ" (bp_order_make.order_make)
2. Выпадающее меню "Работа с запросами":
   - "Посмотреть меню ресторана" (bp_query.form_render)
3. Таблицу заказов с колонками:
   - Номер заказа
   - Статус заказа (с цветовой индикацией)
4. Действия в зависимости от статуса:
   - "Подтвержден":
     ✓ Выбор блюд (bp_dish_list.basket_index)
     ✓ Оплата (bp_payment.payment_form)
     ✓ Текущий счет (bp_query.form_render)
   - "Ждет оплаты":
     ✓ К оплате (bp_payment.payment_form)

        - ✓ Итоговый счет (bp_query.form_render)
    - Кнопка отмены заказа (cancel_order_handler)

Шаблон internal_user_menu_for_manager.html (для менеджеров)
Назначение: Меню менеджера с назначенными заказами
Наследование: Наследуется от orders_base.html
Должен содержать:

1. Навигационное меню с просмотром меню ресторана
2. Таблицу заказов с расширенной информацией:
    - Номер заказа, время, количество человек
    - Номер зала, аванс, количество блюд
    - Стоимость, статус, контакты клиента
3. Действия для заказов со статусом "Полностью оформлен":
    - Правка чека (bp_dish_list.basket_index)
    - Просмотр заказа (bp_query.form_render)
    - Отмена заказа (cancel_order_handler)

Шаблон internal_user_menu_for_director.html (для директора)
Назначение: Меню директора
Наследование: Наследуется от base.html
Должен содержать:

- Ссылку на работу с отчетами (bp_report.start_report)

Шаблон internal_user_menu_for_hall_admin.html (для администратора зала)
Назначение: Меню администратора с необработанными заказами
Наследование: Наследуется от orders_base.html
Должен содержать:

1. Навигационное меню:
    - Работа с отчетами (bp_report.start_report)
    - Выпадающее меню запросов:
        - ✓ Загруженность залов (query_code=1)
        - ✓ Заказы официанта (query_code=2)
        - ✓ Проверка зала (query_code=3)
2. Таблицу необработанных заказов:
    - Основная информация о заказе
    - Кнопка "Обработать" (bp_order_distribute.order_distribute)

Важные моменты взаимодействия с backend:

1. Статусы заказов и их отображение
2. Условная логика отображения действий
3. Передача параметров в URL (order_id, query_code и др.)
4. Обработка render_data['status'] и render_data['data']
5. Различные классы стилей для статусов заказов

Вариант использования: «Работа с запросами»

Пользователи «клиент» и «официант» имеют доступ только к работе с запросами, которые не требуют от пользователя параметров («посмотреть меню», «посмотреть список блюд в заказе»). Их карточка будет аналогичной параметризованным запросам (карточка которых приведена ниже) за исключением ввода определенного параметра.

Только пользователь «администратор залов» имеет доступ к параметризованным запросам. Запросы на просмотр загруженности конкретного зала, просмотр предстоящих заказов для конкретного официанта и проверка на то, загружен ли зал в конкретный день, необходимы для его работы, так как он, в свою очередь является «распределителем» официантов по заказам. Он может отследить какой из залов (по вместимости) является наиболее востребованным, а также в его обязанности может входить равномерное распределение заказов на официантов.

Рассмотрим карточку для параметризованных запросов:

Предусловия: пользователь имеет право на выполнение параметризованного запроса.

Гарантия: пользователь получит данные в соответствии с заданными параметрами, и данные в ИС не должны изменится.

Минимальная гарантия: пользователь получит сообщение, что данных, соответствующих заданным параметрам, нет (или что пользователь не ввел эти параметры).

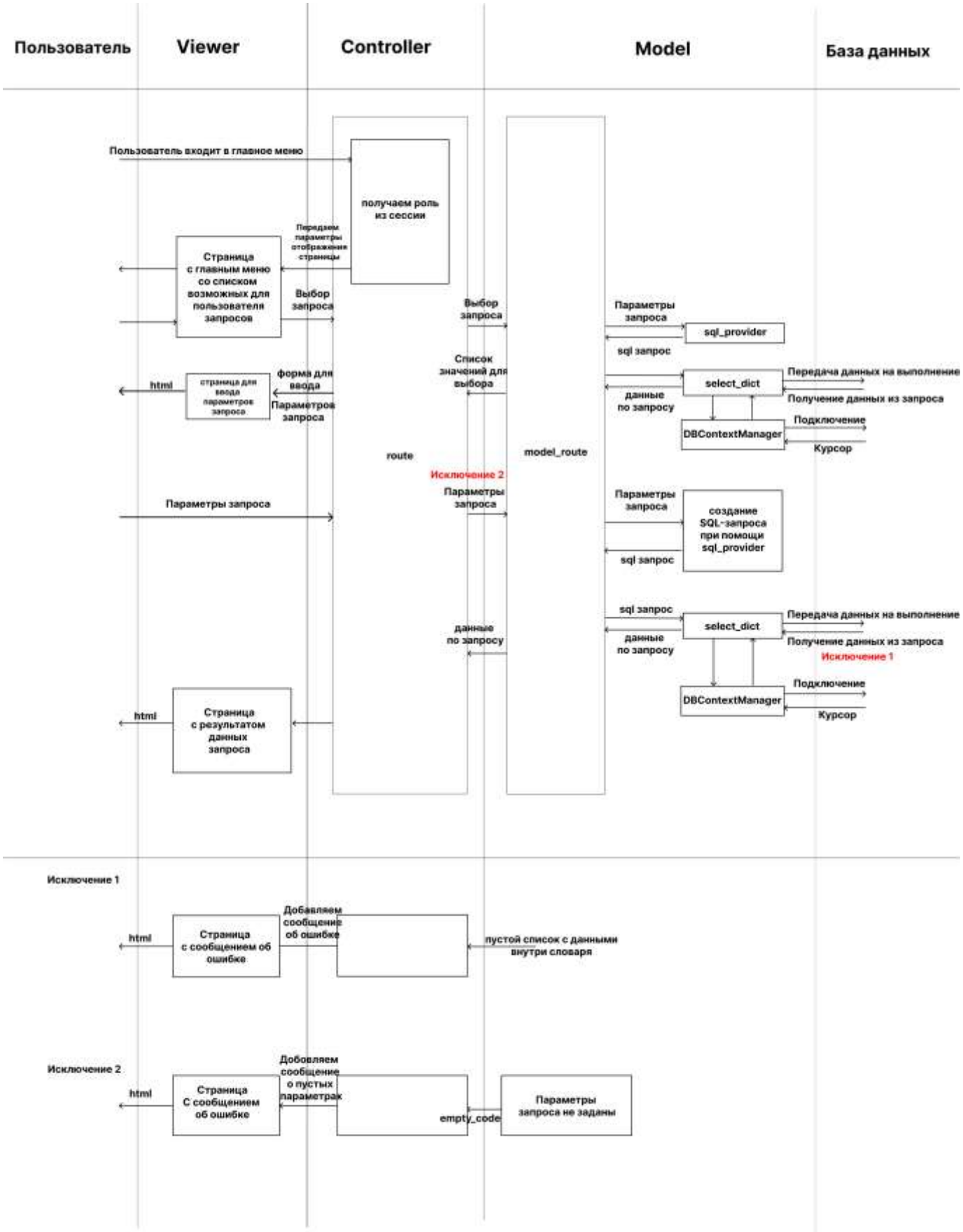
Разберем успешный сценарий для конкретного примера – запрос: «смотреть загруженность зала».

Успешный сценарий работы с запросами:

1. Пользователь переходит к работе с запросом: «смотреть загруженность зала».
2. На экран выводится форма для ввода параметра запроса (номера зала).
3. Пользователь вводит параметр запроса и нажимает «Посмотреть».
4. Система выполняет запрос и показывает результаты запроса; представлена возможность создать еще запрос.

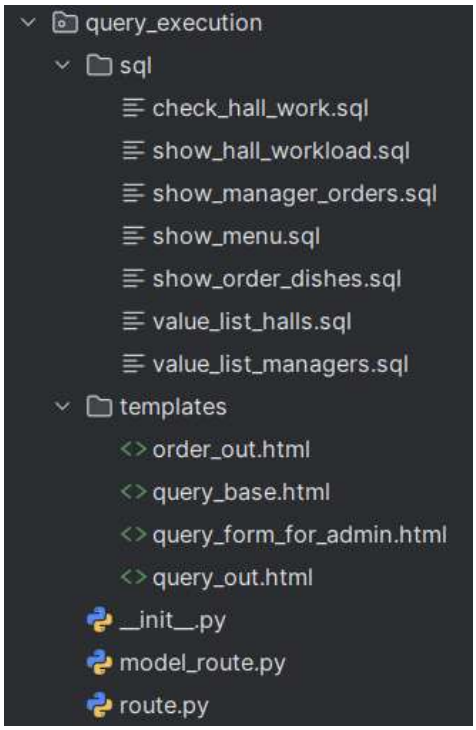
Исключения:

1. Пользователь не добавил параметр запроса. Система выдает ошибку о пустом поле. Представлена возможность создать еще запрос.
2. Система выполняет запрос и присылает пользователю страницу с сообщением о том, что в базе данных не найдена информация, соответствующая введенным параметрам. Представлена возможность создать еще запрос.



Sequence Diagram для MVC работы с запросами

Файловая архитектура реализации варианта использования



Требования к шаблонам blueprint query_execution

Шаблон query_base.html
Назначение: Базовый шаблон для всех страниц работы с запросами.
Наследование: Наследуется от base.html
Должен содержать:

1. Хлебные крошки с текстом "Работа с запросами"
2. Три блока для расширения:
   - query_form_for_admin
   - query_out
   - order_out

Шаблон query_form_for_admin.html
Назначение: Форма для выполнения административных запросов
Наследование: Наследуется от query_base.html
Должен содержать:

1. Форму с методом POST (пустой action)
2. Три варианта отображения в зависимости от query_code:
3. Для query_code == 1:
   - select с именем "hall_choice"
   - Опции формируются из render_data с value от 1 до N
4. Для query_code == 2:
   - select с именем "manager_choice"
   - Опции формируются из render_data со значениями из массива indexes

5. Для query_code == 3:
   - select с именем "hall_choice"
   - Поле типа date с именем "date"
   - Ограничения даты: min_date и max_date
6. Кнопка отправки формы

Шаблон query_out.html
Назначение: Отображение результатов запросов
Наследование: Наследуется от query_base.html
Должен содержать:
- Таблицу с классом orders-table для отображения render_data
- Различные заголовки таблицы в зависимости от status_code и query_code
- Обработку пустых данных с различными сообщениями
- Кнопки для повторного запроса, ведущие на:
  ✓ URL: url_for('bp_query.form_render', query_code=query_code)

4. Шаблон order_out.html
Назначение: Отображение информации о заказах
Наследование: Наследуется от query_base.html
Должен содержать:
- Таблицу с классом orders-table для отображения render_data
- Подсчёт общей суммы заказа (ns.sum_column)
- Обработку случая отсутствия данных

Важные моменты взаимодействия с backend:
Все формы отправляются методом POST на текущий URL
1. Ключевые переменные для обработки:
2. query_code (1, 2, 3) - определяет тип запроса
   - status_code (1, 2) - определяет статус ответа
   - empty_code - определяет наличие данных
   - render_data - содержит данные для отображения
3. URL для навигации: bp_query.form_render с параметром query_code

Варианты использования: «Работа с отчетами»

Отчеты формируются за период в один месяц (для просмотра или создания необходимо указать месяц и год). Существуют два типа отчетов:

Форма отчета «о продажах» (пример: подсчет общей доходности каждого из залов):

- Строка общей отчетности
- ID зала
- Количество заказов
- Общая прибыль заказов

Форма отчета «о сотрудниках» (пример: подсчет зарплат для сотрудников):

- Строка общей отчетности
- ID официанта
- Количество заказов
- Общая прибыль заказов

«Создание отчетов»

Предусловия: пользователь имеет право на составление отчета.

Гарантия: в таблицу отчетности базы данных (соответствующую выбранному типу отчета) будут добавлены строки со значениями, соответствующими выбранному месяцу и году.

Минимальная гарантия: в случае, если такие записи в отчетности есть, или не существует данных, на основе которых они должны быть созданы, пользователь получает сообщение об ошибке и может продолжить работу.

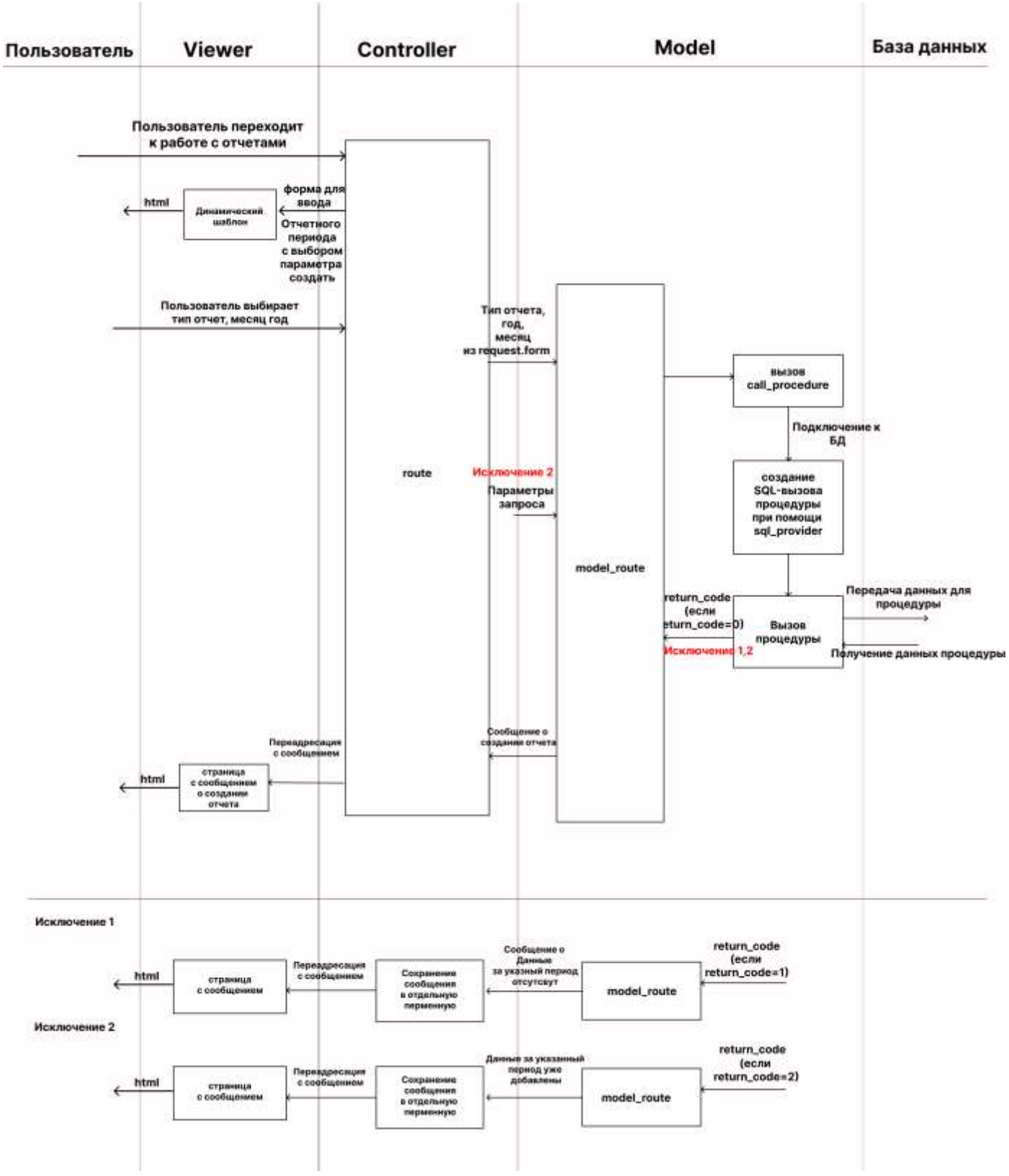
Успешный сценарий работы с отчетами:

1. Пользователь переходит к работе с отчетами.
2. На экран выводится форма для ввода параметров отчета (тип отчета, год, месяц).
3. Пользователь вводит параметры отчета и выбирает «Создать отчет».

4. Система вызывает процедуру, которая добавит новые данные в отчет выбранного типа и за выбранный период, и выведет пользователю сообщение об успешном внесении данных.

Исключения:

1. Система выводит пользователю сообщение о том, что в базе данных не содержатся данные для создания соответствующего отчета (проверка внутри процедуры).
2. Система выводит пользователю сообщение о том, что в базе данных уже есть строки-отчет, соответствующего типа, за выбранные год и месяц (проверка внутри проедуры).



Sequence Diagram для MVC создания отчетов

«Просмотр отчетов»

Предусловия: пользователь имеет право на просмотр отчета.

Гарантия: пользователь получит данные отчетности в соответствии с заданными параметрами.

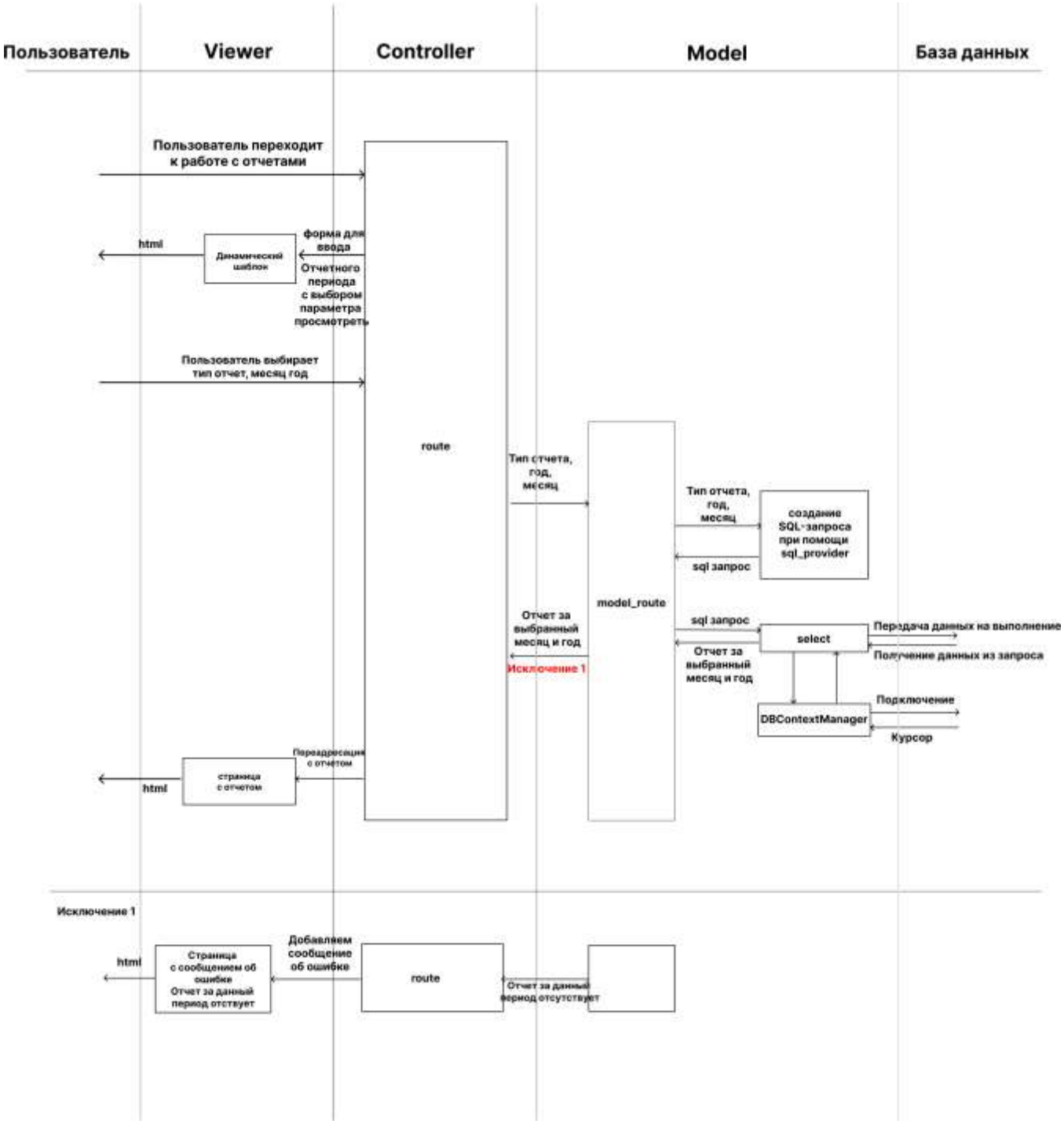
Минимальная гарантия: в случае, если отчет не существует, пользователь получает сообщение об ошибке и может продолжить работу.

Успешный сценарий работы с отчетами:

1. Пользователь переходит к работе с отчетами.
2. На экран выводится форма для ввода параметров отчета (тип отчета, год, месяц).
3. Пользователь вводит параметры отчета и выбирает «Просмотреть отчет».
4. Система выводит отчет выбранного типа за выбранные год и месяц.

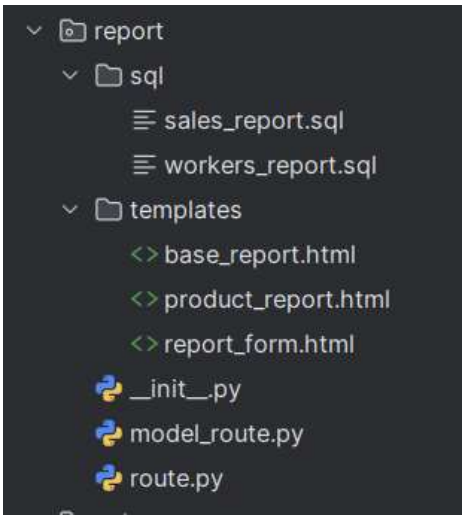
Исключения:

1. Система выводит пользователю сообщение о том, что в базе данных не найден отчет, соответствующему введенным параметрам.



Sequence Diagram для MVC просмотра отчетов

Файловая архитектура реализации варианта использования



Требования к шаблонам blueprint report

Шаблон base_report.html
Назначение: Базовый шаблон для всех страниц, связанных с отчётами
Наследование: Наследуется от base.html
Должен содержать:

1. Хлебные крошки с текстом "Работа с отчетами"
2. Два блока для расширения:
    - report_form - для формы создания отчётов
    - product_report - для отображения отчётов

Шаблон report_form.html
Назначение: Форма для создания и просмотра отчётов
Наследование: Наследуется от base_report.html
Должен содержать:

1. Форму с параметрами:
    - Метод: POST
    - URL: url_for('bp_report.report_handler_result')
2. Поля формы:
    - Выпадающий список "report_choice":
        - ✓ value="1" - Отчёт о продажах
        - ✓ value="2" - Отчёт о сотрудниках
    - Выпадающий список "month_choice":
        - ✓ Значения "01" - "12" для месяцев
    - Числовое поле "year_choice":
        - ✓ Ограничения: min="2020", max="{{ context.year }}"
        - ✓ Значение по умолчанию: context.year
    - Скрытые поля:

- ✓ procedure_name = "public.write_to_product_report"
- ✓ name = "report_card"

3. Кнопки:
    - "create" - только для пользователей с u_group == "hall_admin"
    - "view" - для всех пользователей
4. Обработка ошибок через context.error_message

Шаблон product_report.html
Назначение: Отображение отчётов о продажах и сотрудниках
Наследование: Наследуется от base_report.html
Должен содержать:

1. Навигационную ссылку:
    - URL: url_for('bp_report.start_report')
    - Текст: "Вернуться к отчётам"
2. Заголовок с динамическим содержимым:
    - Для id_rep == 1: "Отчёт о сотрудниках"
    - Для остальных: "Отчёт о продажах каждого зала"
3. Таблицу с классом orders-table (непосредственно отчет)

Важные моменты взаимодействия с backend:

1. Ключевые переменные контекста:
    - context.error_message - сообщения об ошибках
    - context.year - текущий год
    - context.u_group - группа пользователя
    - context.id_rep - тип отчёта
    - context.result - данные отчёта
2. URL для навигации:
    - bp_report.report_handler_result - обработка формы
    - bp_report.start_report - возврат к списку отчётов

Основной бизнес-процесс

Далее будут приведены варианты использования, входящие в основной бизнес-процесс: составление заказа. Для начала рассмотрим возможные статусы заказа в рамках составления заказа

Статусы заказа

1. *В обработке*: данный статус заказ получает после осуществления «резервирования зала».

2. *Подтвержден*: данный статус заказ получает после осуществления «обработки заказа».

3. *Полностью оформлен*: данный статус заказ получает после осуществления «составления списка блюд» и «оплаты».

**Заказ также может быть отменен при необходимости официантом (после того, как его закрепили за данным заказом) или клиентом (до внесения аванса и приобретения статуса «Полностью оформлен»).*

**За отмену заказа отвечает контроллер app.py, который вызывает внешнюю функцию cancel_order, отвечающую за удаление всех данных в рамках конкретного заказа из БД.*

Статус заказа хранится в БД в таблице с информацией о заказах и используется как маркер того, какие действия тот или иной пользователь может выполнять с данным заказом.

Вариант использования: «Резервирование зала»

Предусловия: пользователь имеет право на составление заказа.

Гарантия: пользователь введет необходимую дату и время, а также требуемое количество человек, система зарезервирует под него зал на эту дату и сохранит данные.

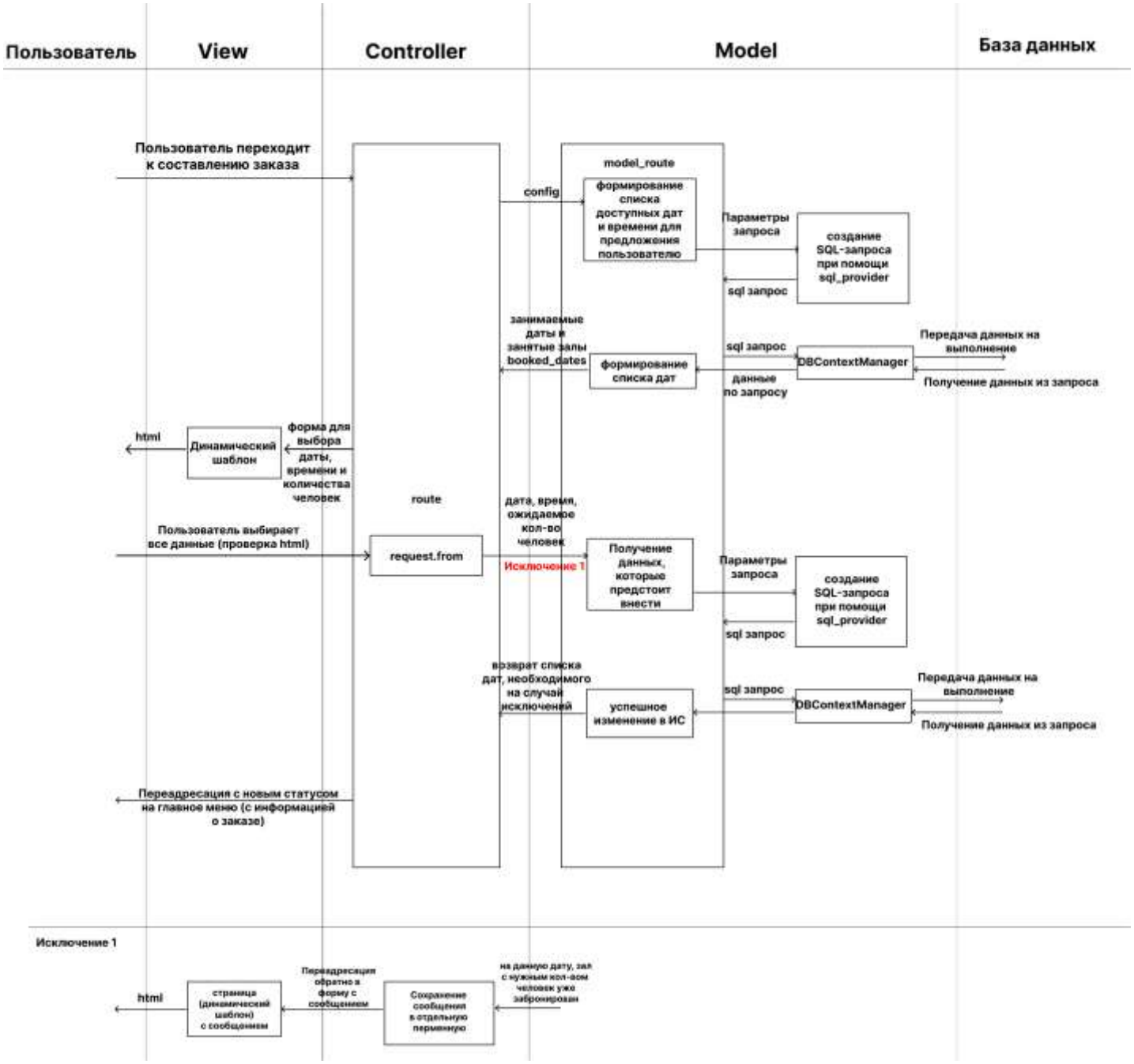
Минимальная гарантия: если на выбранную дату зал с выбранной вместимостью уже занят, пользователь получает сообщение об ошибке и может продолжить работу (зал занят, если в нем планируется провести хотя бы один банкет).

Сценарий работы составления заказов:

1. Пользователь переходит к составлению заказа.
2. Система возвращает форму для ввода значений. Пользователь изначально может выбрать дату только в диапазоне ближайшего месяца.
3. Пользователь вводит данные заказа: выбирает дату и время, а также ориентировочное количество человек.
4. Система возвращает его на главное меню. Заказ получает статус «В обработке», пользователь видит свой заказ и его статус на главном меню.

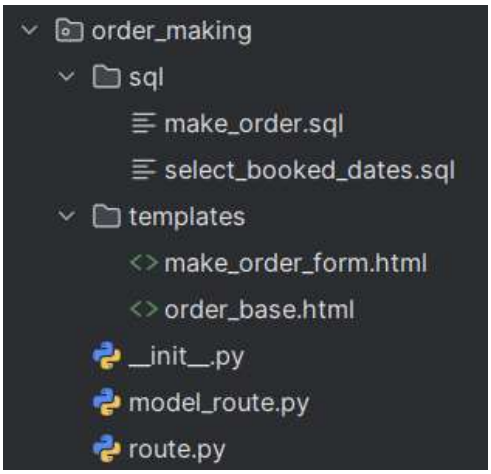
Исключения:

1. Зал, выбранный системой на основе выбранного пользователем ожидаемого количества человек, на дату, выбранную пользователем, уже зарезервирован. Пользователь получает сообщение об этом и должен перевыбрать данные.



Sequence Diagram для MVC резервирования зала

Файловая архитектура реализации варианта использования



Требования к шаблонам blueprint order_making

Шаблон order_base.html
Назначение: Базовый шаблон для страниц оформления заказа
Наследование: Наследуется от base.html
Должен содержать:

- Хлебные крошки с текстом "Составление заказа"
- Блок для расширения make_order_form
- Подключение CSS-файла make_order.css из static

Шаблон make_order_form.html
Назначение: Форма создания нового заказа
Наследование: Наследуется от order_base.html
Должен содержать:

1. Форму с параметрами:
   - Метод: POST
   - URL: url_for('bp_order_make.order_make')
2. Поля формы:
   - Поле даты "date":
     ✓ Тип: date
     ✓ Ограничения: min_date, max_date
     ✓ Обязательное поле
   - Выпадающий список "time" (обязательное поле)
   - Выпадающий список "place_amount"
   - Поле телефона "phone" (типа tel: паттерн: ^\+7\d{10}$)
3. Обработка ошибок через переменную error

Важные моменты взаимодействия с backend:

1. Валидация формы:

- Все поля кроме place_amount обязательны
- Телефон должен соответствовать формату +7 и 10 цифр

2. Передаваемые параметры:
   - min_date, max_date - ограничения для выбора даты
   - time_options - список доступных временных слотов
   - error - сообщения об ошибках
3. URL для отправки формы: bp_order_make.order_make

Вариант использования: «Обработка заказов»

Предусловия: пользователь имеет право на обработку заказов; заказы, которые он видит должны иметь статус «В обработке».

Гарантия: пользователь зарезервирует заказ за свободным, на эту дату, менеджером.

Минимальная гарантия: если на дату заказа нет свободных менеджеров, пользователь получает сообщение об ошибке и может продолжить работу.

Успешный сценарий работы обработки заказов:

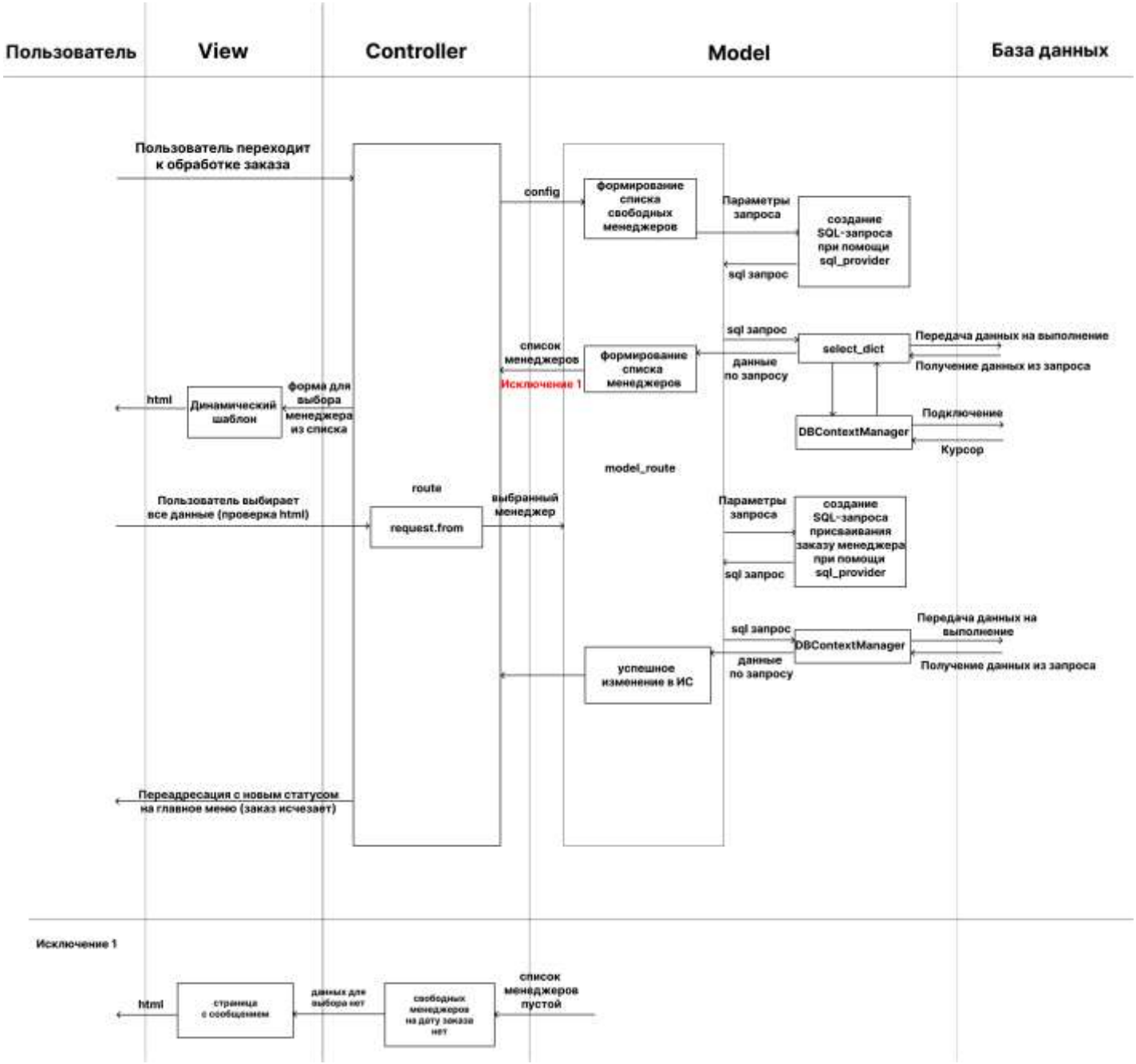
1. Пользователь авторизуется в системе и на главном меню видит заказы «В обработке» (он может обработать каждый из них отдельно, по кнопке). Он переходит, чтобы обработать один из них.
2. На экран выводится форма со свободными менеджерами на дату обрабатываемого заказа, а также телефон клиента этого заказа для обратной связи (при необходимости).
3. Пользователь выбирает менеджера.
4. Система возвращает его на главное меню. Заказ обработан и пропадает из списка заказов «на обработку» на главном меню. Заказ получает статус «Подтвержден».

Исключения:

1. На дату заказа нет свободных менеджеров, система информирует об этом пользователя, у него есть кнопка, по которой он может вернуться в главное меню.

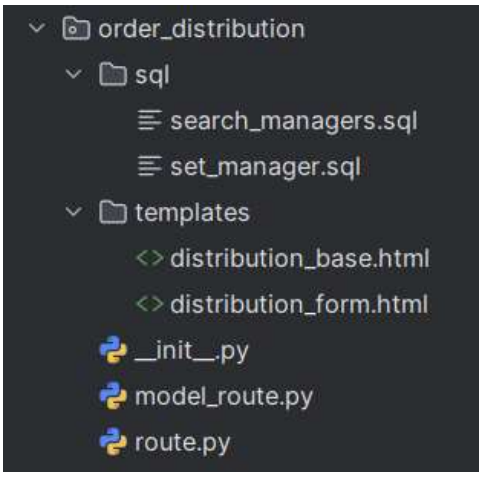
Исключение, которого нет на диаграмме:

1. Заказов «В обработке» нет, система информирует пользователя об этом на главном меню.



Sequence Diagram для MVC обработки заказов

Файловая архитектура реализации варианта использования



Требования к шаблонам blueprint order_distribution

Шаблон distribution_base.html
Назначение: Базовый шаблон для страниц распределения заказов
Наследование: Наследуется от base.html
Должен содержать:

- Хлебные крошки с текстом "Распределение заказов"
- Блок для расширения distribution_form

Шаблон distribution_form.html
Назначение: Форма распределения заказов между менеджерами
Наследование: Наследуется от distribution_base.html
Должен содержать:

1. Информационный блок:
   - Отображение телефона клиента для связи
2. Таблицу со списком менеджеров (orders-table) – если данные есть
3. Для каждого менеджера форма с параметрами:
   - Метод: POST
   - URL: url_for('bp_order_distribute.order_distribute')
   - Скрытые поля:
     - ✓ manager_id - ID менеджера
     - ✓ order_id - ID заказа
4. Кнопка "Выбрать" для назначения заказа
5. Сообщение об ошибке "На данный момент все работающие менеджеры заняты" – если данных для таблицы нет

Важные моменты взаимодействия с backend:

1. Входные параметры:
   - phone - телефон клиента для отображения

- id - идентификатор заказа
- render_data:
- status - наличие свободных менеджеров
- data - список кортежей (manager_id, manager_name)

2. URL для отправки формы: bp_order_distribute.order_distribute
3. Передача данных через скрытые поля формы:
   - manager_id
   - order_id

Вариант использования: «Составление списка блюд»

– фактически, корзина.

Предусловия: пользователь имеет соответствующую роль; его заказ приобрел статус «Подтвержден».

Гарантия: пользователь составит список блюд: выберет необходимые позиции из меню, система сохранит данные.

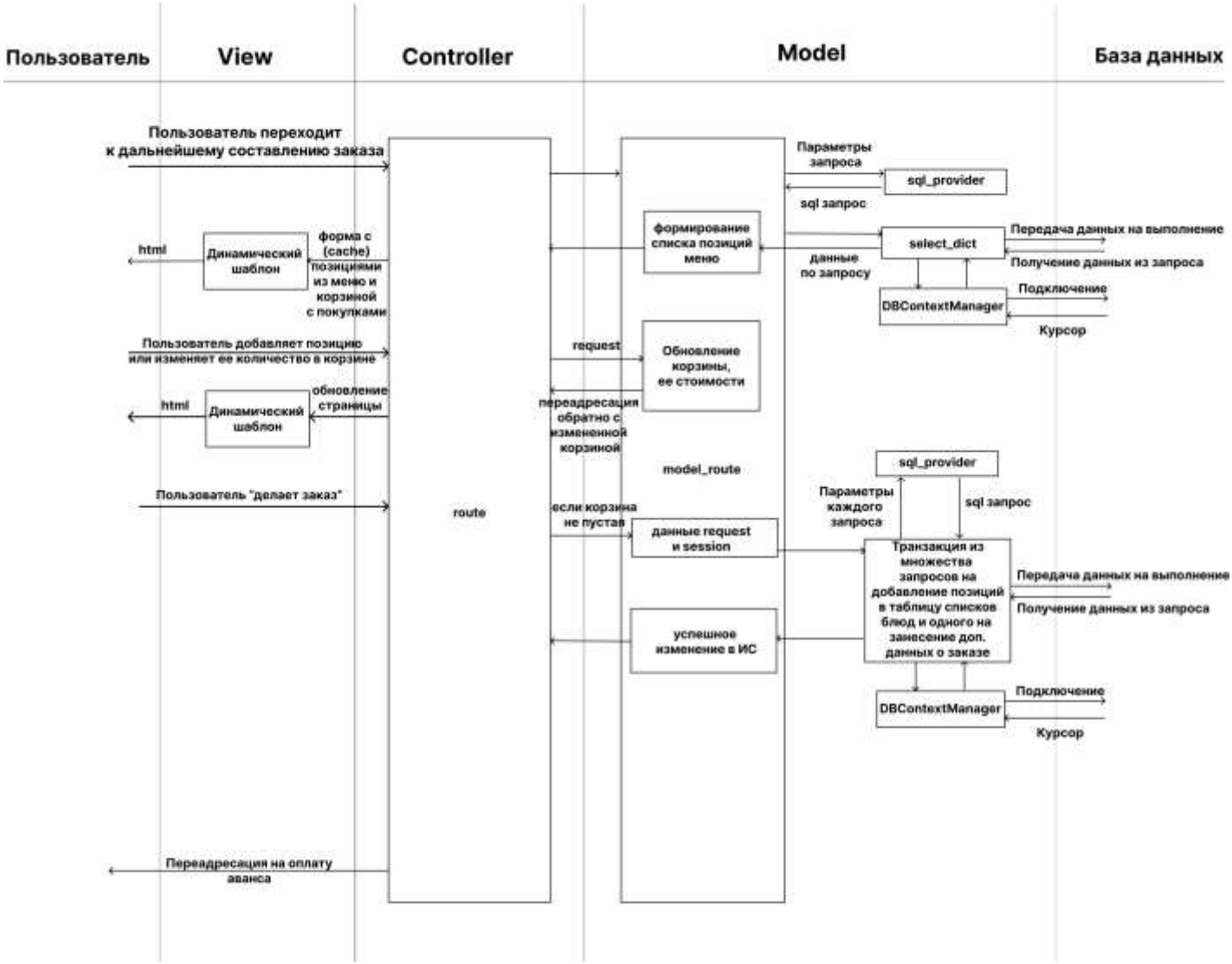
Минимальная гарантия: пользователь нажал «сделать заказа» с пустой корзиной, система оставляет его на той же странице, он может продолжить работу.

Успешный сценарий работы:

1. Пользователь переходит к выбору позиций из меню.
2. Система возвращает шаблон с позициями меню и корзиной. Пользователь может добавлять товары из меню в корзину по кнопке «купить», а также увеличивать или уменьшать их количества по счетчикам в корзине.
3. Пользователь составляет корзину из выбранных позиций и выбирает «сделать заказ».
4. Система возвращает форму **«Оплата»** для внесения аванса.

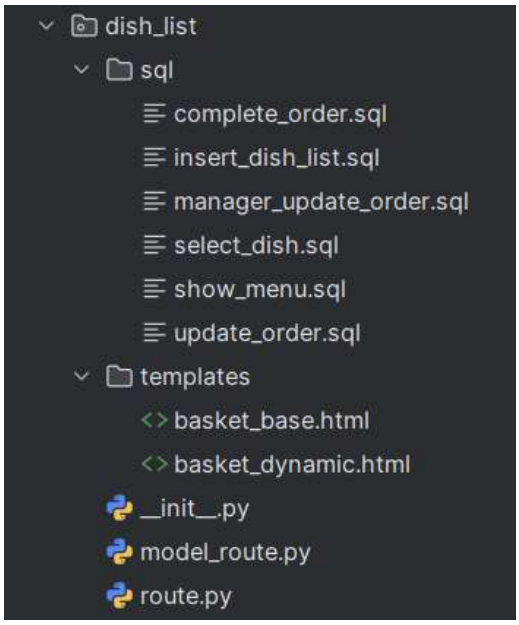
Исключение, которого нет на диаграмме:

1. Заказов со статусом «Подтвержден» нет, пользователь видит это на главном меню.
2. В случае если пользователь выбирает «сделать заказ» с пустой корзиной – ничего не происходит, он остается на той же странице.



Sequence Diagram для MVC составления списка блюд

Файловая архитектура реализации варианта использования



Требования к шаблонам blueprint dish_list

Шаблон basket_base.html
Назначение: Базовый шаблон для страниц работы с корзиной блюд
Наследование: Наследуется от base.html
Должен содержать:

- Хлебные крошки с текстом "Выбор блюд"
- Блок для расширения basket_dynamic
- Подключение внешних ресурсов:
- Bootstrap 5.2.3 CSS и JS
- make_order.css из static

Шаблон basket_dynamic.html
Назначение: Страница выбора блюд и работы с корзиной
Наследование: Наследуется от basket_base.html
Должен содержать:

1. Макрос render_item для отображения блюд с параметрами:
   - item - информация о блюде
   - show_amount - показывать ли количество
   - show_form - показывать ли форму добавления
2. Кнопки действий в зависимости от роли:
   - Для manager:
     - ✓ URL: url_for('bp_dish_list.complete_order') - "Доплат производить не нужно"

- ✓ URL: url_for('bp_dish_list.save_order', role=role) - "Добавить позиции для доплаты"
- Для остальных:
  - ✓ URL: url_for('bp_dish_list.save_order', role=role) - Сделать заказ

3. Основной контент в двухколоночном layout:
   - Левая колонка (8 колонок) с отображением меню ресторана
   - Правая колонка (4 колонки) с отображением корзины
   - Кнопка очистки корзины (URL: url_for('bp_dish_list.clear_basket'))

Важные моменты взаимодействия с backend:

1. Формы отправляются методом POST на текущий URL (главное меню) – для официанта и на «Оплату» – для клиента
2. Ключевые переменные:
   - role - роль пользователя
   - dishes - список доступных блюд
   - basket - содержимое корзины
   - basket_price - общая стоимость
   - message - сообщения об ошибках
3. Идентификация блюд через поле dish_display со значением iddish

Вариант использования: «Оплата»

Предусловия: пользователь имеет на это право; (1) заказ, привязанный к нему, имеет статус «Ждет оплаты»; дата банкета прошла; он знает реальную стоимость заказа (ее отличие от внесенного аванса) ИЛИ (2) пользователь составил заказ и перешел для оплаты аванса.

Гарантия: пользователь сможет совершить оплату.

Минимальная гарантия: баланса пользователя не хватает для оплаты, пользователь получает сообщение об ошибке и может продолжить работу.

Успешный сценарий работы:

1. Пользователь переходит к оплате.
2. На экран выводится его баланс и необходимая цена к оплате.
3. Пользователь нажимает «оплатить».
4. Система снижает его баланс на оплаченную сумму.

4.1 Для случая (1): заказ приобретает статус «Завершен». Бизнес-процесс «завершение заказа» окончен. Пользователь перенаправлен на главное меню и видит это.

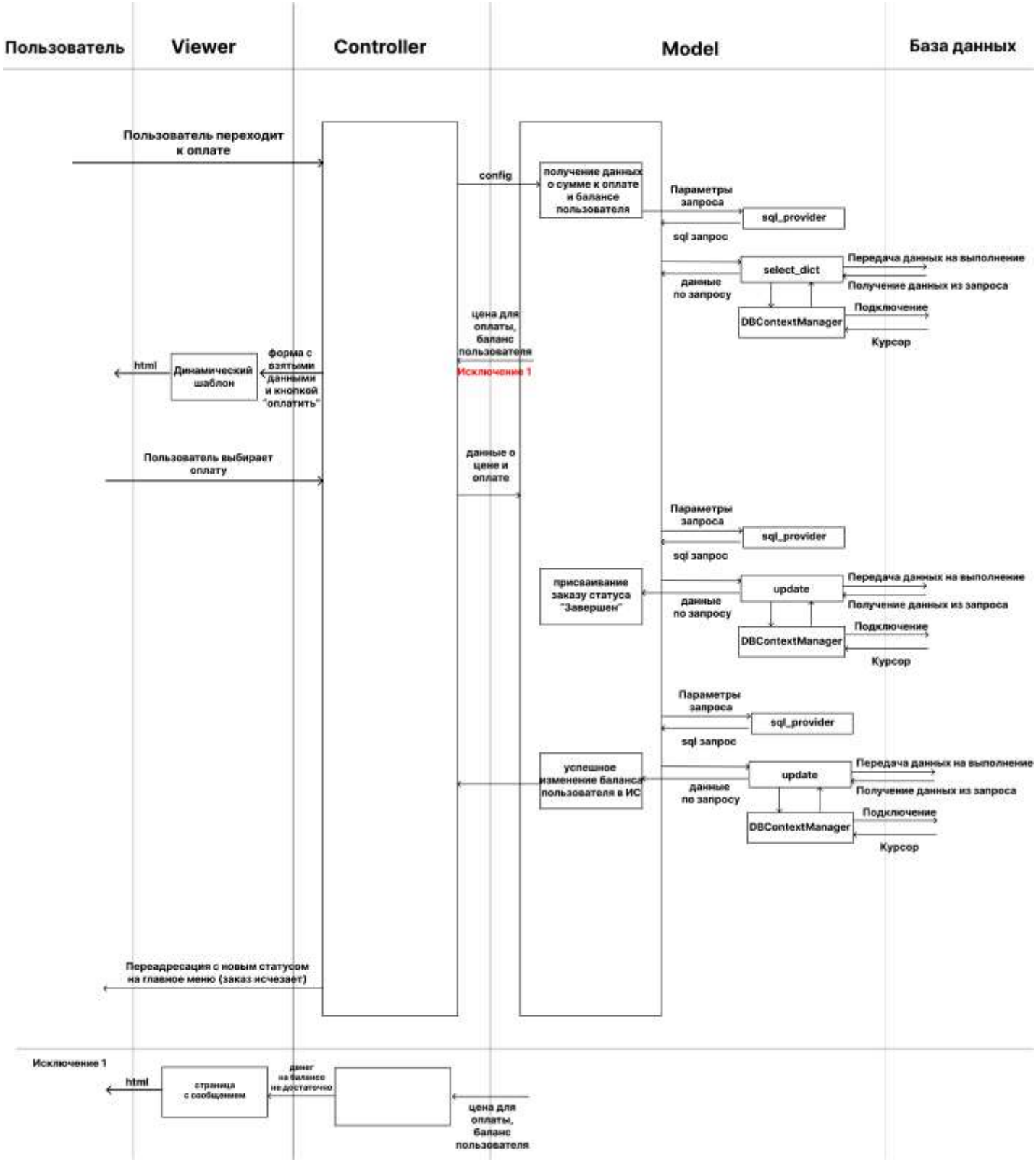
4.2 Для случая (2): система возвращает его на главное меню (а также вносит аванс в таблицу заказа). Заказ приобретает статус «Полностью оформлен». Бизнес-процесс «составление заказа» окончен.

Исключения:

1. Баланс пользователя ниже цены к оплате. Пользователь получает сообщение об этом на экран.

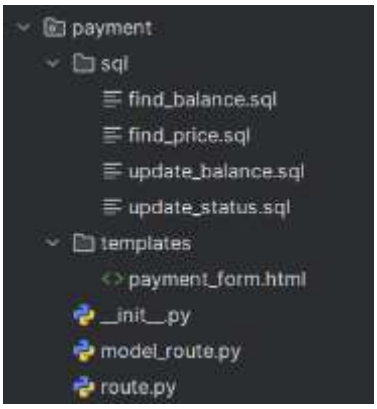
Исключение, которого нет на диаграмме:

1. Для случая (1): заказов со статусом «Ждет оплаты» нет, пользователь видит это на главном меню.
2. Пользователь покинул «Оплату», не совершив ее. В данном случае статус заказа не изменится, однако список выбранных им блюд сохранится в системе. Пользователь вместо выбора блюд, сможет сразу оплатить заказ или отменить его.



Sequence Diagram для MVC оплаты

Файловая архитектура реализации варианта использования



Требования к шаблону payment_form.html из blueprint payment

Назначение: Форма оплаты заказа
Наследование: Наследуется от base.html
Должен содержать:

1. Хлебные крошки:
   - Текст "Оплата"
2. Форма оплаты:
   - Метод: POST
   - Action: пустой (отправка на текущий URL)
   - Содержимое:
     - ✓ Заголовок с номером заказа (order_id)
     - ✓ Информация о балансе пользователя (balance)
     - ✓ Сумма к оплате (price)
     - ✓ Кнопка "Оплатить"
     - ✓ Отображается только если нет сообщения об ошибке
3. Обработка ошибок:
   - Блок для отображения сообщения об ошибке (message)
   - Класс error-message для стилизации
   -

Важные моменты взаимодействия с backend:

1. Входные параметры:
   - order_id - номер заказа
   - balance - текущий баланс пользователя
   - price - сумма к оплате
   - message - сообщение об ошибке (если есть)
2. Форма отправляется на текущий URL (главное меню)
3. Действие определяется наличием параметра buy в POST-запросе

Особенности:

- Кнопка оплаты скрывается при наличии сообщения об ошибке
- Все суммы отображаются в рублях
- Использует кастомные стили для кнопок и сообщений об ошибках

Дополнительный бизнес-процесс

Далее будут приведены варианты использования, входящие в дополнительный бизнес-процесс: завершение заказа. Для начала рассмотрим возможные статусы заказа в рамках завершения заказа

Статусы заказа

1. *Ждет оплаты*: данный статус заказ получается после осуществления «редактирования чека заказа».

2. *Завершен*: данный статус заказ получается после осуществления второй «оплаты» или сразу после «редактирования чека заказа» в случае если за время самого банкета новых позиций не было выбрано.

Карточка варианта использования «Оплата» была расписана выше.

Вариант использования: «Редактирование чека заказа»

– фактически, корзина уже для официанта.

Предусловия: пользователь имеет на это право; заказ, привязанный к нему, имеет статус «Полностью оформлен»; дата банкета прошла; он знает реальную стоимость заказа (ее отличие от внесенного аванса).

Гарантия: пользователь сможет добавить реальную стоимость заказа.

Минимальная гарантия: пользователь выберет завершить заказ без доплат. Заказ примет статус «Завершен». Или пользователь нажал «сделать заказа» с пустой корзиной, система оставляет его на той же странице, он может продолжить работу.

Успешный сценарий работы:

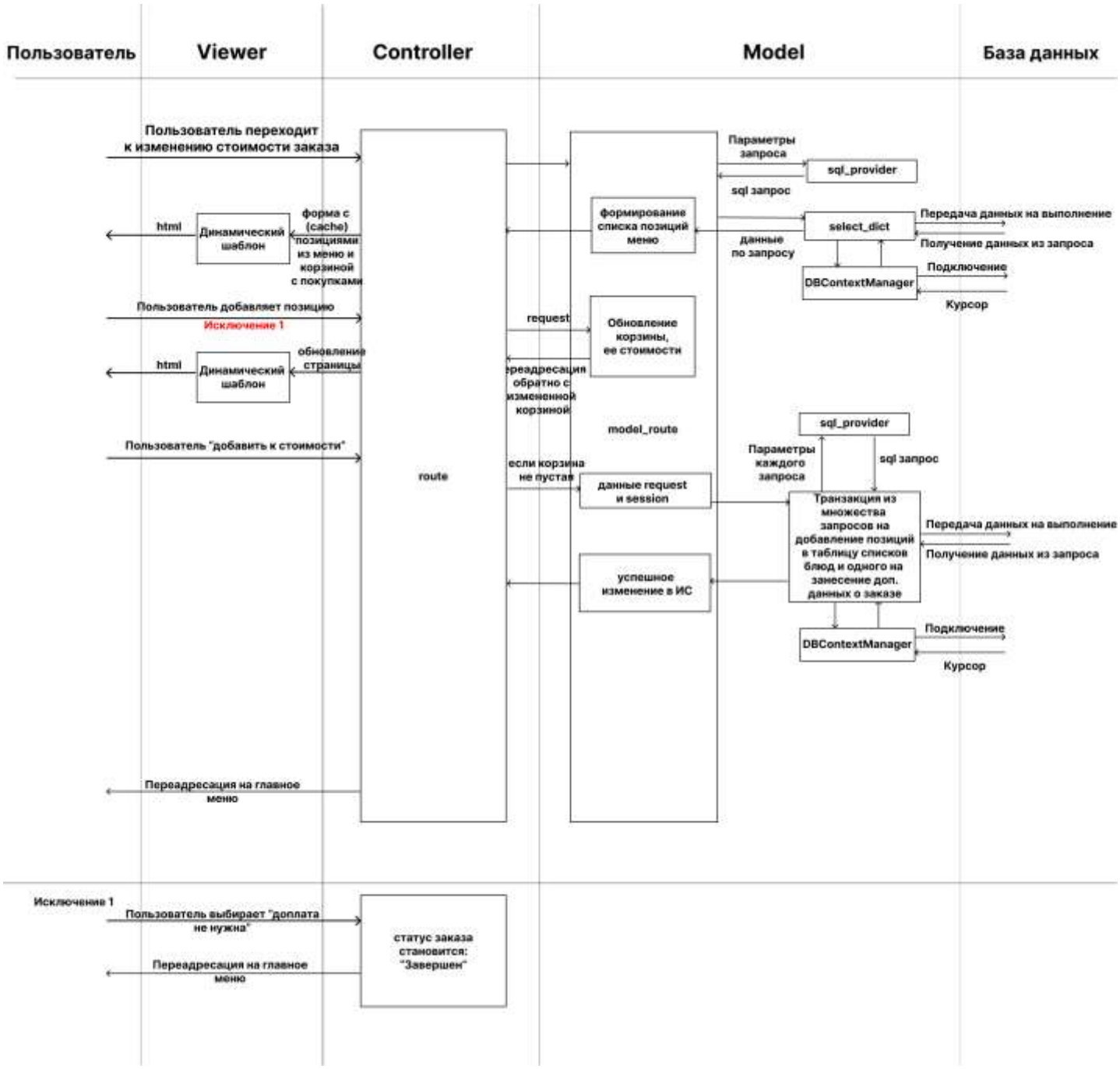
1. Пользователь авторизуется в системе и на главном меню видит заказ «Полностью оформлен». День этого заказа уже прошел. Пользователь переходит для ввода реальной стоимости.
2. На экран выводится тот же шаблон корзины, но добавляется кнопка, по которой он может без добавления товаров завершить заказ (см. исключение).
3. Пользователь выбирает позиции, сравнивает их сумму с чеком и в корзину.
4. После добавления всех позиций он наживает кнопку «добавить к заказу».
5. Система возвращает его на главное меню. Заказ приобретает статус «Ждет окончательной оплаты».

Исключения:

1. Ничего кроме аванса докуплено клиентами не было, у пользователя есть вариант оставить аванс окончательной суммой по кнопке. Заказ приобретает статус «Завершен». Бизнес-процесс окончен.

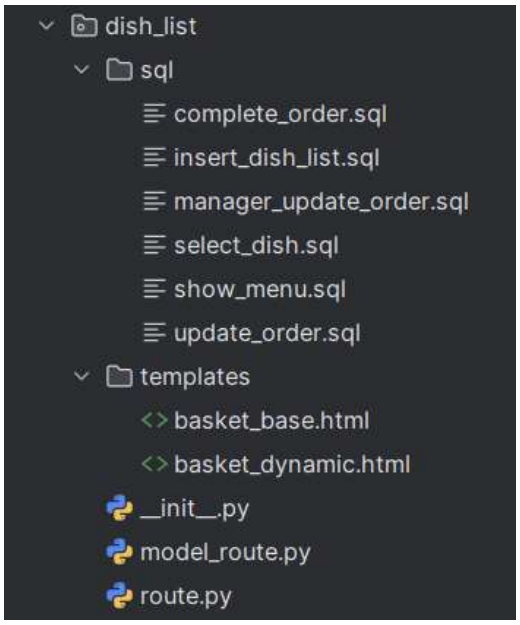
Исключение, которого нет на диаграмме:

1. Заказов со статусом «Полностью оформлен» нет, пользователь видит это на главном меню.
2. В случае если пользователь выбирает «сделать заказ» с пустой корзиной – ничего не происходит, он остается на той же странице.



Sequence Diagram для MVC редактирования чека заказа

Файловая архитектура реализации варианта использования



Требования к шаблонам blueprint dish_list уже описаны в варианте использования «Составление списка блюд».

Тестовые данные

Тестовые данные для «Авторизации»:

user_group	login	password	result
manager	manager2	manager2_pas	вход в систему
manager	manager3	manager3_pas	вход в систему
manager	manager5	manager5_pas	вход в систему
manager	manager6	manager6_pas	вход в систему
director	director1	director1_pas	вход в систему
hall_admin	ha1	ha1_pas	вход в систему
client	client1	client1_pas	вход в систему
client	client2	client2_pas	вход в систему
manager	abc	123	сообщение об ошибке
client	abc	123	сообщение об ошибке

Тестовые данные для «Работы с запросами» (параметризованными):

1. Просмотр загруженности залов

выбор зала	ответ
4	{3 даты}
3	исключение

2. Просмотр заказов официантов

выбор официанта	ответ
Иванов Федор	{заказы на официанта}

3. Проверка зала

выбор зала	выбор даты	ответ
4	2024-12-28	зал занят, {время}
3	2024-12-28	зал свободен (исключение)

Тестовые данные для «Работы с отчетами»:

1. Создание:

   • Отчет о продажах

   Для успешного сценария: декабрь, 2024

   Для исключения: декабрь, 2024 (после создания)

   • Отчет о сотрудниках

   Для успешного сценария: декабрь, 2024

   Для исключения: декабрь, 2024 (после создания)

2. Просмотр:

   • Отчет о продажах

   Для успешного сценария: сентябрь, 2020

   Для исключения: январь, 2021

   • Отчет о сотрудниках

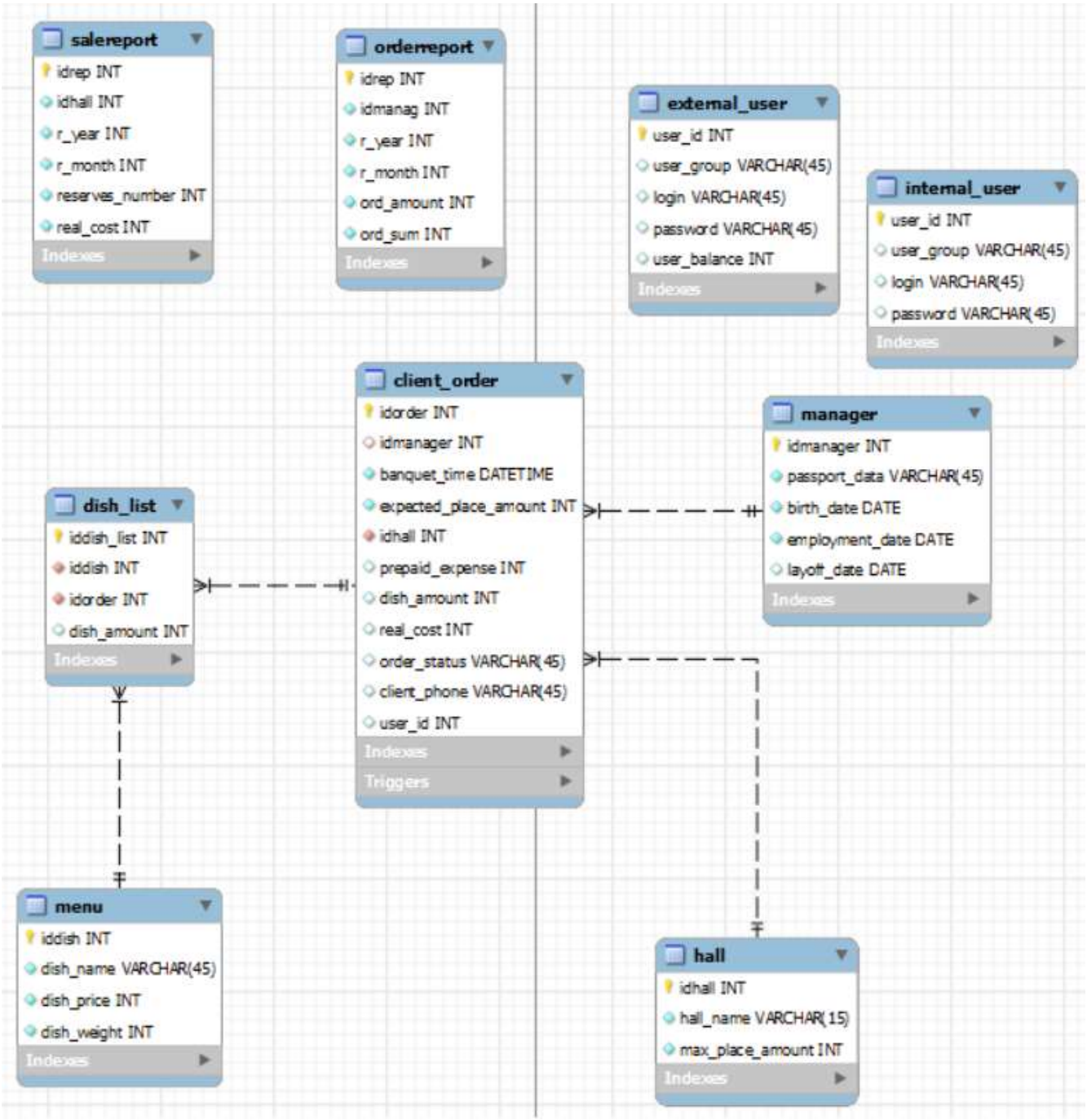
   Для успешного сценария: апрель, 2024

   Для исключения: сентябрь, 2023

Тестовые данные для «Резервирования заказа» (телефон клиента и выбор времени не влияют на результат):

выбор даты	посадка	result
2024-12-31	до 50 человек	изменение заказа в ИС
2024-12-28	до 50 человек	сообщение об ошибке

Инфологическая модель базы данных



Заключение

В ходе выполнения работы была создана информационная система «Банкетный зал». Для работы с системой разработан веб-интерфейс на языке программирования Python с использованием фреймворка Flask. В процессе разработки выполнено предварительное проектирование, составлена вся необходимая документация и проработана архитектура приложения.

Система охватывает ключевые аспекты бизнес-процесса, включая автоматизацию процесса заказа банкета и обслуживания клиентов. В приложении реализована система авторизации с разграничением прав доступа, что обеспечивает безопасную и эффективную работу с данными. Среди функциональных возможностей системы были разработаны следующие варианты использования: выбор варианта использования в главном меню, авторизация пользователя, работа с запросами, работа с отчетами, полный процесс заказа банкета, его обслуживание и оплата.

Созданное приложение предоставляет конечным пользователям удобный веб-ориентированный интерфейс для взаимодействия с базой данных, упрощая процессы тестирования и анализа. В процессе выполнения курсовой работы приобрел навыки программирования на Flask, проектирования веб-приложений и подготовки проектной документации. Полученный опыт может быть применен в будущей профессиональной деятельности для разработки аналогичных систем.