ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Программная инженерия»

Домашняя работа №5 по дисциплине “"Проектирование архитектуры программных систем”

Интеграция и внешние коммуникации системы

Выполнили студенты группы БПИ223
Абдуллаев А.Ш.
Жалилов А.
Курманова А.

Москва 2025

ОГЛАВЛЕНИЕ

Веб-приложение «SmartTable»

«SmartTable» — это веб-приложение, предназначенное для автоматизации процесса обслуживания клиентов в ресторанах и кафе. Платформа включает в себя два компонента: административную панель для ресторанов и Telegram MiniApp для посетителей. Приложение упрощает процесс заказа и оплаты, позволяет клиентам просматривать меню, делать заказы, вызывать официанта и делить счет с компанией. Для ресторанов система предоставляет удобные инструменты для управления меню, заказами и сотрудниками, а также позволяет отслеживать статус заказов для упрощения взаимодействия с клиентами.

Диаграмма 4К Уровень

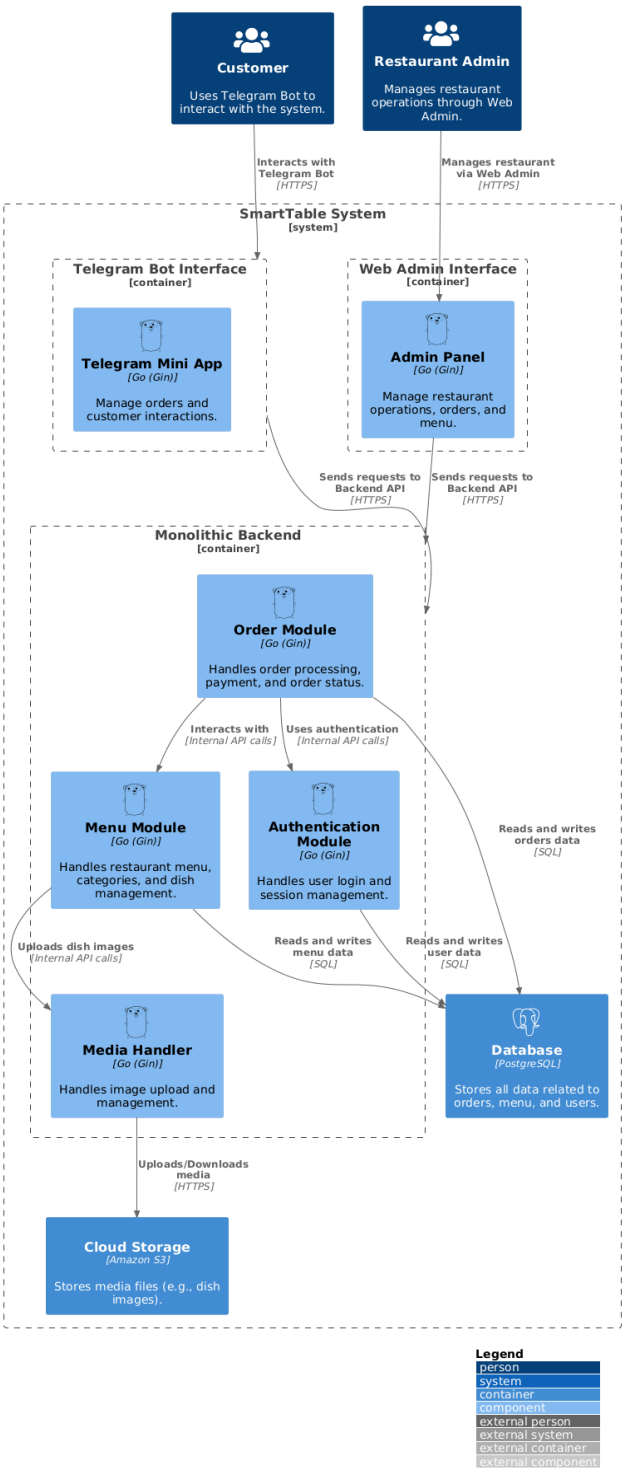
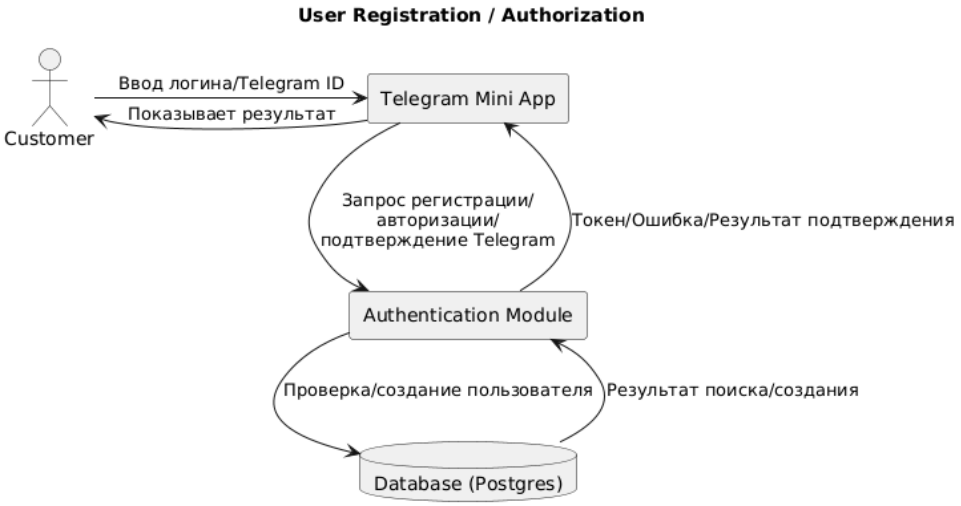
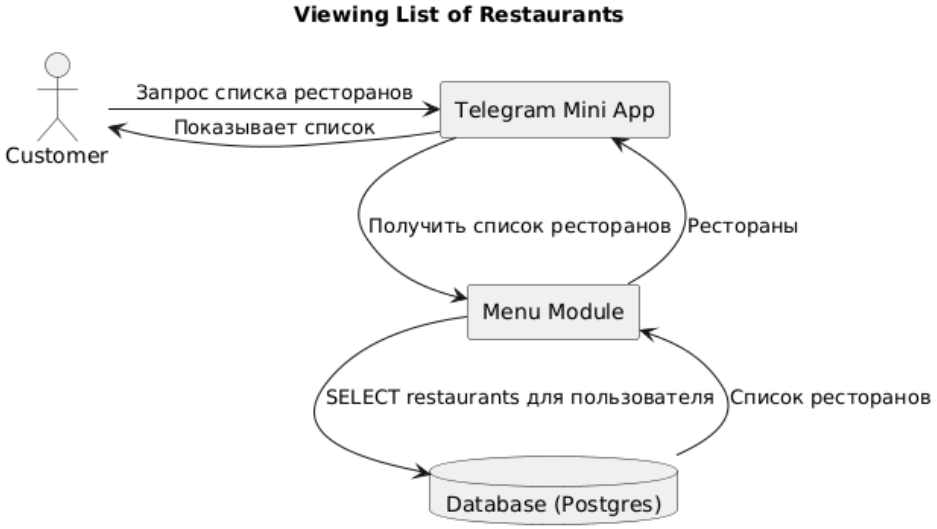


Диаграмма потоков данных

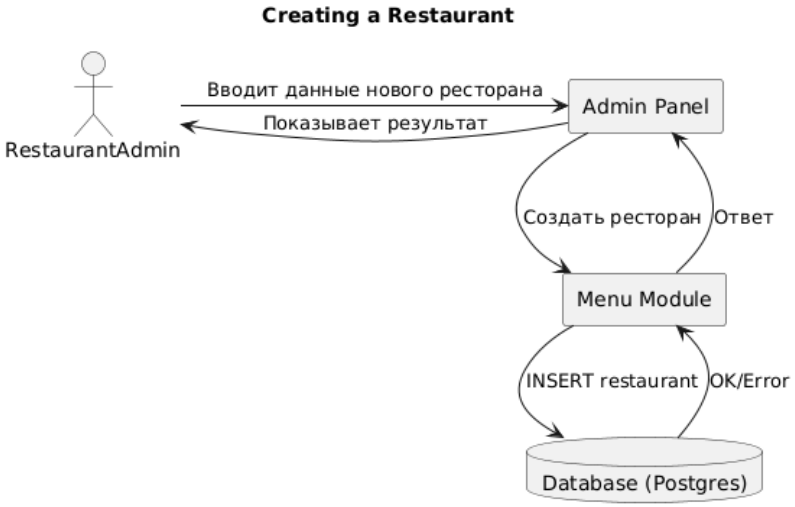
1. Регистрация/Авторизация пользователя и подтверждение логина через Telegram



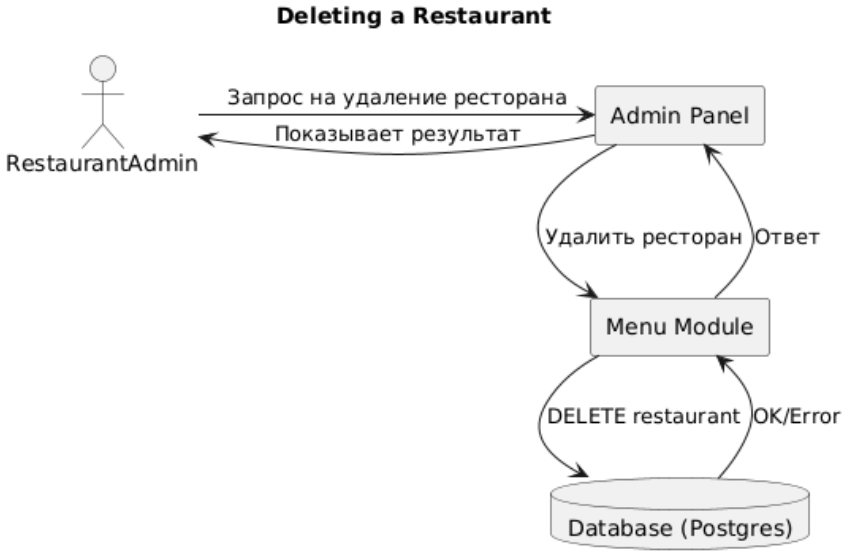
2. Отображение списка ресторанов пользователя



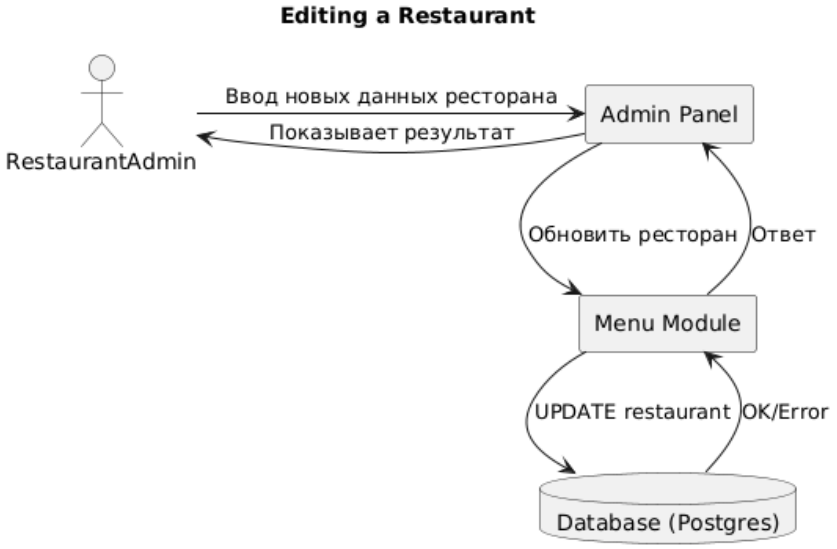
3. Создание ресторана



4. Удаление ресторана

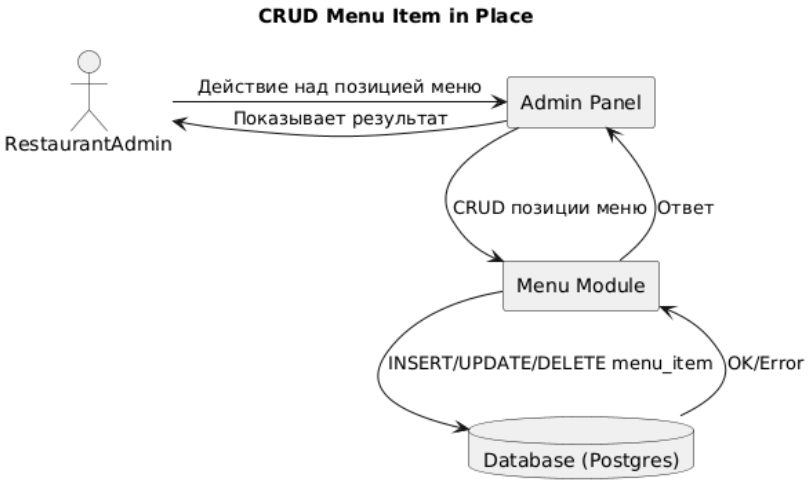


5. Редактирование ресторана

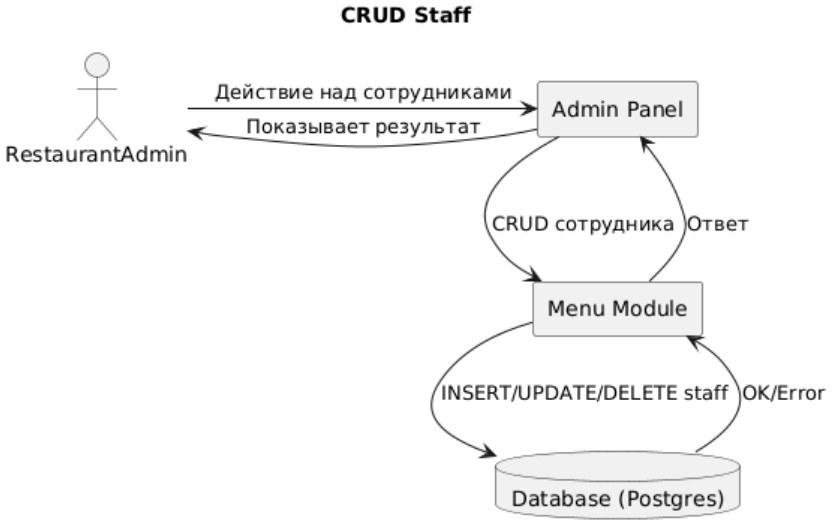


6. CRUD Плейса ресторана

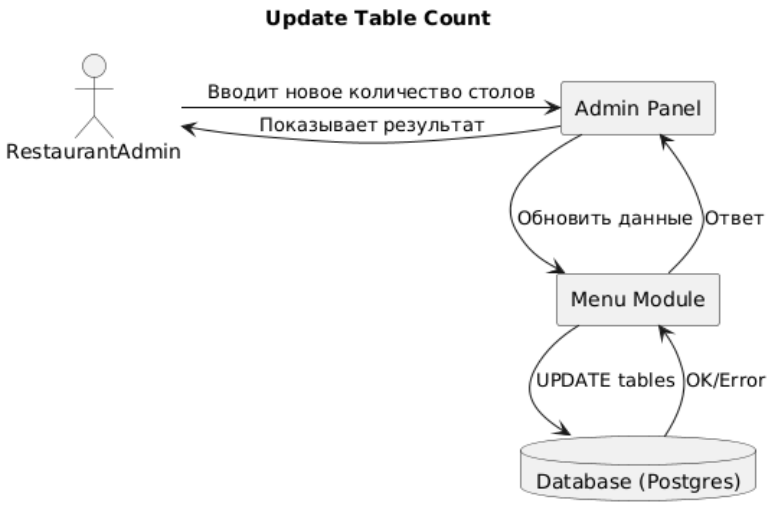
7. CRUD позиции меню плейса



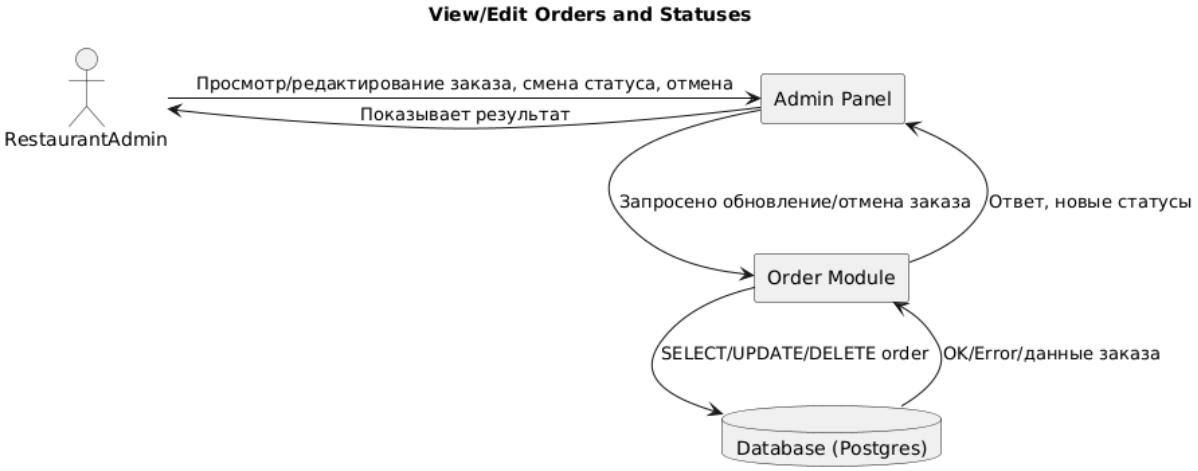
8. CRUD состава сотрудников



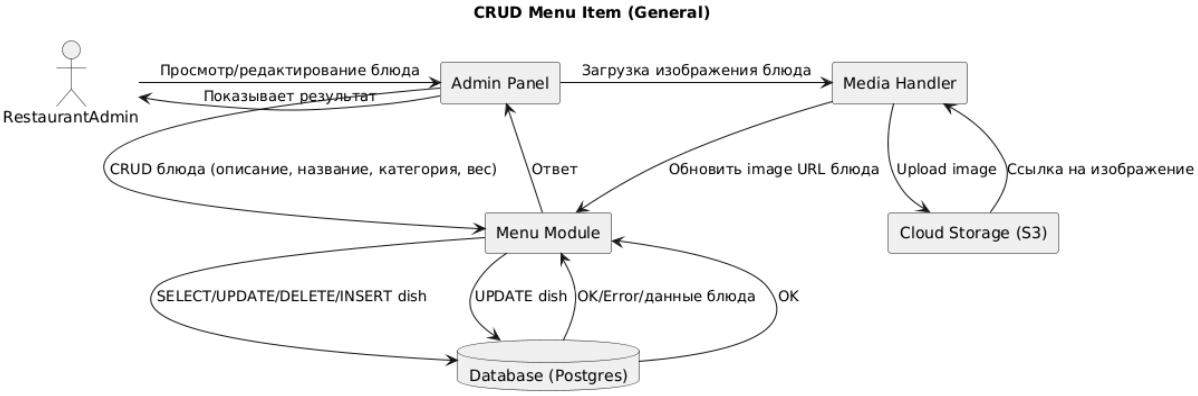
9. Редактирование количества столов



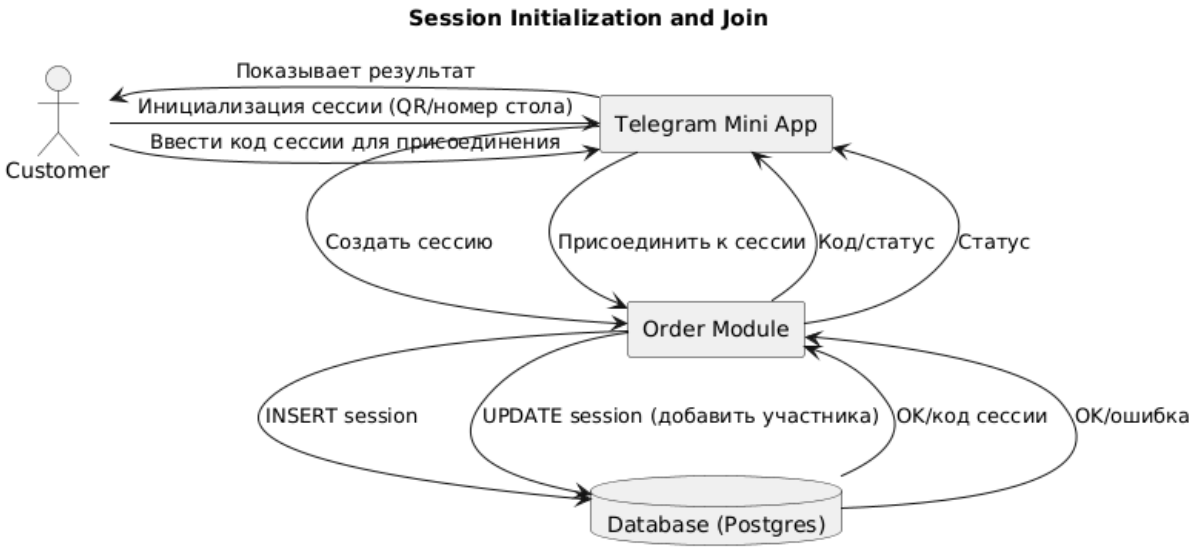
10. Просмотр и редактирование текущих заказов, статусы



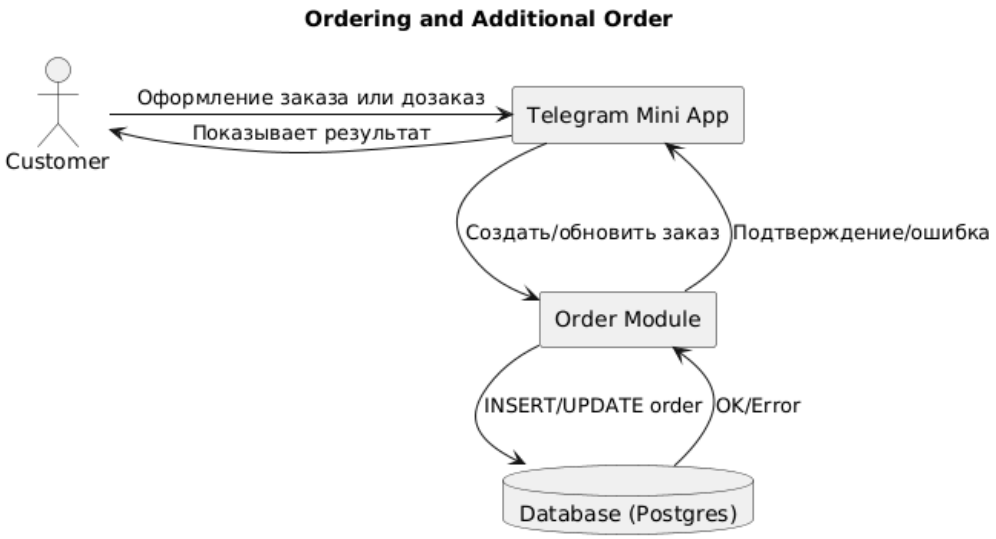
11. Просмотр и редактирование блюд (CRUD блюдо)



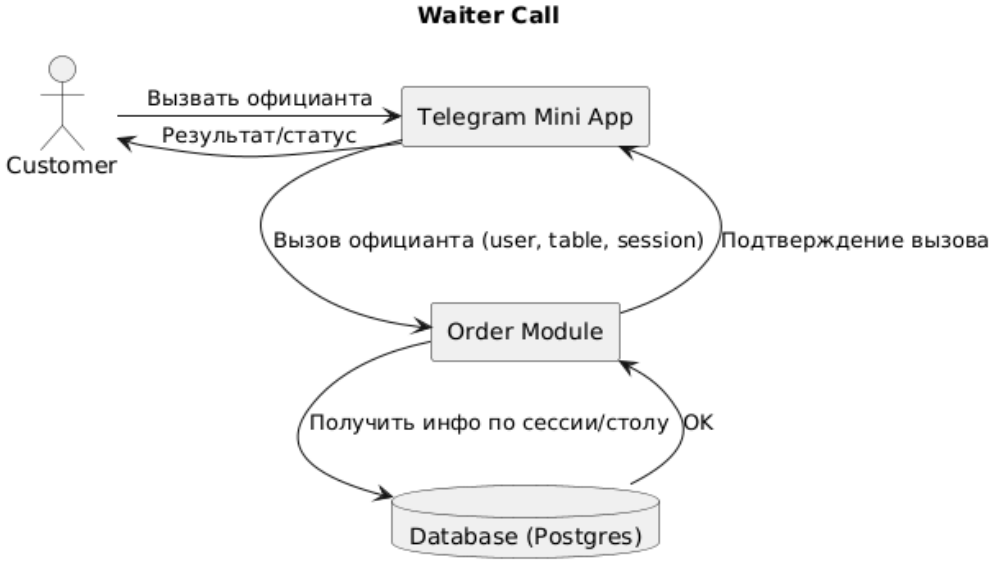
12. Инициализация и присоединение к сессии



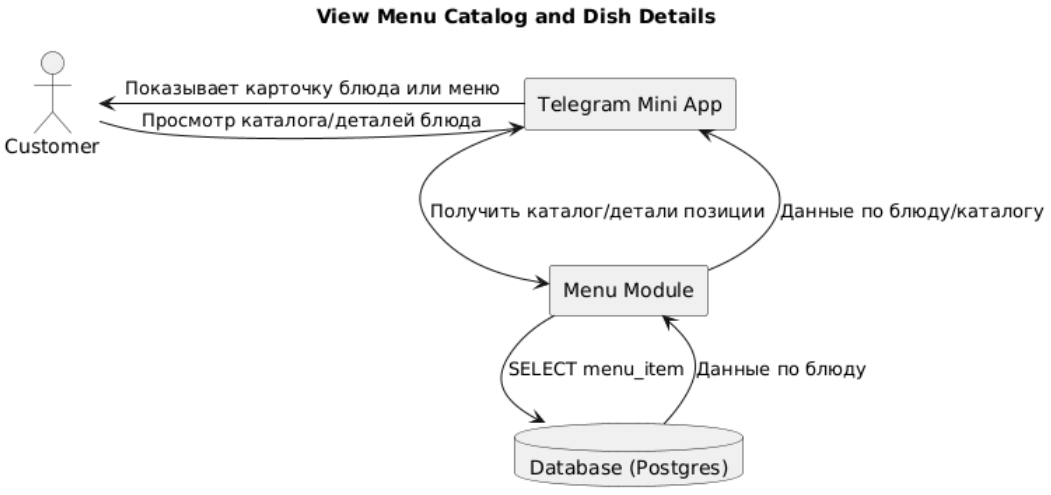
13. Оформление заказа и дозаказ



14. Вызов официанта (ещё раз, для completeness)



15. Просмотр каталога плейса и детальных карточек



ЗАР по интеграции с внешними системами

Название решения: Интеграция с внешними системами для SmartTable

1. Контекст решения

Для взаимодействия с внешними сервисами предусмотрены несколько интеграций, включая:

1. **Взаимодействие с Telegram**: Бот для взаимодействия с клиентами ресторана, который работает через Telegram Web App.

2. **Интеграция с Yandex Cloud**: Для загрузки изображений для блюд и других медиа-файлов используется **Yandex Cloud** (Object Storage).

3. **API взаимодействие с админкой и клиентским приложением**: Все запросы от клиентов и администраторов обрабатываются через **REST API**.

Система должна обеспечивать:

- Регистрацию пользователей через Telegram, включая идентификацию и аутентификацию.

- Хранение и управление медиа-файлами в **Yandex Cloud**.

- Взаимодействие с клиентами и администраторами через **REST API** с использованием **JWT** для авторизации.

2. Решение

Внешние системы и их интеграция:

1. **Telegram Web App (Фронтенд для клиентов)**:

    - Фронтенд взаимодействует с бэкендом через **REST API**.

    - Клиенты могут просматривать меню, оформлять заказы и вызывать официантов через Telegram бот, используя SDK на Golang (`telebot`).

    - Бот получает сообщения от клиента, такие как `/start`, извлекает данные (telegram_login, telegram_id, telegram_chat_id) и отправляет запросы на сервер для регистрации и аутентификации.

2. **Админка (Фронтенд для администраторов)**:

    - Веб-админка взаимодействует с бэкендом через **REST API** для управления ресторанами, меню и заказами.

3. **Yandex Cloud (Хранение медиа-файлов)**:
    - Для хранения изображений блюд и других медиа-файлов используется **Yandex Cloud**.
    - Бэкенд отправляет запросы на загрузку изображений в **Yandex Object Storage** с использованием Yandex Cloud SDK.
    - Также обеспечена возможность получения и удаления медиа-файлов по запросу.

3. Применяемые паттерны

1. **REST API**: Используется для взаимодействия между клиентским приложением, админкой и бэкендом.
2. **JWT**: Для авторизации и аутентификации пользователей, для обращения к приватным эндпоинтам
3. **Observer** (взаимодействие с Telegram): Бот отслеживает сообщения от клиентов и уведомляет администраторов о новых действиях.
4. **Adapter**: Для интеграции с Yandex Cloud и Telegram API.

4. Взаимодействие между системами

1. **Frontend (Telegram Web App)** -> **Backend (REST API)**: Запросы на создание заказа, просмотр меню, вызов официанта.
2. **Frontend (Admin Web Interface)** -> **Backend (REST API)**: Запросы на создание ресторанов, управление меню, управление заказами.
3. **Backend** -> **Yandex Cloud**: Запросы на загрузку и получение изображений для блюд.
4. **Telegram Bot** -> **Backend**: Получение данных от клиента (логин, айди, чат айди) для регистрации.
5. **Backend** -> **Yandex Cloud**: Загрузка изображений блюд в Yandex Object Storage.

5. Проблемы и ограничения

- **Зависимость от сторонних сервисов**:
    - Зависимость от **Telegram** и **Yandex Cloud** для правильного функционирования.
    - При недоступности Telegram или Yandex Cloud, необходимо будет обрабатывать ошибки и уведомлять пользователей.

- **Безопасность данных**:
    - Важно обеспечить безопасное хранение и передачу **JWT** токенов и данных пользователей.

6. Оценка результативности

Будет проверена позже

7. Альтернативы

1. Использование собственного хранилища вместо Yandex Cloud

- **Преимущества**:
    - Полный контроль над хранилищем.
    - Возможность настройки и оптимизации хранилища под свои нужды.
- **Минусы**:
    - Усложнение инфраструктуры.
    - Необходимость управлять масштабируемостью, безопасностью и отказоустойчивостью вручную.

2. Внедрение собственного механизма аутентификации вместо Telegram

- **Преимущества**:
    - Полный контроль над процессом авторизации.
    - Возможность адаптации под нужды бизнеса.
- **Минусы**:
    - Дополнительные усилия по реализации безопасности, хранению паролей, восстановлению паролей и т. д.
    - Усложнение пользовательского опыта.

3. Использование другого облачного хранилища (например, Google Cloud Storage)

- **Преимущества**:

    - Возможность выбора более подходящего решения в зависимости от ценовой политики.

    - Хорошо поддерживаемая инфраструктура с расширенными функциями.

- **Минусы**:

    - Нужно будет изменить интеграцию с облачным хранилищем, что потребует дополнительных усилий.

8. Последствия

Положительные

- **Гибкость и масштабируемость**:

    - Легко интегрировать новые внешние системы (например, сменить облачное хранилище или добавить нового провайдера аутентификации).

    - **Yandex Cloud** и **Telegram** предоставляют высокую доступность и масштабируемость.

- **Производительность**:

    - Использование **Yandex Cloud** обеспечивает быструю загрузку и хранение медиа-файлов.

    - Уменьшение нагрузки на сервер с использованием облачного хранилища.

- **Безопасность**:

    - **JWT** и **Telegram API** обеспечивают безопасное взаимодействие и управление доступом.

    - Все данные защищаются с помощью **HTTPS**.

Отрицательные

- **Зависимость от внешних сервисов**:

    - В случае недоступности **Telegram** или **Yandex Cloud** система может временно не функционировать.

- **Дополнительные сложности**:

    - Требуется поддержка нескольких внешних сервисов, что увеличивает сложность инфраструктуры.

9. Риски

1. **Невозможность доступа к Telegram или Yandex Cloud**:
    - При недоступности этих сервисов пользователи не смогут взаимодействовать с системой.
    - Возможность использования альтернативных каналов связи с пользователями.
2. **Проблемы с безопасностью данных**:
    - Возможность утечек данных, если не будет правильно настроено безопасное хранение токенов и медиа-данных.
3. **Изменения в API внешних сервисов**:
    - Изменения в API Telegram или **Yandex Cloud** могут потребовать обновления интеграции.

OpenAPI на взаимодействие с внешним S3-хранилищем

Ссылка на API: https://pastebin.com/k2iTMRfX

OpenAPI сервиса SmartTable

Ссылка на API: https://pastebin.com/WLWPzbsr

Telegram API

Ссылка на докуменатицию:
https://github.com/tdlib/telegram-bot-api?tab=readme-ov-file
Используемое SDK в коде: https://github.com/tucnak/telebot

Примечание

Ссылка на репозиторий, где часть эндпоинтов уже реализованы:
https://github.com/aubhona/smart-table/tree/main/backend/smart-table/docs/api