# Текстовое описание задачи и структуры проекта «Rfae»

**Содержание**

[Текстовое описание задачи и структуры проекта «Rfae» 1](#_Toc212162401)

[1. Задача 1](#_Toc212162402)

[2. Архитектура хранилища 2](#_Toc212162403)

[2.1. PostgreSQL 2](#_Toc212162404)

[2.2 MongoDB 2](#_Toc212162405)

[3. Файловая структура проекта 3](#_Toc212162406)

[4. Демонстрация работы запросов 4](#_Toc212162407)

## 1. Задача

Построить реляционную БД (PostgreSQL) для сетевого кафе с (MongoDB) для записи операций и отзывов. Обеспечить:

* Все **пункты меню (MenuItem)** хранятся в центральной базе и используются во всех филиалах.;
* Поддерживается возможность **переопределения цен** или доступности блюд на уровне конкретного филиала;
* В системе реализован **полный цикл заказа**: создание, добавление позиций, расчет суммы, изменение статуса;
* Добавление сотрудников филялов, создание заявки на поставку продуктов, редактирование меню, категорий и продукции;
* Сбор и хранение операций пользователей и отзывов клиентов в системе.

## 2. Архитектура хранилища

### 2.1. PostgreSQL

Архитектура построенная на postgresql включает в себя, сущности базы данных: CafeBranch, Employee, Customer, MenuCategory, MenuItem, Order, OrderItem, Payment, Supplier, SupplyOrder, Inventory;

Схема БД изображена на рисунке 1.

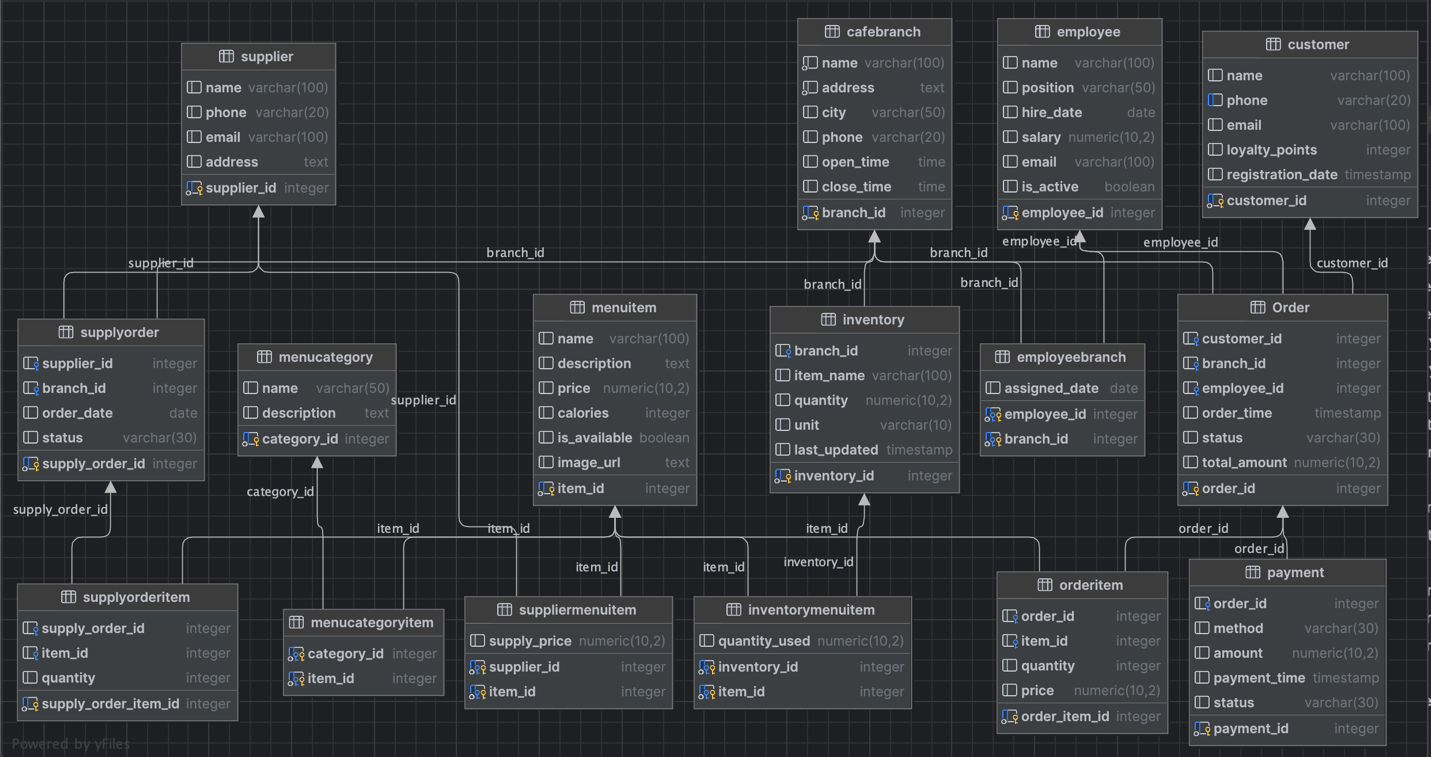


Рис. 1 – Схема БД Кафе

### MongoDB

Архитектура построенная на mongodb включает в себя коллекции: logs, rewiews.

Подробное описание коллекций с операциями и отзывами, связанных с ними представлено на рисунках 2 и 3.

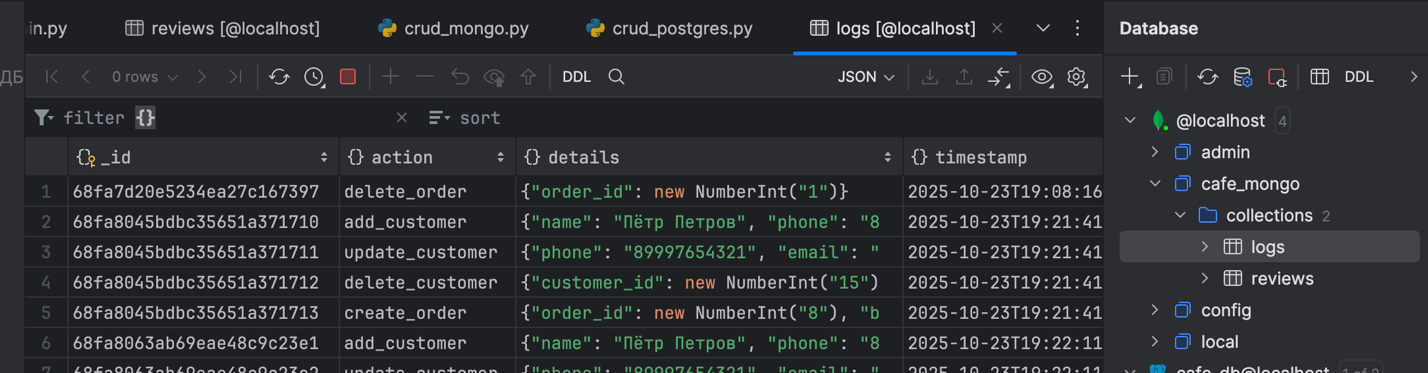


Рис. 2 – Операции с базой данных

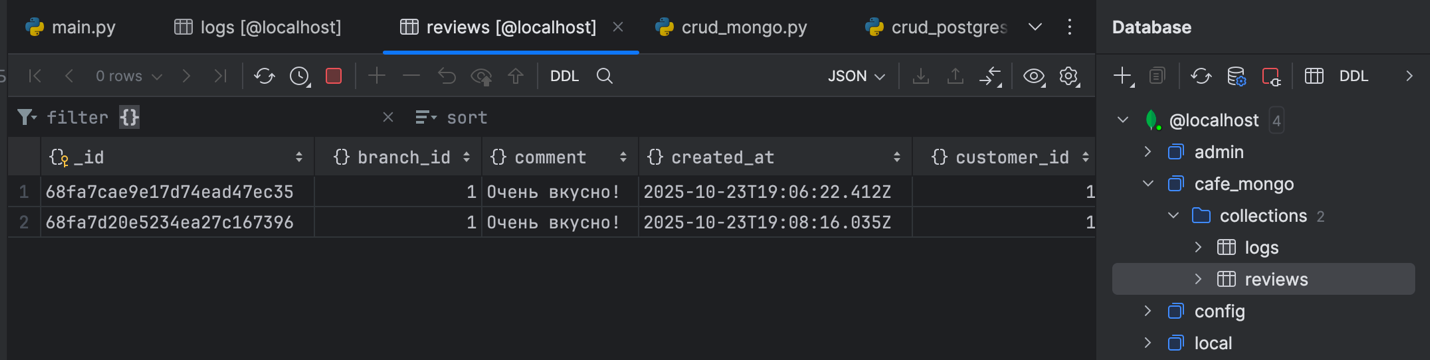


Рис. 3 – Отзывы

## 3. Файловая структура проекта

Подробное описание файловой структуры проекта представлено в таблице 1.

Таблица 1. Файловая структура проекта.

|  |  |
| --- | --- |
| Папка / Файл | Назначение |
| App/main.py | Главная точка входа, которая объединяет PostgreSQL и MongoDB. Здесь происходит демонстрация CRUD и логирования. |
| app/ crud\_postgres.py | Здесь сосредоточена вся работа с PostgreSQL: создание, чтение, обновление и удаление данных по таблицам кафе. |
| App/ crud\_mongo.py | Здесь реализуется логика для дополнительных коллекций: feedback, order\_history, logs. |
| docker-compose.yml | Контейнеризация проекта, чтобы PostgreSQL и MongoDB запускались вместе. |
| requirements.txt | Файл для проекта с актуальными библиотеками |
| Postgres/Dockerfile | Подготовка и инициализация базы данных при старте контейнера. |
| Postgres/ init.sql | Создание структуры базы данных с ключами, связями и ограничениями. |
| Postgres/ migration.sql | Наполнение базы данных начальными тестовыми данными. |

## 4. Демонстрация работы запросов

Тестирование данного инструмента для демонстрации работы с базами данных можно описать как последовательное прогонение всех функций в режиме “живого” сценария работы кафе. Сначала мы проверяем действия клиента, что позволяет убедиться, что функции для работы с таблицей клиентов и заказами работают корректно. Скрипт создает нового клиента, присваивает ему уникальный идентификатор и сразу же логирует это действие. Далее обновляются данные клиента, например, телефон и email, после чего производится вывод всех клиентов, чтобы визуально убедиться, что изменения применились. После этого тестируется удаление клиента, и снова проверяется состояние таблицы, чтобы убедиться, что удаление прошло корректно. Одновременно проверяется корректность логирования этих операций в MongoDB, что дает возможность отслеживать историю действий.

Следующим шагом является проверка взаимодействия с меню и филиалами. Сценарий выводит все доступные позиции меню, их описание и цену, а также список филиалов с указанием города и адреса. Это позволяет проверить корректность выборки данных и работу join-запросов в базе. После этого создается заказ, к которому добавляются позиции из меню. Скрипт автоматически обновляет итоговую сумму заказа и выводит всю информацию о нем, включая позиции и стоимость. Таким образом тестируется правильность работы функций create\_order, add\_order\_item и update\_order\_total, а также их интеграция с клиентской таблицей.

Далее проверяются действия сотрудников. Скрипт выводит список всех клиентов, что позволяет сотруднику видеть текущую базу. Потом изменяется статус одного из заказов, создается заявка на поставку с добавлением нескольких товаров, и проверяется, что эти действия корректно отражаются в базе. Также тестируется управление складом: создается новая запись в инвентаре, затем изменяется количество, и выводится обновленное состояние склада. Такой сценарий позволяет убедиться, что функции для работы со складом и поставками корректно изменяют данные и что все изменения можно отследить через логирование.

Административные действия проверяются путем добавления нового сотрудника, изменения его данных и последующего удаления. Также тестируется удаление заказа и ручное логирование этого действия в MongoDB. Сценарий позволяет убедиться, что все операции CRUD для сотрудников и заказов работают корректно, что данные корректно отображаются при выборке и что логирование фиксирует все действия администратора.

Наконец, отдельным этапом тестируются сложные связи между сущностями. Скрипт выводит категории меню и привязанные к ним позиции, поставщиков и связанные с ними товары, сотрудников и их филиалы, а также поставщиков для каждой позиции меню. Этот блок демонстрации позволяет убедиться в правильности работы функций для many-to-many связей и join-запросов, а также проверяет корректность отображения данных в различных контекстах. Все эти действия выполняются последовательно, что позволяет наблюдать за взаимосвязями между таблицами и корректностью бизнес-логики.