САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №1

Выполнил:

Платонова Александра

Группа K3439

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

Необходимо спроектировать набор следующих диаграмм:

- общая архитектура решения (сервисы и их взаимосвязи, клиент-серверное взаимодействие);
- диаграмма компонентов;
- диаграммы БД по каждому сервису;
- диаграммы основных пользовательских сценариев (те сценарии, которые позволяют вашим приложением полноценно воспользоваться, пройти весь путь).

Проект: Прикладное программное обеспечение деятельности отдела заселения муниципальных общежитий администрации города.

Ход работы

В первую очередь была спроектирована общая архитектура приложения, представленная на рисунке 1. Клиентская часть – Frontend приложение, взаимодействующее с сервером через API Gateway. Сервер состоит из микросервисов: Hostel Service (управление общежитиями, комнатами и адресами), Resident Service (управление жителями), CheckInOut Service (управление заселением/выселением) и Payment Service (управление платежами). Каждый сервис имеет свою базу данных для обеспечения независимости. Взаимосвязи осуществляются через REST API.

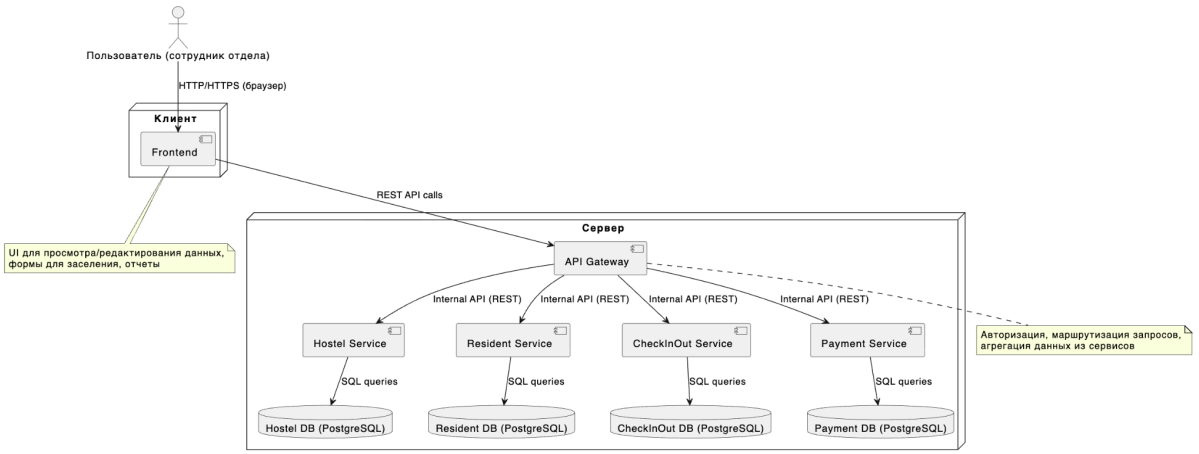

Рисунок 1 – Общая архитектура приложения

Далее была построена диаграмма компонентов (рис. 2), которая отражает логическое разбиение системы на уровни: клиентский интерфейс, API-шлюз, бизнес-сервисы и уровень хранения данных. Выделение API Gateway позволяет унифицировать доступ клиента к различным сервисам и централизовать вопросы безопасности и авторизации. Каждый бизнес-сервис реализует строго определенную функциональность. Такое разделение компонентов облегчает тестирование, повторное использование и дальнейшую модернизацию системы.

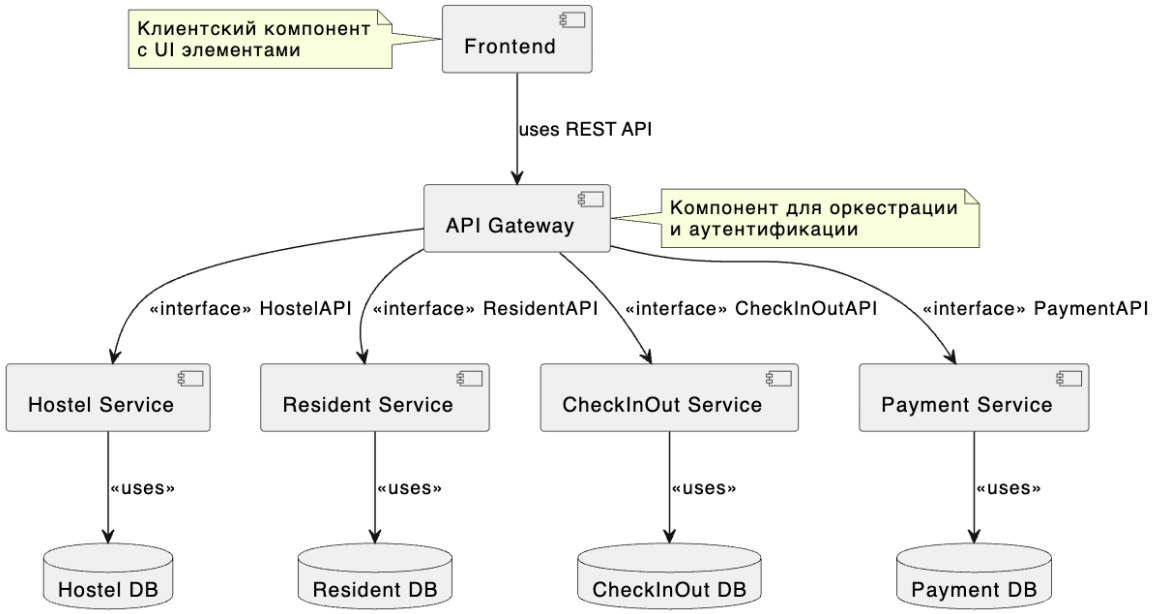

Рисунок 2 – Диаграмма компонентов

Далее были составлена диаграммы БД для каждого из микро-сервисов. На рисунке 3 представлены все ключевые сущности системы, распределенные по четырем независимым базам данных микросервисов и визуально разделенные цветовым кодированием.

Каждая БД отвечает за свою предметную область: Hostel DB отвечает за структуру общежитий, адреса и помещения; Resident DB ответственна за персональные данные граждан, категории очереди и документы; CheckInOut DB – за договоры и историю заселения/выселения; Payment DB отвечает за тарифы, начисления и фактические платежи.

Сплошные линии обозначают прямые внешние ключи внутри одной базы данных, пунктирные показывают межсервисные связи,

реализованные через обмен идентификаторами (foreign keys по ID). Такой подход обеспечивает высокую степень изоляции сервисов, упрощает масштабирование и повышает отказоустойчивость системы в целом. Описание сущностей:

1. Hostel DB
– Address, адресные данные зданий (улица, дом, район, почтовый индекс);
– Hostel, общежитие в целом (название, год постройки, общее количество мест, статус);
– Room, отдельная комната в общежитии (номер, этаж, вместимость, текущий статус, площадь);
– RoomEquipment, оборудование и инвентарь в комнате (тип, количество, инвентарный номер).
2. Resident DB
– Person, физическое лицо (ФИО, дата рождения, паспортные данные, телефон);
– Resident, запись о гражданине в очереди на жилье (номер в очереди, категория, доход на человека, статус);
– Document, документы, подтверждающие нуждаемость (тип, номер, дата выдачи, скан).
3. CheckInOut DB
– Contract, договор на проживание (номер, даты начала/окончания, статус, ссылки на жителя и комнату);
– CheckInOutRecord, фактические события заселения и выселения (даты въезда/выезда, количество мест).
4. Payment DB
– Rate, действующий тариф проживания (название, цена за место, период действия);
– Accrual, ежемесячное начисление оплаты (сумма, период, ссылка на договор и тариф);
– Payment, факт оплаты (сумма, дата, способ оплаты, статус).

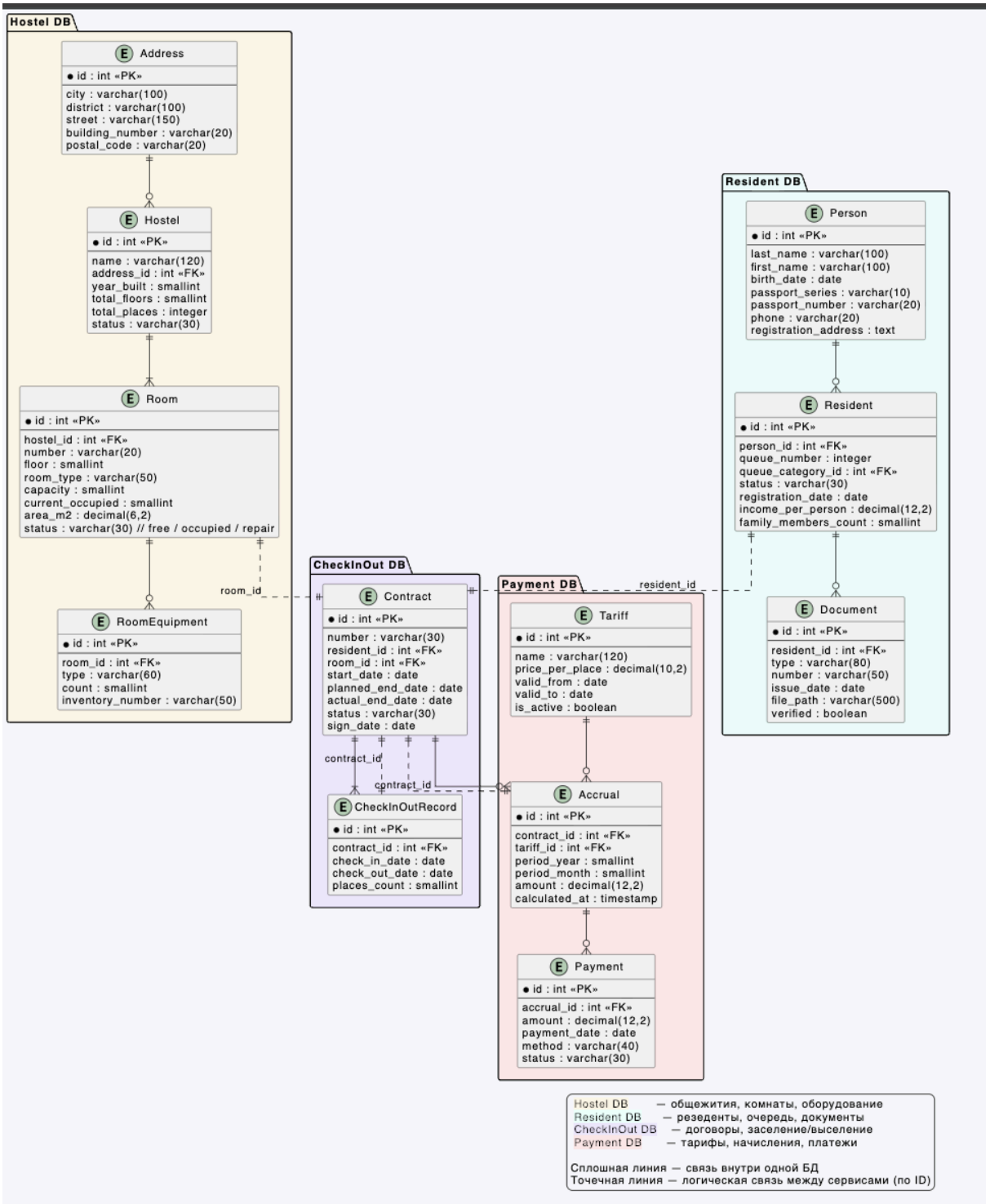

Рисунок 3 – Диаграммы БД для каждого микро-сервиса

Затем были составлены различные пользовательские сценарии. На рисунке 4 изображен полный процесс заселения очередника в свободную комнату общежития. Сначала система проверяет данные жителя, находит подходящие свободные комнаты по заданным критериям (вместимость, общежитие, этаж), после чего сотрудник выбирает комнату и подтверждает

заселение. В результате создается договор, фиксируется дата въезда, статус комнаты меняется на «занята», а вся информация записывается в соответствующие базы данных.

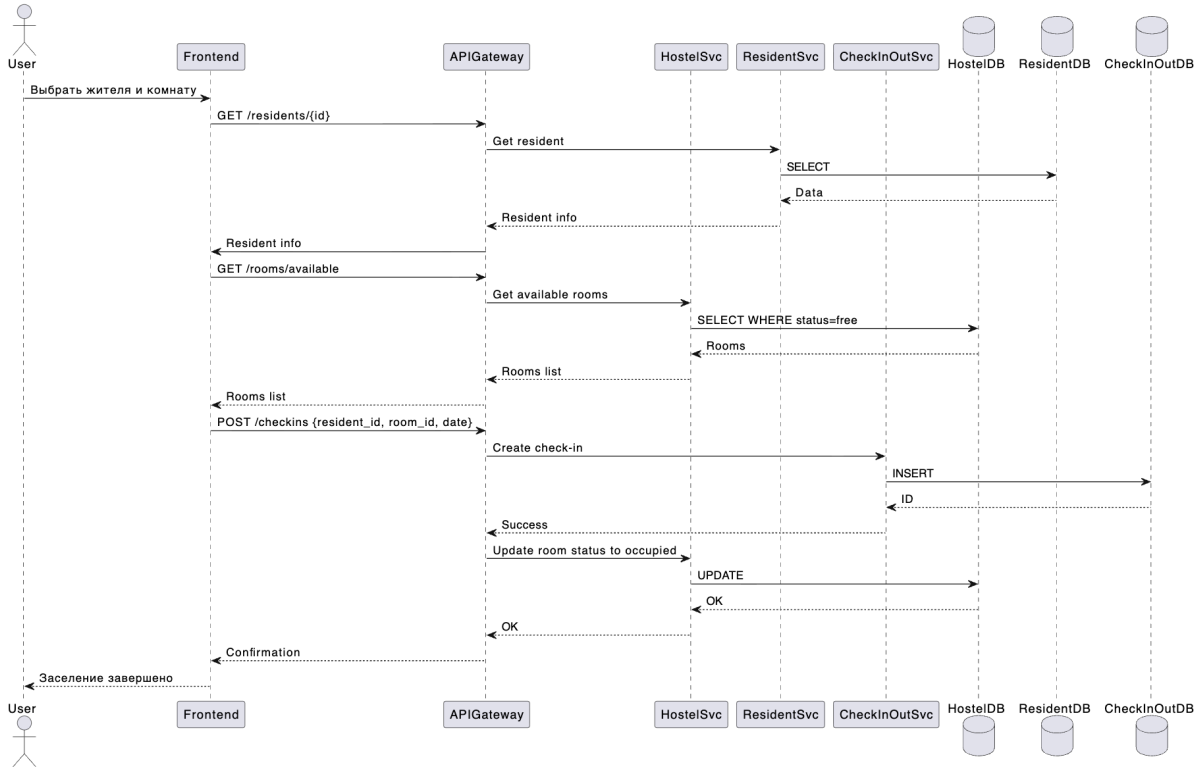

Рисунок 4 – Сценарий заселения в общежитие

Следующий сценарий описывает процесс регистрации платежа за проживание по действующему договору (рис. 5). Сотрудник видит текущее начисление за месяц, получает информацию о сумме, после чего фиксирует факт оплаты с указанием даты, способа и суммы. После успешной регистрации платежа обновляется статус начисления, а при необходимости – информация о задолженности.

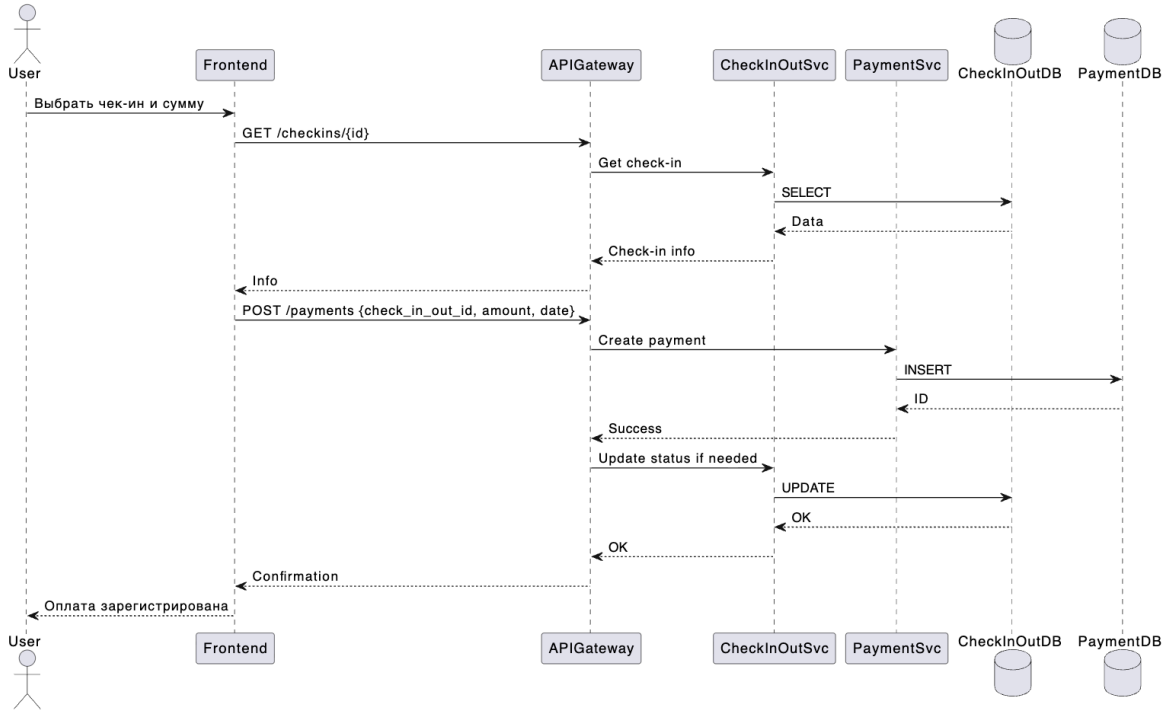

Рисунок 5 – Сценарий оплаты проживания

Следующий сценарий - переселение резидента в другую комнату. Сотрудник находит текущий активный договор жителя, выбирает новую комнату (которая должна быть свободной), создает запись о переезде, обновляет статус старой и новой комнаты, фиксирует дату и причину переселения. Все должно происходить атомарно, чтобы не возникло ситуации «две комнаты заняты одним человеком» или «комната осталась висеть как занятая». В случае ошибки на любом этапе необходимо выполнить роллбэк и откатить все выполненные ранее операции. Этот сценарий показан на рисунке 6.

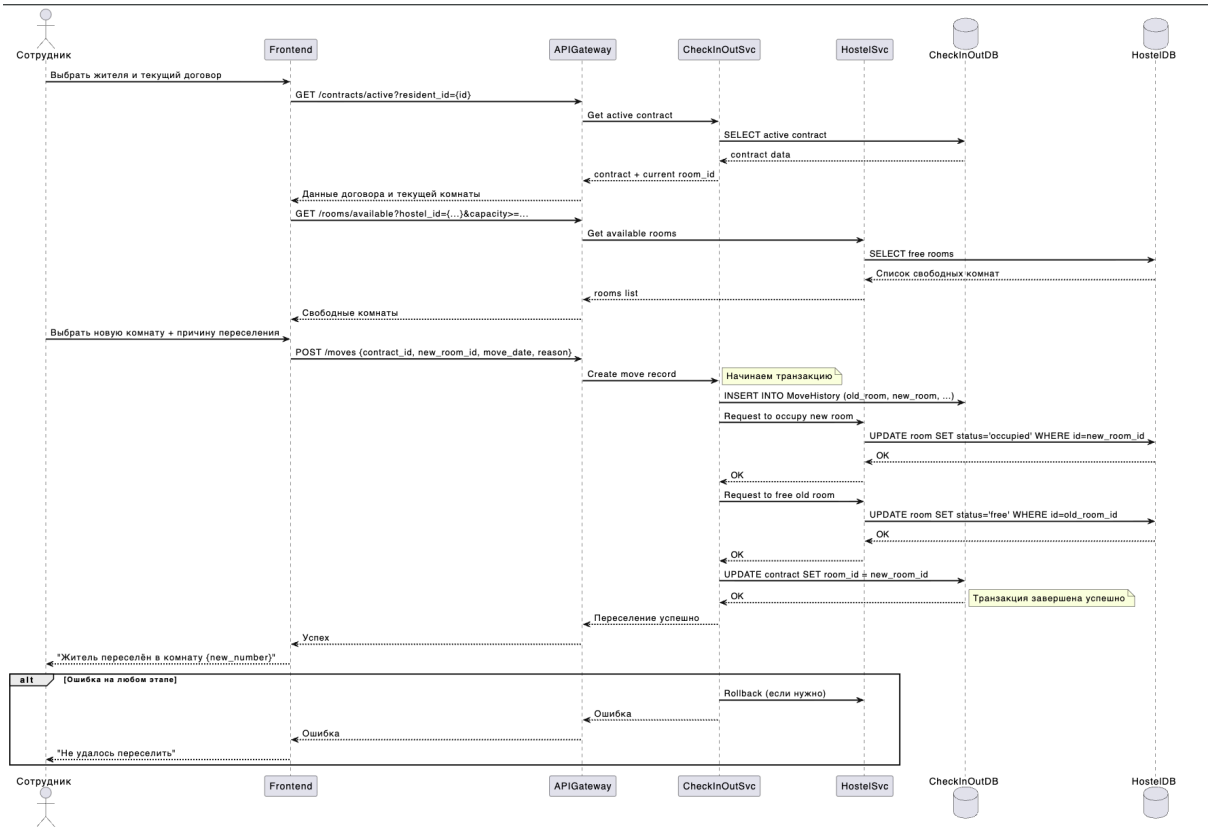

Рисунок 6 – Сценарий переселения резидента

Вывод

В ходе выполнения домашнего задания система была разделена на 4 микро-сервиса, каждый из которых отвечает за отдельный участок бизнес-логики. На основании этого разделения были составлены следующие диаграммы: общая диаграмма архитектуры, диаграмма компонентов, модели баз данных каждого из микро-сервисов. Также были разработаны пользовательские сценарии, с помощью которых можно наглядно оценить взаимодействие между компонентами системы.