ТЗ №2 — Fullstack-разработчик

Задание 1

Цель задания:

Реализация веб-приложения для отслеживания статуса заказов (Order Tracking Application).

Функциональные требования:

Backend (.NET 8, ASP.NET Core):

- Создание и получение заказов:
  - Заказ содержит поля: номер заказа (OrderNumber), описание (Description), статус (Status: создан, отправлен, доставлен, отменен), дата создания (CreatedAt), изменения (UpdatedAt).
- Асинхронная отправка событий в очередь Kafka/RabbitMQ при изменении статуса заказа.
- Сервис подписки на изменения статуса заказа для передачи уведомлений пользователям через WebSocket или Server-Sent Events.

Frontend (React):

- Создание интерфейса для отображения и отслеживания статуса заказов:
  - Список заказов и форма создания нового заказа.
  - Страница деталей заказа с возможностью отслеживания статусов заказа (автообновление через WebSocket/SSE).
  - (**Опционально**) Реализовать список отслеживаемых заказов с помощью менеджера состояния.

Технические требования и стек:

- Backend:
  - .NET 8, ASP.NET Core Web API
  - Использование ORM системы для работы с БД. Предпочтительно - EF Core
  - БД: желательно использовать PostgreSQL или Microsoft SQL Server
  - Kafka/RabbitMQ для событийного взаимодействия
  - Observability: логирование (вывод на консоль или в текстовый файл)
- Frontend:
  - React (обязательное использование TypeScript)
  - WebSocket, Server-sent events (по выбору)
  - Redux/Zustand или аналогичные инструменты для простого управления состоянием

Требования к реализации кода:

- Чистый, читаемый код, SOLID, разделение на компоненты и слои.
- Создание файла README с инструкцией запуска backend и frontend (достаточно простого сценария запуска).

Дополнительные требования (опционально):

- Docker-контейнер, docker-compose.yml (для удобной сборки и запуска).

- Покрытие кода unit-тестами.
- Наличие XML-документация к классам, методам и свойствам.
- Добавление трассировки (OpenTelemetry)

Задание 2

Цель задания:

Необходимо написать табличную функцию SQL, которая будет возвращать по ClientId и интервалу дат (тип Date) поденные суммы платежей. Если за указанный день не было платежей, то функция должна возвращать 0. Интервалы дат могут охватывать несколько лет.

Структура данных:

Есть таблица платежей клиентов ClientPayments следующего вида:

```
ClientPayments
(
        Id bigint, -- первичный ключ таблицы
        ClientId bigint, -- Id клиента
        Dt datetime2(0), -- дата платежа
        Amount money, – сумма платежа
)
```

Пример входных данных:

Таблица client.Payments

Id	ClientId	Dt	Amount
1	1	2022-01-03 17:24:00	100
2	1	2022-01-05 17:24:14	200
3	1	2022-01-05 18:23:34	250
4	1	2022-01-07 10:12:38	50
5	2	2022-01-05 17:24:14	278
6	2	2022-01-10 12:39:29	300

Примеры работы:

Результат работы функции №1:

Входные данные:

- ClientId = 1
- Sd = 2022-01-02

- Ed = 2022-01-07

Результат функции:

Dt	Сумма
2022-01-02	0
2022-01-03	100
2022-01-04	0
2022-01-05	450
2022-01-06	0
2022-01-07	50

Результат работы функции №2:

Входные данные:

- ClientId = 2
- Sd = 2022-01-04
- Ed = 2022-01-11

Результат функции:

Dt	Сумма
2022-01-04	0
2022-01-05	278
2022-01-06	0
2022-01-07	0
2022-01-08	0
2022-01-09	0
2022-01-10	300
2022-01-11	0