Демонстрационный проект для доставки товаров с самовывозом из пункта выдачи

GitHub: https://github.com/sevolchenko/items-delivery-demo

Этот проект — учебный. Для меня он стал возможностью развить навыки в написании микросервисов на Spring Boot, а также попробовать работу с Git и Docker. Другому разработчику было важно показать, как можно использовать Camunda для управления бизнес-процессами для демонстрации на митапе для студентов. Вместе мы сделали простой проект, чтобы попрактиковаться в работе с микросервисами и их взаимодействием.

Проект представляет собой микросервисную систему, разработанную на языке Kotlin с использованием фреймворка Spring Boot. Система автоматизирует процесс обработки клиентских заявок на получение товаров — от оформления заявки до выдачи товара клиенту оператором.

Система построена по микросервисной архитектуре, взаимодействие между сервисами осуществляется как синхронно (для синхронного взаимодействия между микросервисами используется REST API с описанием через OpenAPI (Swagger)), так и асинхронно через Apache Kafka.

Проект использует Docker для контейнеризации каждого микросервиса.

Используемые технологии

- **Язык:** Kotlin
- **Фреймворк:** Spring Boot
- **Взаимодействие между микросервисами:** Apache Kafka
- **Хранилище данных:** PostgreSQL

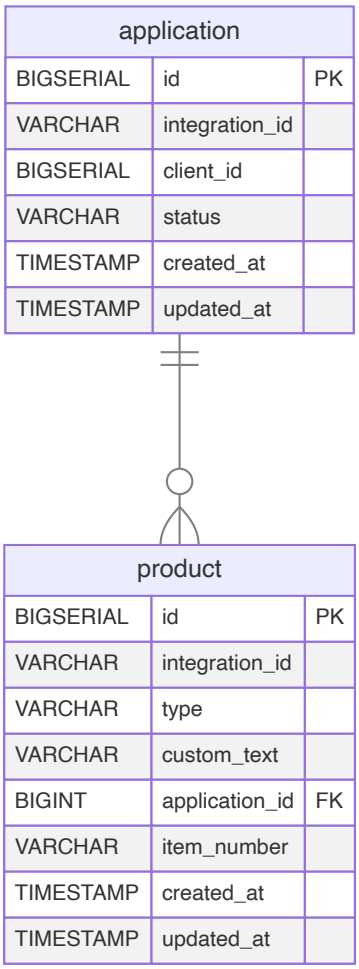
Веб-интерфейс

Фронтенд-сервис для взаимодействия с клиентом: принимает заказы, отображает статусы и передает данные в бэкенд.

Хранилище заявок

Основное хранилище, которое хранит заявки, их статусы и связанные данные (ID клиента, тип товара и т. д.)

Схема базы данных:



Оркестратор

Оркестратор бизнес-процесса: управляет статусами заявок, координирует взаимодействие между сервисами

Хранилище товаров

Каталог товаров: отвечает за резервирование, предоставление характеристик (размеры, тип) и управление их доступностью

Схема базы данных:

item		
VARCHAR	id	PK
VARCHAR	type	
VARCHAR	color	
VARCHAR	status	
TIMESTAMP	created_at	
TIMESTAMP	updated_at	

Центр уведомлений

Сервис рассылки: отправляет клиентам коды получения, уведомления о статусе заказа

Приложение для оператора

Бэкенд для управления задачами операторов: создание, назначение и обновление статусов задач на сборку

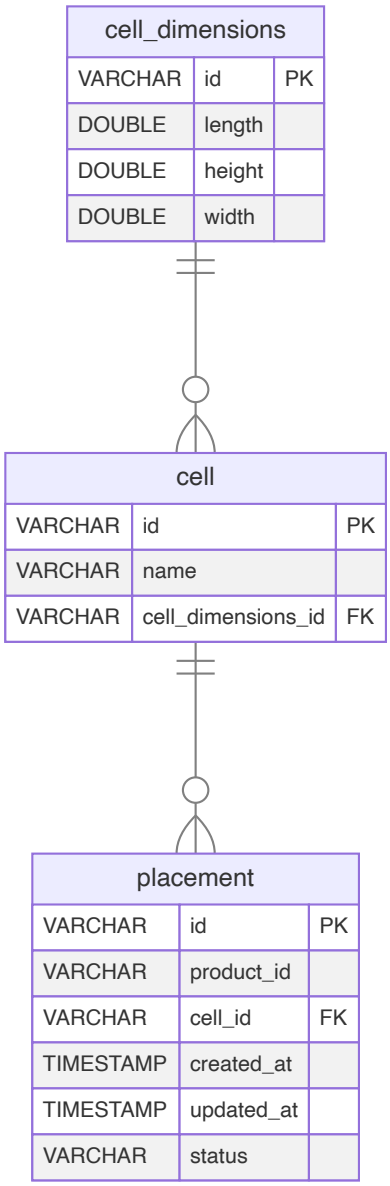
Схема базы данных:

task		
BIGSERIAL	id	PK
VARCHAR	task_id	
VARCHAR	application_id	
VARCHAR	placement_id	
VARCHAR	pickup_code	
VARCHAR	type	
VARCHAR	status	
VARCHAR	operator_id	
TIMESTAMP	created_at	
TIMESTAMP	updated_at	

Склад

Управление складскими ячейками: резервирование места под товар, поиск свободных ячеек, завершение хранения

Схема базы данных:



Интерфейс оператора

Фронтенд для оператора склада: отображает задачи, инструкции по размещению/извлечению товара и статусы

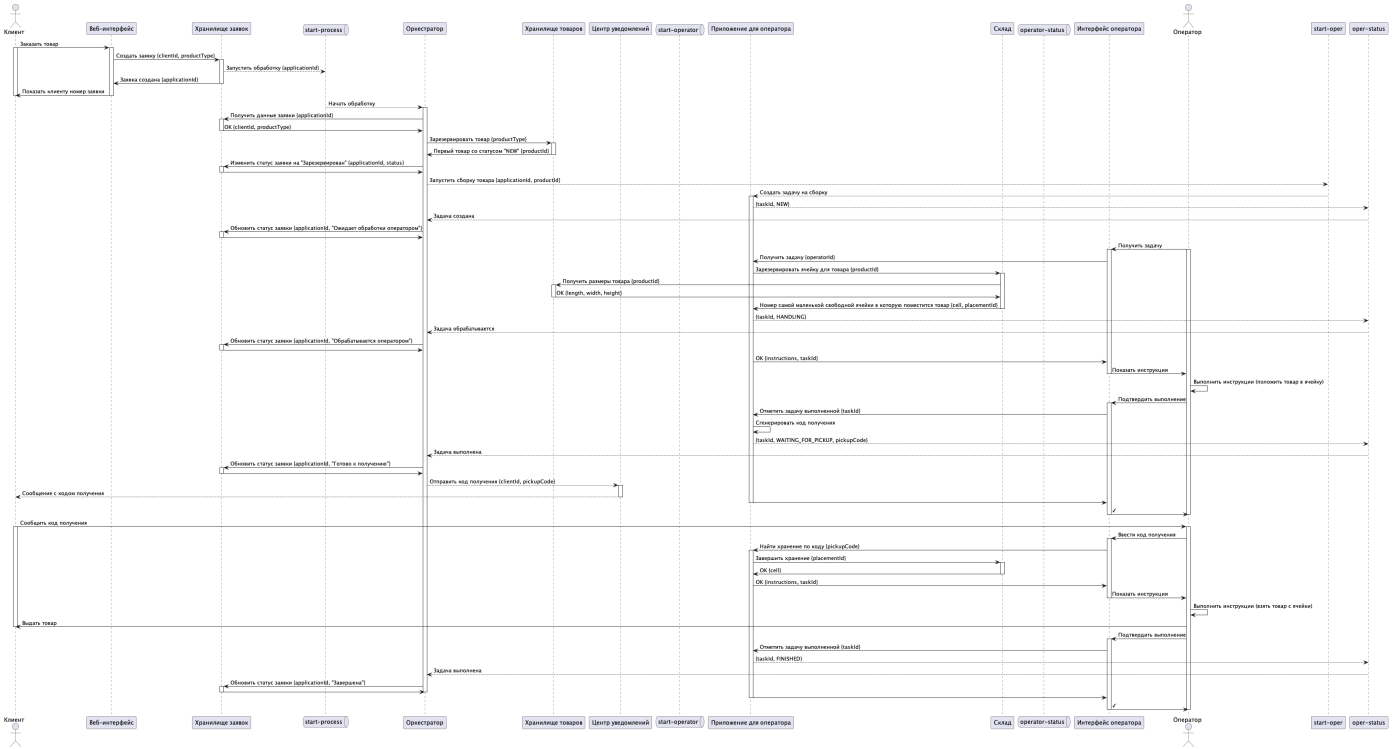
Основные этапы бизнес-процесса

1. **Создание заявки**: Клиент оформляет заказ через веб-интерфейс. Заявка сохраняется в и передается в Kafka-топик `start-process`.
2. **Обработка заявки**: Оркестратор читает событие, извлекает информацию о заявке, резервирует товар и обновляет статус.

3. **Сборка товара**: Оркестратор инициирует задачу сборки через Kafka-топик `start-operator`. Оператор получает задачу, резервирует ячейку, помещает товар и подтверждает выполнение. Система уведомляет клиента о готовности.
4. **Выдача товара**: Клиент сообщает оператору код. Оператор находит товар и выдает его клиенту, после чего задача помечается как завершённая.

Диаграмма последовательности взаимодействия микросервисов при обработке заявки клиента

Диаграмма последовательности, описывающая процесс обработки заявки клиента от создания до выдачи товара, включая взаимодействие всех задействованных микросервисов (application-storage, items-controller, items-keeper, operator-back) и внешних участников (клиент, оператор).



На диаграмме представлено базовое последовательное взаимодействие компонентов в рамках стандартного сценария. Обработка ошибок и исключительные случаи (такие как отсутствие товаров нужного типа или свободных ячеек) не рассматриваются.