**Требования к функционалу:**

1 – Просмотр новостей

2 – Возможность записаться на прием к врачу

3 - Возможность следить за блогом и рекомендациями врача

4 – Индивидуальная онлайн консультация с врачом

5 – Админка

**Бизнес ценность проекта.**

В первую очередь проект носит сугубо образовательный характер и не призван решать реальных задач бизнеса. Рарзработка проекта ведется с целью тренировки микросервисной архитектуры, подхода DDD и TDD. Изучить работу с брокером сообщений RabbitMQ и изучение технологии контейниризации Docker и оркестратора Kubernetes. Проект будет работать с реляционной базой данных PostgreSql и предпологает добавление в один из сервисов работу с нереляционной базой данных MongoDb через ORM EntityFrameworkCore. Разработчик в своем уме и понимает, что некоторые из вышеперечисленных технологий не совсем подходят для реализации маленького домашнего проекта, но как сказал Мао Дзе Дун: «А хуле нет, когда да»

**Описание архитектуры проекта**

Так как проект является учебным выбор архитектуры сведен к микросервисам на основе шаблона “Api Gateway”

Api Gateway реализован на базе «Clean Architecture» с подходом “Domain Driven Design” и “Test Driven Development”

Взаимодействие проекта апи с другими частями проекта проходит с приминением приципа CQRS с использованием библиотеки MediatR.

Решение разделено на 3 основных проекта-слоя:

1 – Слой Api. Проект ответственнен за получение запросов и возвращение ответов клиенту. Также в проекте производятся настройки библиотеки документации Swagger и всех промежуточных цепочек запроса такие как присвоения индивидуального уникального ключа для каждого запроса для отслеживания цепочки вызвов сервиса

2 – Слой Application. Проект ответственнен за предоставление внешних интерфейсов взаимодействия с методами реалзиации инфраструктуры проекта для достижения слобой взаимосвязи проектов посредством применения принципа инъекции зависимостей. Содержит в себе проект Domain в котором описаны бизнес модели. Описывает поведение валидаров тела запроса и поведение логирования. Содержит в себе команды и запросы для разделение процессов записи и чтения данных.

3 – Слой Infrastructure. Проект ответственнен за взаимодействией с другими сервисами, перенаправление запросов в другие ответственные сервисы. Взаимодействует в случае необходимости с базой данных и производит аггрегаций бизнес моделей. Также содержит методы реализации внутренних сервисов решения из слоя Application.

Таким образом слой Api имеет доступ к слою Application и к слою Infrastructure. Слой Infrastructure имеет доступ к слою Application. Подобная связь между слоями гарантирует слабую связность и недопускает зацикливания зависимостей. Слой Application может быть зависимостью для любого из проектов, но не должен иметь собственных зависимостей (исключение слой Shared).

Опциональным слоем в решении является слой Shared в котором сфокусированы реализации различных расширений и библиотек. Может быть зависимостью для любого из проектов, но не должен иметь собственных зависимостей.

**Библиотеки и технологии которые необходимо использовать в проекте**

**Ocelot –** в качестве шлюза

**MediatR –** для разделения логики записи и чтения данных

**FluentValidation –** для упрощения процесса валидации моделей

**EntityFrameworkCore –** для работы с хранилищем данных

**RabbitMQ** – в качестве брокера сообщений для асинхронного общения между сервисами

**Docker** – для контейнеризации приложений

**Kubernetes** – для оркестрации контейнеров

**ELK** – для логирования