

ЭТАП 2. АРХИТЕКТУРА ПРОГРАММНОГО РЕШЕНИЯ

Заказчик:	Хади Салех, НИУ ВШЭ
Название проекта:	Анимированный переводчик для обмена сообщениями
Исполнители	Малашенко Борис Тарасович, Семенкович Тимофей Алексеевич

1) ОПИСАНИЕ И НАЗНАЧЕНИЕ ПРОГРАММЫ

Разработка API для синтеза речи и веб-мессенджера для общения в личных и групповых чатах с возможностью перевода сообщений и синтеза речи.

Благодаря приложению клиенты смогут общаться с носителями других языков “на равных”, понимая не только основную информацию, написанную собеседником, но и его эмоции, переданные в синтезированной речи.

Итоговый продукт должен отвечать условиям надежности, удобства сопровождения и быть удобным в использовании.

2) Описание модели Жизненного Цикла

В данной работе мы будем использовать гибкую модель разработки (рис. 1). Данная модель подходит нашей разработке из-за небольшой команды разработчиков, высокой вовлеченности заказчика и возможности разработки требований по ходу написания проекта. Agile обладает следующими преимуществами:

- Возможность развивать продукт, добавляя новый функционал по ходу разработки;
- Качество продукта обеспечивается тестированием на протяжении всей разработки;
- Снижение риска, так как команда выявляет и исправляет любые проблемы на ранней стадии.

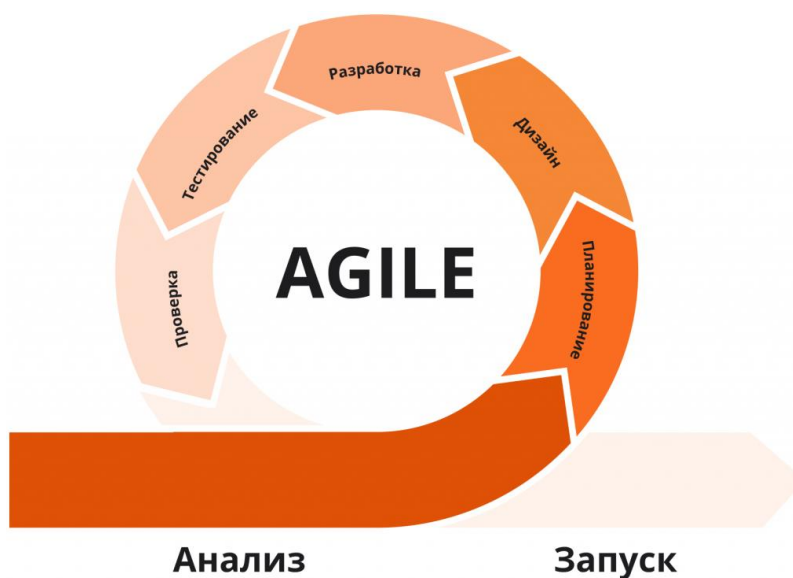


Рисунок 1 - Гибкая модель разработки Agile

3) Описать используемые паттерны проектирования (Design Patterns)

1. Наблюдатель (Observer)

- Создает механизм подписки и позволяет реализовать отправку сообщений в личном или групповом чате.
- Это поведенческий паттерн проектирования, который создает механизм подписки, позволяющий одним объектам следить и реагировать на события, происходящие в других объектах.

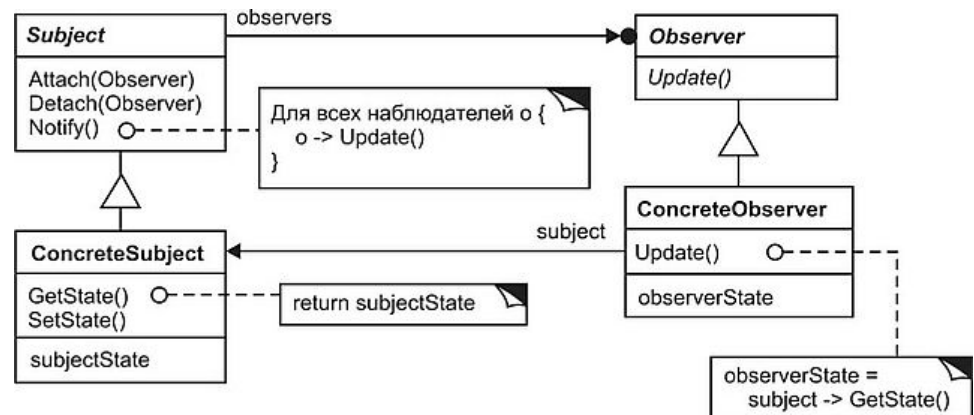


Рисунок 2 - UML-диаграмма наблюдателя

- Применение данного паттерна позволит реализовать механизм отправки сообщений в чате.
- #### 2. Фасад (Facade)
- Предоставляет бизнес-логике простой интерфейс для обращения к API синтеза и распознавания речи.
 - Структурный паттерн проектирования, который предоставляет простой интерфейс к сложной системе классов, библиотеке или фреймворку.

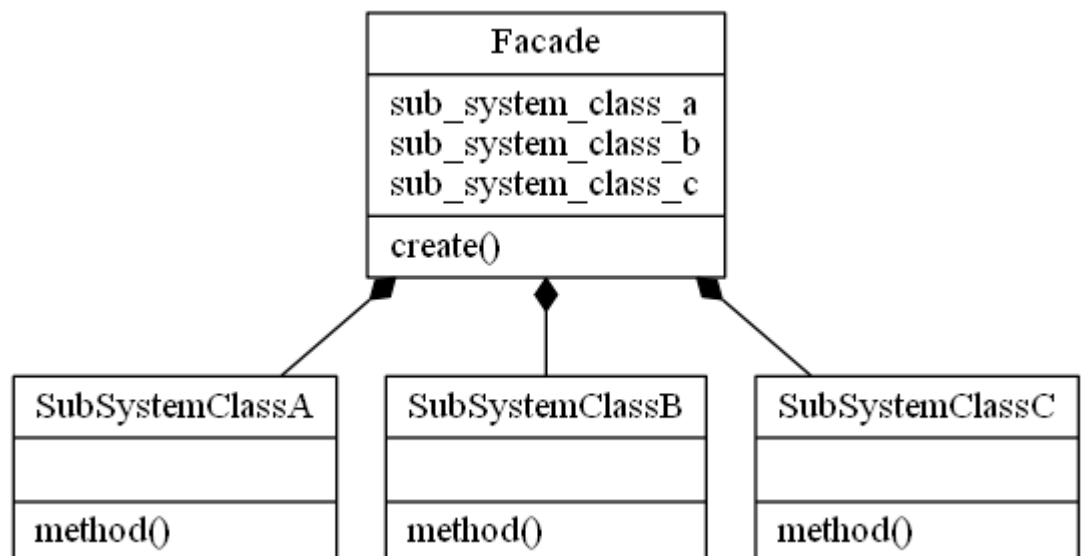


Рисунок 3 - UML-диаграмма фасада

- Применение данного паттерна позволит реализовать простое взаимодействие бизнес-логики и API.

4) Описание типа архитектуры

Для разработки использована микросервисная архитектура (рис. 4). Микросервисы будут обновляться независимо от веб-приложения и будут далее использоваться в других проектах. Кроме того приложение станет устойчивым к сбоям и будет возвращать ответ за корректное время, что соответствует надежности и удобству в использовании.

Отдельный сервер для базы данных выделен не будет.

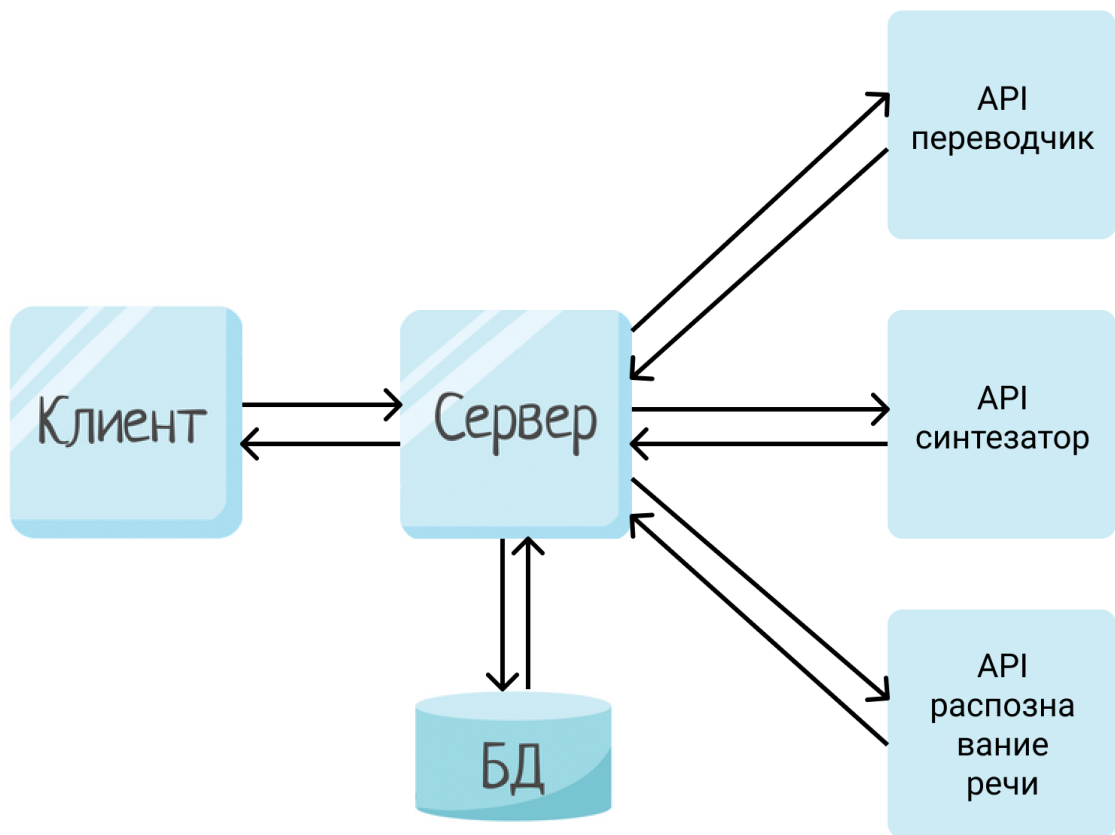
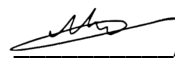


Рисунок 4 - архитектура приложения

Заказчик

Личная подпись  / Салех Х.М.

Ответственный по проекту

Личная подпись  / Малащенко Б.Т.

Дата:

16.01.2022