Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Вятский государственный университет»**

(ВятГУ)

**ОТЧЕТ**

**по Производственной практике,**

**ПРЕДДИПЛОМНОЙ ПРАКТИКЕ**

|  |  |  |
| --- | --- | --- |
|  | | |
| *(Ф.И.О. обучающегося)* | | |
| 02.03.02 Фундаментальная информатика и информационные технологии, направленность (профиль) 51 Разработка программного обеспечения | | |
| *(направление подготовки (специальность), направленность (профиль))* | | |
|  | | |
| Место прохождения практики | |  |
|  | *(наименование организации, структурного подразделения организации)* | |
|  | | |
| *(наименование организации, структурного подразделения организации)* | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Итоговая оценка: |  | | | | |
| Руководитель  практики от университета |  |  |  |  | Е. В. Разова |
|  | *(дата)* |  | *(подпись)* |  | *(Ф.И.О.)* |

Киров, 2025 г.

**СОДЕРЖАНИЕ**

[Введение 2](#_Toc199192247)

[Характеристика места прохождения практики 3](#_Toc199192248)

[Отчет о выполнении индивидуального задания 4](#_Toc199192249)

[- Постановка задания 4](#_Toc199192250)

[- Цель, задачи, которые необходимо решить для достижения поставленной цели 4](#_Toc199192251)

[- Обзор имеющихся подходов к решению 5](#_Toc199192252)

[- Обоснование выбора методов 8](#_Toc199192253)

[- Описание решения 9](#_Toc199192254)

[- Экспериментальное исследование качества реализованного решения 23](#_Toc199192255)

[- Результаты и выводы 24](#_Toc199192256)

[Заключение 26](#_Toc199192257)

[Библиографический список 27](#_Toc199192258)

[Приложение 28](#_Toc199192259)

## Введение

Производственная практика, преддипломная практика проходила в \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ с 28.04.2025 г. по 27.05.2025 г.

**Цель преддипломной практики** – закрепление и углубление полученных теоретических знаний, приобретение практических навыков и компетенций, подготовка к профессиональной деятельности, в основном путём самостоятельного решения задач, сформулированных в соответствии с темой выпускной квалификационной работы.

**Задачи преддипломной практики:**

* профессиональная ориентация, формирование полного представления о своей профессии;
* решение прикладных задач в области применения современных математических методов и информационных технологий, сбор материала по теме выпускной квалификационной работы;
* получение практических навыков и компетенций в организации работы в области применения современных математических методов и информационных технологий;
* выполнение индивидуального задания.

Индивидуальное задание на период прохождения производственной практики – разработка чат-бота в поддержку абитуриента ВятГУ.

## Характеристика места прохождения практики

Производственная практика, преддипломная практика проходила на базе общества с ограниченной ответственностью «АЛЬТ СТУДИЯ», г. Киров

Описание рабочего места (используемое аппаратное обеспечение):

– персональный компьютер с 16 гигабайтами оперативной памяти и процессором Intel(R) Core(TM) i3-10100F (CPU @ 3.60GHz, 3600 МГц, ядер: 4, логических процессоров: 8)

Программное обеспечение, используемое в работе:

* Docker;
* VS code.

## Отчет о выполнении индивидуального задания

# - Постановка задания

Современные абитуриенты сталкиваются с трудностями при поиске и анализе информации, связанной с поступлением в высшие учебные заведения, включая ВятГУ. При этом наблюдается высокая вовлечённость молодежи в цифровые коммуникации, в частности в использование мессенджера Telegram, ежедневная аудитория которого превышает 60 миллионов пользователей в России. Однако в настоящий момент Вятский государственный университет не предоставляет специализированного цифрового инструмента в формате Telegram-бота, предназначенного для информационной поддержки абитуриентов.

Таким образом, возникает необходимость разработки Telegram-бота, обеспечивающего:

* структурированный и удобный доступ к актуальной справочной информации для абитуриентов;
* автоматическую рекомендацию образовательных программ на основе предпочтений пользователя и введённых экзаменов;
* взаимодействие в формате диалога, привычного целевой аудитории.

# - Цель, задачи, которые необходимо решить для достижения поставленной цели

**Цель работы** – разработка Telegram-бота для предоставления справочной информации и рекомендаций абитуриентам ВятГУ на основе данных, размещенных на официальных источниках ВятГУ.

Для реализации поставленной цели необходимо решить следующие **задачи**:

1. прохождение инструктажа по ознакомлению с правилами внутреннего трудового распорядка, охраны труда, безопасности, санитарно-эпидемиологическими правилами и гигиеническими нормативами, а также вводный инструктаж на рабочем месте;
2. определение конкретных задач практики в разрезе выполнения индивидуального плана;
3. постановка и подготовка задач, разработка чат-бота в поддержку абитуриента ВятГУ;
4. проектирование, сбор и предварительная обработка исходных данных;
5. разработка чат-бота в поддержку абитуриента ВятГУ: выбор архитектуры приложения, создание UML-моделей;
6. разработка и реализация алгоритмов, тестирование ПО;
7. экспериментальное исследование предметной области, сбор данных;
8. разработка базы данных для исследования, проведение расчетов, обработка и анализ полученных результатов, формирование выводов;
9. подготовка и оформление отчета по практике;
10. подготовка к защите и защита отчета по практике;

# - Обзор имеющихся подходов к решению

**Admissions KFU [1]**

Официальный бот для иностранных абитуриентов Казанского (Приволжского) федерального университета (КФУ). Предназначен для повышения информированности абитуриентов из-за рубежа. Бот позволяет подписываться на интересующие программы подготовки, присылает уведомления о сроках приёма документов и начале вступительных испытаний. С его помощью иностранный студент может узнать о программах обучения, конкурсах, минимальных баллах и необходимых документах. Поддерживает русский, английский и испанский языки.

Этот бот не имеет возможности задания вопроса в свободной форме и не имеет функциональности рекомендации какого-либо направления вуза. Как было сказано ранее, бот предназначен для студентов из-за рубежа.

**Московский политех [3]**

Бот Московского политехнического университета. Позволяет задавать вопросы в свободной форме. Если не находит ответ в базе направляет на оператора.

Не ясно, что в целом умеет бот. По всей видимости, он предусматривает только задание вопроса в свободной форме.

**Хочу в Политех [4]**

Бот университета СПбГМТУ (Санкт-Петербургский государственный морской технический университет) представляет собой информационный сервис для абитуриентов. Позволяет открывать нормативные документы не выходя из телеграмма через мини приложения, cодержит информацию для иностранных студентов. Предоставляет сведения о приёмной кампании, нормативных документах, мероприятиях и контактах приёмной комиссии. Стоит отметить, что к большинству ответов приложена ссылка, чтобы проверить актуальность данных, либо в случае документа дата вступления в силу. Бот хорошо структурирован и позволяет выбрать год планируемого поступления.

Сочетание Inline и обычной клавиатуры заставляет пользователя часто переключаться между ними, что не очень удобно. Картинки занимают много места, но не несут никакой пользы. Также отсутствует раздел с рекомендациями.

**Составление требований**

На основе преимуществ и недостатков рассмотренных решений и на основе требований заказчика были составлены следующие требования к боту.

**Пользовательские требования**

- Пользователь должен иметь возможность получить актуальную информацию о направлениях подготовки, где возможно требуется прикрепить ссылку на официальный источник;

- требуется хорошо структурировать меню бота для наилучшего пользовательского опыта;

- пользователь должен иметь возможность получить рекомендации на основе интересов, уровня ЕГЭ и других параметров;

- бот должен уметь принимать вопросы в свободной форме и отвечать на основе базы знаний.

**Бизнес-правила**

- информация должна поступать только из официальных источников;

- основная цель бота – помочь абитуриенту с вопросами и с определение направления подготовки;

- бот не должен сообщать сведения, не подтвержденные в официальных источниках.

**Функциональные требования**

Структурированное меню должно включать:

- пункт частые вопросы, разбитые по категориям;

- пункт рекомендации направления.

Обрабатывая естественный язык, система должна давать ответ из базы знаний, при отсутствии подходящего ответа требуется предложить переформулировать запрос

Рекомендательная система должна:

- включать выбор интересов или предметов ЕГЭ;

- давать возможность сформулировать вопрос в свободной форме без выборов ЕГЭ и интересов;

- уметь подбирать подходящих направлений.

**Функциональные требования**

Система должна справляться с опечатками пользователя.

**Технические требования**

- Использовать очередь для соблюдения лимитов на отправку сообщений;

- соблюдать ограничения на сообщения;

- использовать сервисы для кеширования запросов к базе данных.

**Требования к интерфейсам**

- Бот должен использовать inline кнопки там, где это возможно;

- интерфейс должен быть интуитивно понятным и единообразным. Он не должен требовать специальных технических знаний;

- ответы должны быть краткими и адаптированными под мобильные устройства;

- интерфейс должен быть спроектирован так, чтобы его можно было легко дополнить новыми функциями.

**Требования к данным**

Данные должны быть нормализованы и хранится в СУБД. База данных должна содержать следующую информацию:

- пользователи. Необходимо хранить ID пользователя для отправки сообщений;

- категории и вопросы. База данных должна хранить в себе дерево категорий и вопросы, которые относятся к той или иной категории;

- образовательные программы. Требуется хранить основные сведения (название, ссылка, уровень образования, форма и срок обучения, описание и информация о карьере);

- экзамены и предметы. Необходимо хранить список вступительных испытаний для каждой программы с указанием типа конкурсной программы, предметов и того, являются ли они обязательными.

- синонимы для предметов вступительных испытаний. Для реализации сервиса исправления опечаток в предметах, требуется составить словарь синонимов для каждого предмета;

- проходные и средние баллы. База данных должна хранить данные конкурсной статистики, учитывая основания поступления (бюджет, платное и т.д.);

- важные даты процесса поступления. Требуется хранить ключевые точки для возможности уведомлять пользователя.

# - Обоснование выбора методов

В результате сравнительного анализа и практических требований к проекту Telegram-бота были приняты следующие ключевые технические решения:

- язык программирования – Python. Благодаря огромному сообществу (25,35 % по TIOBE, май 2025), удобству, встроенной поддержке асинхронности, а также обширному набору специализированных библиотек и фреймворков. Python обеспечивает высокую скорость разработки, простоту сопровождения и лёгкую интеграцию решений на основе машинного обучения;

- фреймворк – aiogram. Выбран за асинхронность, встроенную поддержку FSM и middleware, регулярные обновления и расширение через aiogram\_dialog. Он позволяет строить сложные сценарии диалога с управлением состояния;

- основаная СУБД для хранения данных – **PostgreSQL.** Объектно-реляционная СУБД с активным сообществом, продвинутыми возможностями параллелизма и расширений, стабильной производительностью и высоким рейтингом (4-е место DB-Engines, 49% использования по опросу StackOverflow 2024). PostgreSQL обеспечивает надёжное хранение структурированных данных и лёгкую интеграцию с ORM;

- векторная база данных – Qdrant. Выбрана за готовый Docker-образ, поддержку фильтрации по метаданным, высокую производительность и открытую лицензию Apache 2.0. Qdrant позволяет эффективно решать задачу семантического поиска по векторным представлениям текстов. Кроме того, она хорошо интегрируется с Python;

- кэш – **Redis.** Опытный и зрелый проект (с 2009 г.), лидирует среди key-value хранилищ (1-е место DB-Engines), предлагает богатый набор структур данных, встроенный Pub/Sub, TTL и механизмы персистентности. Redis обеспечивает достаточно быстрое кэширование и высокую отказоустойчивость;

- контейнеризация с помощью Docker. Docker гарантирует единообразие окружения, изоляцию зависимостей и ускоряет развертывание всех сервисов (бот, PostgreSQL, Redis, Qdrant, сервис поиска по базе данных, тестовая база данных, CI контейнер). Интеграция с GitHub Actions позволяет автоматически проверять корректность сборки и выполнения тестов при каждой отправке в репозиторий.

В совокупности, выбранные решения обеспечивают оптимальный баланс между быстротой разработки, надёжностью, производительностью и возможностями масштабирования проекта в долгосрочной перспективе.

# - Описание решения

Архитектура разработана с учётом требований надёжности, масштабируемости и гибкости. Проект реализован в виде набора изолированных микросервисов, каждый из которых отвечает за отдельный участок функциональности, что потенциально должно облегчить сопровождение и развитие системы в будущем.

Далее компоненты, из которых состоит система.

Бот-сервис – основной модуль, реализующий взаимодействие с пользователями. Он отвечает за обработку сообщений, запуск диалогов, отображение интерфейсов, кроме того, именно он отвечает за базу данных (в нем находятся миграции), так как все остальные сервисы направлены на взаимодействие с ним.

Redis. Используется в качестве промежуточного слоя. Предназначен для создания trottling middleware, которые ограничивают спам атаки от пользователя, и для кеширования часто используемых данных с целью повышения скорости и снижения нагрузки на основную СУБД.

Сервис векторного поиска. Отдельный микросервис, реализующий семантический поиск по базе часто задаваемых вопросов. Он использует Qdrant в связке с моделью sentence-transformers для хранения и поиска. Выделение этой функциональности в отдельный сервис позволяет разгрузить основной бот и повысить устойчивость всей системы за счёт изоляции ресурсоёмких операций. Кроме того, такая архитектура упрощает масштабирование и предоставляет возможности для последующего расширения. На данном этапе с ним взаимодействует только бот сервис через FastAPI.

PostgreSQL. Сюда загружается вся нужная информация об образовательных программах. Структура управляется с помощью alembic из основного сервиса бота. Доступ к данным осуществляется через ORM SQLAlchemy

Qdrant. Векторное хранилище данных. Обслуживает сервис векторного поиска. Позволяет выполнить быстрый поиск по embedding-представлениям вопросов.

Контейнер запуска тестов. Используется для автоматического запуска тестов. Включает в себя копии исходных файлов и связан с отдельной тестовой базой данных, обеспечивая изоляцию рабочей среды.

Тестовая PostgreSQL. Используется исключительно для тестирования, для изоляции рабочей среды.

Контейнер миграции базы данных. Специальный сервис только для выполнения миграций базы данных. Используется для автоматизации развертывания и принципа разделения ответственности.

Рассмотрим рисунок, наглядно демонстрирующий взаимодействие компонентов между собой.

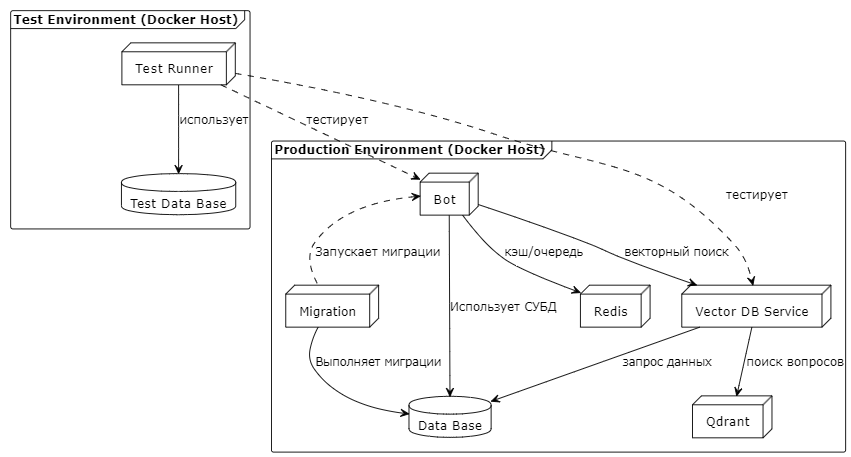


Рисунок 1 – Процесс развертывания.

Подробнее разберем процесс развертывания, изображенный на рисунке выше:

- вначале разворачиваются независимые контейнеры: Data Base, Qdarant, Redis, Test Data Base;

- после развертывании контейнер Migration запускает миграционные скрипты и выполняет их над Data Base;

- одновременно с этим Test Runner запускает тесты;

- запускается Vector DB Service и сразу же обращается к базе данных для обновления Qdaran;

- в конце запускается Bot, который так или иначе используется все компоненты в Prodaсtion окружении.

Такой поток взаимодействия обеспечивает четкое разграничение ответственности и надежное функционирование всей системы. Кроме того, процесс автоматический.

Для реализации целевого сервиса-бота была выбрана чистая архитектура. В основе этой архитектуры лежит правило зависимостей, которое гласит, что внутренние слои приложения не должны зависеть от реализаций внешних слоев, а должны определять интерфейсы-контракты, которые будут реализовывать внешние слои. Рассмотрим рисунок ниже.

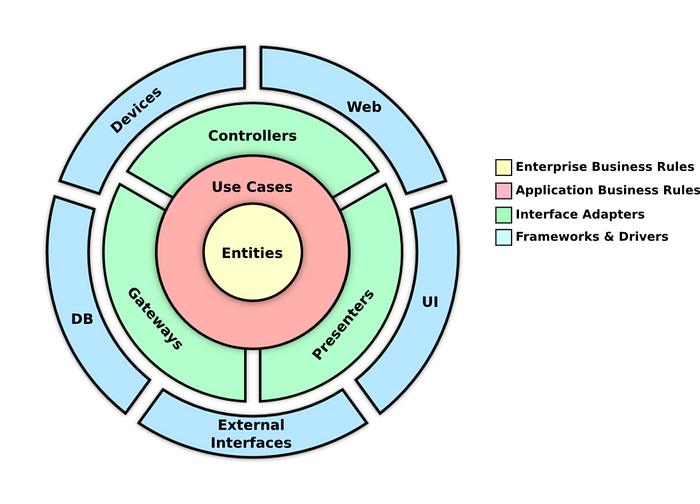


Рисунок 3.3 – Визуализации чистой архитектуры. Источник рисунка: [3].

В этой архитектуре выделяют 4 слоя:

- доменный слой (на рисунке Enterpice Busisness Rules). Он содержит бизнес-объекты и логику, которая не зависит от внешних систем и должна меняться;

- слой приложения (на рисунке Application Business Rules). В нем описываются бизнес-правила и сценарии использования. Он использует интерфейсы, а не конкретные реализации;

- слой представления (на рисунке Interface Adapters). Отвечает за взаимодействие с пользователем через интерфейс Telegram. В нем расположены интерфейсы репозиториев для взаимодействия с базой данных и интерфейсы других сервисов. Тут происходит внедрение зависимостей;

- инфраструктурный слой (на рисунке Frameworks & Drivers). Здесь расположены конкретные реализации для связи с внешними сервисами.

Благодаря строгому разделению ответственности и правилу зависимостей, бизнес-логика остаётся изолированной от внешних технологий, что позволяет легко заменять инфраструктурные компоненты, адаптироваться к новым требованиям и повторно использовать ядро приложения в других интерфейсах или проектах [5].

Теперь рассмотрим основные взаимодействия с пользователем. Ниже диаграмма использования, которая показывает, какие функции включены в бота.

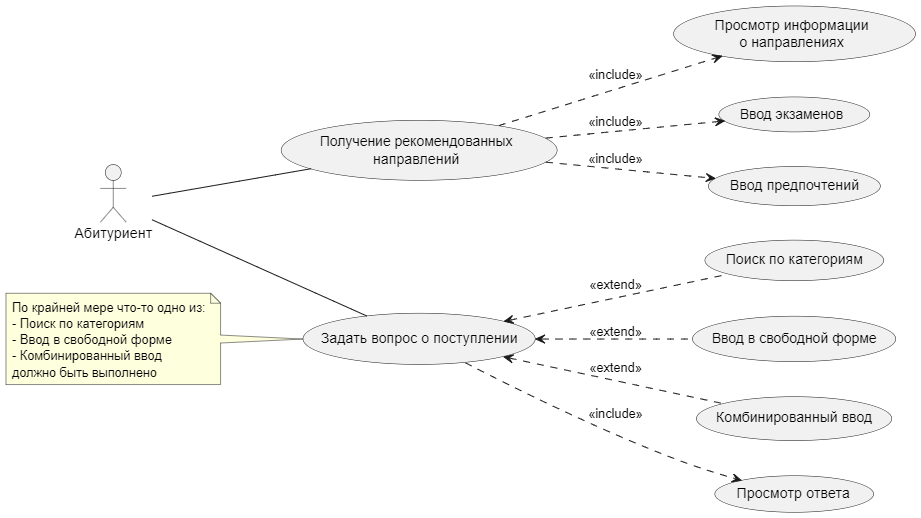


Рисунок 2 – Диаграмма использований бота.

**Вопрос о поступлении**

Пользователь может задать интересующий его вопрос о поступлении двумя основными способами: через навигацию по категориям либо во вводе в свободной форме. Это позволяет учитывать как формализованные, так и неструктурированные запросы.

Диаграмма активности на рисунке ниже демонстрирует, как пользователь может взаимодействовать с системой.

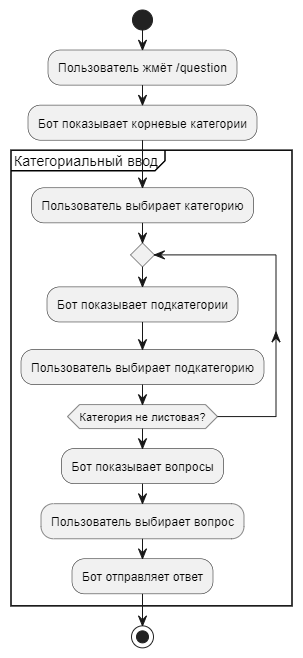


Рисунок 3 – Диаграмма активности (категориальный ввод).

Для рассмотрения процесса с точки зрения взаимодействия компонентов системы далее приведена диаграмма последовательности, иллюстрирующая сценарий выбора категорий пользователем. На ней продемонстрировано взаимодействие с хранилищем Redis и PostreSQL.

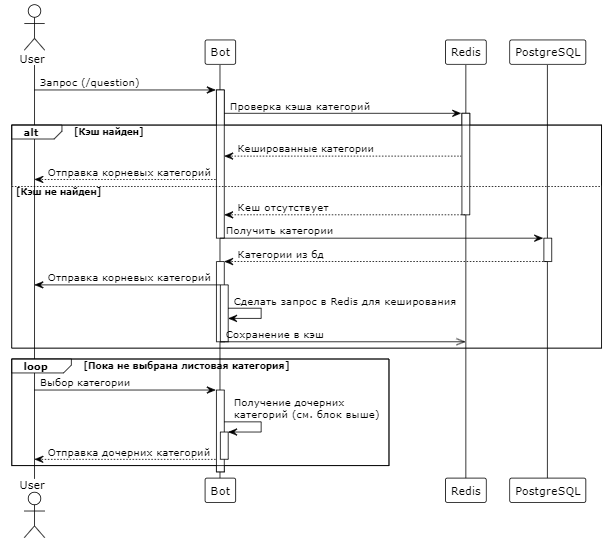


Рисунок 4 – Диаграмма последовательности (категориальный ввод).

Вариант с категорией предполагает пошаговое движение по иерархии тематических разделов. Но пользователь может не понять, в какую категорию нужно зайти, чтобы увидеть свой вопрос. Для решения этой проблемы в любой момент пользователь имеет возможность задать вопрос с клавиатуры в свободной форме. Ниже на рисунке диаграмма последовательности для этого сценария.

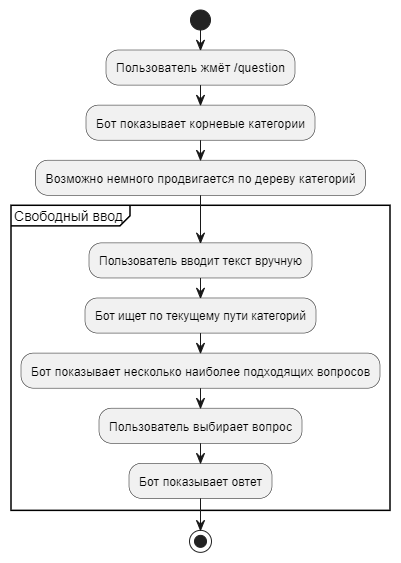


Рисунок 5 – Диаграмма последовательности (свободный ввод).

Также пользователь может нажать кнопку назад, чтобы перейти к предыдущему пункту.

Для рассмотрения процесса с точки зрения взаимодействия компонентов системы далее приведена диаграмма последовательности, иллюстрирующая сценарий задания вопроса пользователем в свободной форме. На ней продемонстрировано взаимодействие с хранилищем Vector DB Service и Qdrant.

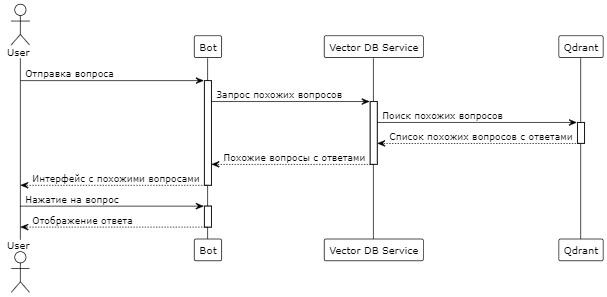


Рисунок 6 – Диаграмма последовательности (свободный ввод).

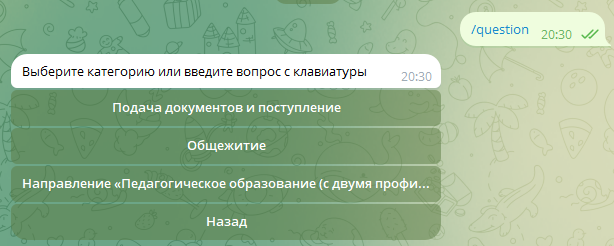
Для лучшего понимания логики взаимодействия с пользователем в мессенджере Telegram ниже представлены скриншоты, иллюстрирующие пошаговую работу сценария «Вопрос о поступлении». 

Рисунок 7 – Начало диалога задания вопроса боту.

Пользователь нажал на кнопку «Подача документов и поступление». Бот изменил меню, как показано на рисунке ниже

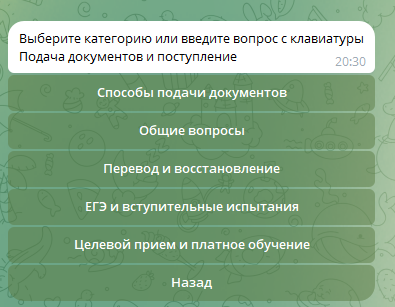


Рисунок 8 – Категории вопросов.

Пользователь ввел запрос с клавиатуры «Бюджетные места в ВятГУ» и отправил сообщение. Бот изменил меню, как показано на рисунке ниже

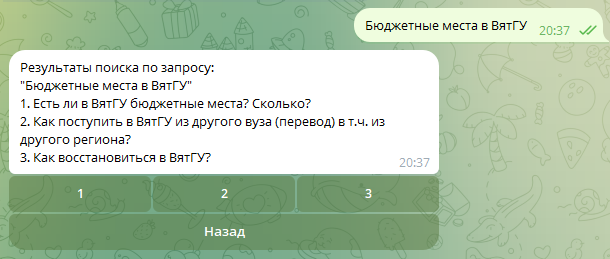


Рисунок 9 – Окно вопросов.

Пользователь нажал на кнопку «1». Бот отправил сообщение, как на рисунке ниже.

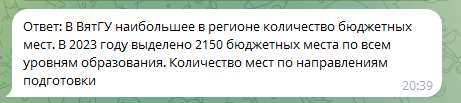


Рисунок 10 – Сообщение с ответом на вопрос.

**Получение рекомендованных направлений**

Данный сценарий начинается с команды /recommend, после чего бот запускает пошаговый процесс сбора информации об экзаменах и предпочтениях пользователя. На рисунке ниже диаграмма активности для этого сценария.

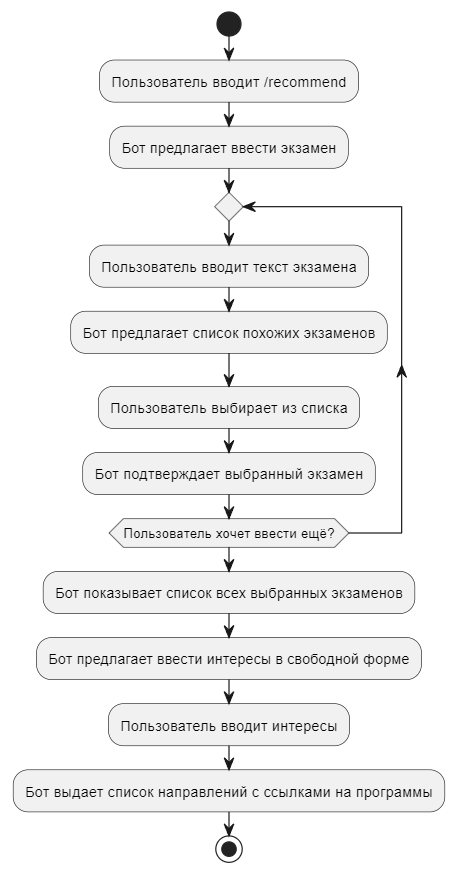


Рисунок 11 – Диаграмма активности сценария получения рекомендованных направлений.

Для рассмотрения процесса с точки зрения взаимодействия компонентов системы далее приведена диаграмма последовательности, иллюстрирующая выборы экзаменов для сценария получения рекомендаций по направлениям. На ней продемонстрировано взаимодействие с сервисом исправления опечаток в экзаменах и базой данных.

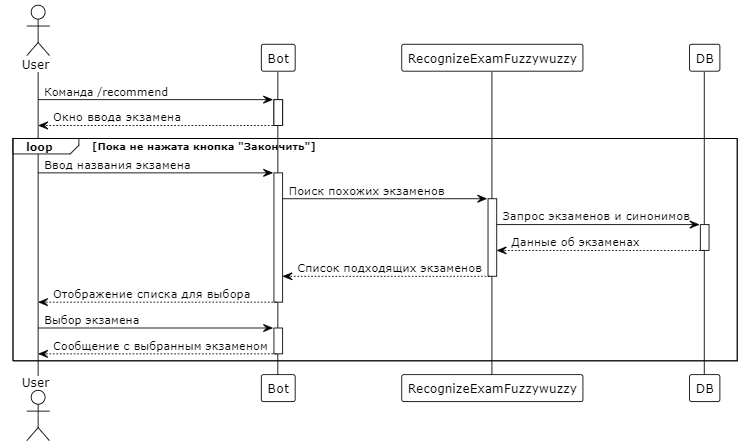


Рисунок 12 – Диаграмма последовательности выбора экзаменов.

Для лучшего понимания логики взаимодействия с пользователем в мессенджере Telegram ниже представлены скриншоты, иллюстрирующие пошаговую работу сценария «Получение рекомендованных направлений».

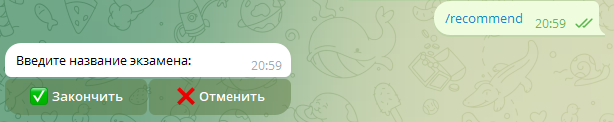


Рисунок 13 – Начало диалога рекомендации направлений.

Пользователь ввел на клавиатуре «математика», бот отправил интерфейс, как на рисунке ниже.

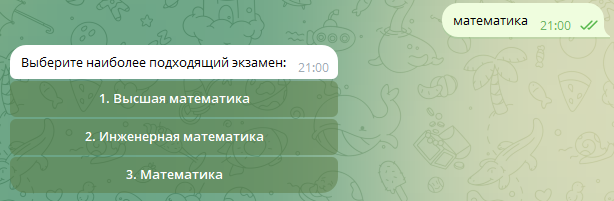


Рисунок 14 – Меню уточнение экзамена.

Пользователь нажал на кнопку «3. Математика», бот отправил сообщение, что экзамен добавлен и предложил ввести новый экзамен.

Пользователь ввел подобным образом экзамены: информатика русский язык – и нажал на закончить. Бот отправил экзамены, которые выбрал пользователь и предложение ввести интересы в свободной форме (рисунок ниже).

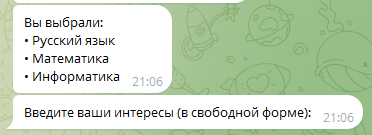


Рисунок 15 – Сообщение с выбранными экзаменами и предложением ввести интересы.

Пользователь ввел сообщение «Я люблю программировать и хотел бы стать разработчиком по», бот отправил меню, как на рисунке ниже.

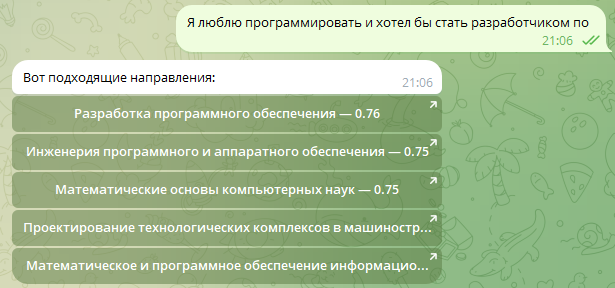


Рисунок 16 – Меню с наиболее подходящими направлениями подготовки.

Пользователь нажал на первую кнопку. Telegram вывел меню с предложением открыть ссылку на указанную программу (рисунок ниже).

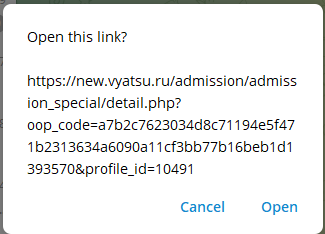


Рисунок 17 – Предложение перейти по ссылке.

# - Экспериментальное исследование качества реализованного решения

Разработка приложения велась с соблюдением чистой архитектуры [5], что позволило легко протестировать компоненты независимо друг от друга.

В рамках проекта были проведены следующие виды тестирования:

- **модульное**. Включало проверку отдельных компонентов (например, сервис исправления опечаток) в изоляции;

- **интеграционное**. Включало проверку взаимодействия между модулями, включая бизнес-логику и базу данных;

- **функциональное. Включало** проверку пользовательских сценариев (задание вопроса, просмотр ответов, подбор направлений).

- **тестирование интерфейса. Включало** проверку корректности отображения сообщений, inline-кнопок и реакций Telegram-бота в различных клиентах;

- **кросс-платформенное. Включало** проверку приложения в мобильных и десктопных версиях Telegram.

В процессе тестирования использовались следующие инструменты:

- pytest для модульных тестов;

- telegram desktop и telegram mobile для ручного тестирования

- фикстуры для изоляции завсимостей;

- docker для воспроизведения окружения.

Кроме того, было настроено автоматическое выполнение тестов с помощью GitHub Actions, что предупреждает разработчика о необходимости исправить код при ошибках в тестовом контейнере.

# - Результаты и выводы

В рамках данной работы был спроектирован и реализован Telegram-бот, предназначенный для информационной поддержки абитуриентов Вятского государственного университета. Основной целью проекта являлось создание удобного, доступного и функционального инструмента, способного отвечать на часто задаваемые вопросы о поступлении, а также помогать абитуриентам с выбором образовательных программ на основе их экзаменов и интересов.

Архитектура приложения построена в соответствии с принципами чистой архитектуры, что обеспечивает высокую модульность, тестируемость и гибкость решения. Были проведены различные виды тестирования, включая модульное, интеграционное и функциональное, а также настроено автоматическое модульное тестирование с использованием GitHub Actions для повышения надёжности разработки.

Проект полностью контейнеризирован с использованием Docker, что позволяет быстро и удобно развернуть приложение в любом окружении. Использование docker-compose обеспечивает воспроизводимость среды и облегчает настройку всех необходимых для работы сервисов.

Система поддерживает как ручной, так и интеллектуальный ввод вопросов. Она может масштабироваться и дополняться новыми функциями без существенного изменения архитектуры.

Результаты работы подтверждают, что использование современных подходов к проектированию программных систем, таких как чистая архитектура и контейнеризация, способствует созданию устойчивых и развиваемых решений, способных эффективно решать задачи взаимодействия с пользователем в образовательной сфере.

## Заключение

Производственная практика способствовала формированию компетенций, необходимых для самостоятельного решения производственных задач в области разработки программного обеспечения, а также подготовке и сбору материала и апробации решений задач, поставленных в рамках ВКР.

В ходе практики был осуществлен поиск и отбор источников по теме исследования. Для этого осуществлялась работа со следующими электронными информационными и библиотечными ресурсами, системами цитирования, базами научных публикаций:

– Google Scholar (scholar.google.com);

– eLIBRARY.RU – российская научная электронная библиотека;

– КиберЛенинка (cyberleninka.ru).

В ходе работы было отобрано по теме исследования 5 источников, из них:

– 1 монография;

– 4 источника на русском языке, 1 источник на иностранных языках;

Уникальность текста обзора составляет 91 % по результатам проверки с помощью системы обнаружения заимствований Text.ru.

Наиболее интересным при проведении исследования оказалось попробовать на практике использование технологий Docker и Redis. Развёртывание микросервисов в контейнерах, настройка взаимодействия между сервисами и интеграция Redis в качестве инструмента кэширования и очередей сообщений позволили получить ценный практический опыт работы с современными инструментами разработки.

Наиболее сложным при проведении исследования оказалось оформление отчёта по практике: требовалось тщательно структурировать материал, корректно оформить все разделы в соответствии с методическими требованиями и отразить как технические детали, так и теоретические аспекты проделанной работы.

## Библиографический список

1. KPFU Admissions Bot [Электронный ресурс]. URL: <https://t.me/kpfu_admissions_bot> (дата обращения: 26.05.2025).
2. Вятский государственный университет: официальный сайт [Электронный ресурс]. URL: <https://new.vyatsu.ru/> (дата обращения: 26.05.2025).
3. Moscow Polytech Bot [Электронный ресурс]. URL: <https://t.me/MoscowPolytechBot> (дата обращения: 26.05.2025).
4. SPbPU Bot [Электронный ресурс]. URL: <https://t.me/SPbPUbot> (дата обращения: 26.05.2025).
5. Martin R. C. Clean architecture [Электронный ресурс].

## Приложение

Ссылка на репозиторий https://github.com/DimaOshchepkov/vyatsu\_applicant\_bot