

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»  
Профиль подготовки: «Программное обеспечение средств вычислительной  
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

**ТЕЛЕГРАММ-БОТ «ФИТНЕС-МАГАЗИН»**

Студент

 09.06.2025  
(подпись, дата)

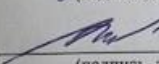
Чинов Д.Д.  
(ФИО полностью)

Руководитель

 09.06.2025  
(подпись, дата)

Егоров В.Ю.  
(фамилия, инициалы)

Нормоконтролёр

 09.06.2025  
(подпись, дата)

Бычков А.С.  
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 28.05.25 г. № 16 )

Заведующий кафедрой

  
(подпись)

М.А. Митрохин  
(фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от  
16 июня 2025 г. № 34 )

Секретарь ГЭК

  
(подпись)

Е.И. Гурин  
(фамилия, инициалы)

Пенза 2025

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет  
вычислительной техники

Кафедра  
«Вычислительная техника»

«Утверждаю»  
Заведующий кафедрой  
М.А. Митрохин  
« 06 » мая 2025 г.

**ЗАДАНИЕ  
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Чинов Даниил Дмитриевич гр.21ВВП2 факультета вычислительной техники

2. Тема работы Телеграмм-бот «Фитнес-магазин»

Тема утверждена приказом ПГУ № 601/о от "06" мая 2025г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Егоров В.Ю.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать Телеграмм-бот «Фитнес-магазин» для покупки фитнес товаров. Бот должен позволять пользователю выполнять просмотр товаров, которые есть в магазине. Клиентская часть программы должна обладать понятным интерфейсом. Также клиентская программа должна позволять пользователю добавлять товар в корзину, смотреть баланс, корзину и историю покупок. Сервер должен включать в себя сохранение и передачу данных в базу данных. Язык программирования C#, среда разработки Microsoft Visual Studio. В качестве системы управления базами данных использовать PostgreSQL.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области: изучить существующие аналоги, изучить способы проектирования и разработки телеграмм-бота. Спроектировать архитектуру приложения, реализовать серверную и клиентскую части, настроить взаимодействие с базой данных. Выбрать язык программирования, среду разработки и базу данных.

6. Календарный график выполнения работы

| № п/п | Наименование этапов работы                                    | Объем работы | Срок выполнения       | Подпись руководителя                                                                |
|-------|---------------------------------------------------------------|--------------|-----------------------|-------------------------------------------------------------------------------------|
| 1     | Изучение предметной области                                   | 40           | 06.05.2025-10.05.2025 |  |
| 2     | Анализ требований                                             | 16           | 11.05.2025-12.05.2025 |  |
| 3     | Выбор технологий, применяемых при проектировании и разработке | 24           | 13.05.2025-15.05.2025 |  |
| 4     | Описание алгоритма                                            | 24           | 16.05.2025-18.05.2025 |  |
| 5     | Проектирование и Разработка ПО                                | 80           | 19.05.2025-28.05.2025 |  |
| 6     | Оформление выпускной работы                                   | 16           | 29.05.2025-30.05.2025 |  |

Дата выдачи задания "06" мая 2025г.

Руководитель бакалаврской работы  06.05.2025 Егоров В.Ю.  
(подпись, дата)

Задание к исполнению принял студент  06.05.2025 Чинов Д.Д.  
(подпись, дата)

Работу к защите допустить  
Декан факультета  16.06.2025 Фионова Л.Р.  
(подпись, дата)

## Реферат

Бакалаврская работа содержит 80 листов, 24 рисунка, 8 источников, 3 приложения

БОТ, ТЕЛЕГРАММ, МАГАЗИН, ФИТНЕС, БЗА ДАННЫХ, C#, SQL, POSTGRESQL

Цель работы – реализация бота для Telegram.

Объектом разработки является бот для заказа фитнес товаров.

В результате проделанной работы был реализован бот для системы Telegram, через который можно купить нужные товары для тренировок в зале. Разработан интуитивный каталог с товарами, разработана удобная корзина и история сделок. Разработка производилась с использованием масштабного стека технологий, к которым относятся: язык программирования C#, язык структурированных запросов SQL. Для хранения данных выбрана СУБД PostgreSQL, а для её администрирования применялся инструмент PGAdmin4.

|          |      |               |                                                                                     |          |                            |      |        |
|----------|------|---------------|-------------------------------------------------------------------------------------|----------|----------------------------|------|--------|
|          |      |               |                                                                                     |          | ПГУ1.090301.49.001 ПЗ      |      |        |
| Изм.     | Лист | № докум.      | Подпись                                                                             | Дата     |                            |      |        |
| Разраб.  |      | Чинюв Д.Д.    |  | 29.05.25 |                            |      |        |
| Провер.  |      | Егоров В.Ю.   |  | 29.05.25 |                            |      |        |
| Н. Контр |      | Бычков А.С.   |  | 29.05.25 |                            |      |        |
| Утверд.  |      | Митрохин М.А. |  | 29.05.25 |                            |      |        |
|          |      |               |                                                                                     |          | Пояснительная записка      |      |        |
|          |      |               |                                                                                     |          | Лит.                       | Лист | Листов |
|          |      |               |                                                                                     |          |                            | 2    | 80     |
|          |      |               |                                                                                     |          | ФВТ, каф. ВТ<br>гр. 21ВВП2 |      |        |

## Содержание

|                                                   |    |
|---------------------------------------------------|----|
| Реферат .....                                     | 2  |
| Содержание .....                                  | 3  |
| Перечень сокращений и обозначений .....           | 4  |
| Введение .....                                    | 5  |
| 1 Анализ предметной области.....                  | 6  |
| 1.1 Анализ темы.....                              | 6  |
| 1.2 Анализ аналогов .....                         | 7  |
| 1.3 Постановка задачи.....                        | 12 |
| 2 Выбор инструментов разработки.....              | 14 |
| 2.1 C# .....                                      | 14 |
| 2.2 PostgreSQL.....                               | 15 |
| 2.3 Rest.....                                     | 16 |
| 3 Архитектура .....                               | 17 |
| 3.1 Структура БД .....                            | 22 |
| 4 Тестирование .....                              | 25 |
| 5 Безопасность жизнедеятельности.....             | 33 |
| 5.1 Требование к помещению и рабочему месту. .... | 33 |
| 5.2 Требование к освещению. ....                  | 34 |
| 5.3 Требования к режиму труда и отдыха.....       | 36 |
| Заключение.....                                   | 38 |
| Список использованных источников .....            | 39 |
| Приложение А – UML-диаграммы .....                | 40 |
| Приложение Б – Листинг программы .....            | 45 |
| Приложение В – Презентация.....                   | 75 |

## Перечень сокращений и обозначений

В данной работе применяются следующие сокращения и обозначения

БД – база данных

ПК – персональный компьютер

Спортпит – спортивное питание

СУБД – система управления базами данных

IDE (от англ. Integrated Development Environment) – интегрированная среда разработки

SQL (англ. Structured Query Language) – язык структурированных запросов

UML (от англ. Unified Modeling Language) – унифицированный язык моделирования

|      |      |          |         |      |                              |      |
|------|------|----------|---------|------|------------------------------|------|
|      |      |          |         |      | <i>ПГУ1.090301.49.001 ПЗ</i> | Лист |
|      |      |          |         |      |                              | 4    |
| Изм. | Лист | № докум. | Подпись | Дата |                              |      |

## Введение

Современные технологии кардинально меняют подходы к ведению бизнеса, и одним из самых ярких примеров этого является использование мессенджеров для автоматизации взаимодействия с клиентами. В частности, Телеграм-боты становятся все более популярными, поскольку они предлагают удобный и эффективный способ общения и совершения покупок.

Для фитнес-магазинов такие боты могут стать настоящим прорывом. Они не только упрощают процесс заказа товаров, но и значительно улучшают клиентский опыт. С помощью Телеграм-ботов клиенты могут быстро получать информацию о наличии товаров, акциях и скидках, а также получать персонализированные рекомендации в зависимости от своих предпочтений и целей.

Кроме того, использование ботов позволяет оптимизировать бизнес-процессы. Например, автоматизация обработки заказов и запросов клиентов освобождает время сотрудников, позволяя им сосредоточиться на более важных задачах, таких как улучшение сервиса или работа с клиентами. В результате, фитнес-магазины могут не только повысить свою эффективность, но и укрепить отношения с клиентами, предлагая им более качественный и удобный сервис.

Таким образом, внедрение Телеграм-ботов в работу фитнес-магазинов — это не просто модный тренд, а необходимость для тех, кто хочет оставаться конкурентоспособным в быстро меняющемся мире.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.49.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 5    |

# 1 Анализ предметной области

Прежде всего нужно выяснить, кто наши пользователи и какие задачи они хотят решить с помощью приложения. Затем определите, в каких условиях и зачем они будут обращаться к вашему сервису. И только после этого переходите к продумыванию архитектуры – её продуманная реализация гарантирует стабильную работу, отвечает ожиданиям аудитории и позволяет заранее отловить возможные технические сложности.

## 1.1 Анализ темы

В наши дни тренировки в фитнес–зале перестали быть экзотикой и превратились в привычный элемент жизни миллионов людей. Яркая популяризация фитнеса, бодибилдинга, пауэрлифтинга и кроссфита в социальных сетях и масс– медиа подталкивает всё новых и новых поклонников к покупке абонементов в спортклубы. Кто–то ищет способ сбросить лишний вес, кто–то – прокачать выносливость, а кто–то просто хочет найти место, где можно перезагрузиться после рабочего дня. Поход в зал становится не разовым экспериментом, а регулярным ритуалом, который приучает следить за собой и дарит энергию на будни.

Регулярные занятия оказывают комплексное влияние на организм: улучшается кровообращение, активизируется обмен веществ, мышцы получают необходимую нагрузку, а тело – подтягивается и обретает рельеф. Со временем меняется не только внешний вид: тренировки помогают справляться со стрессом, укрепляют силу воли и дарят чувство личного достижения. У кого–то это перерастает в настойчивое стремление участвовать в соревнованиях, у кого–то – в желание прокачивать новые качества своего тела, а кто–то просто получает удовольствие от групповых занятий и поддерживающей атмосферы.

Параллельно с ростом числа посетителей залов усиливается спрос на всё, что помогает в тренировочном процессе. Потребители охотно покупают

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.49.001 ПЗ | Лист |
|      |      |          |         |      |                       | 6    |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

функциональную одежду и обувь, спортивные коврики, эспандеры и ремни, а также вкладываются в домашние тренажёры. Не меньшим спросом пользуются пищевые добавки: протеиновые коктейли, аминокислоты, витаминные комплексы и специальные смеси для восстановления после нагрузок. Каждая такая покупка – это шаг навстречу более эффективным тренировкам и заботе о собственном здоровье.

Именно здесь на помощь приходит Телеграмм-бот «Фитнес-магазин». В отличие от традиционных сайтов и приложений, бот встроен в привычный мессенджер и готов к работе 24/7. Он мгновенно отвечает на запросы, показывает каталог товаров с фото и описаниями, позволяет фильтровать позиции по цене, категории и наличию на складе. Добавить товар в корзину можно одним нажатием кнопки, а форма оплаты в чате сводит весь процесс к секундам: подтвердил – оплатил – получил уведомление об отправке.

Пользователю не придётся перебирать десяток вкладок в браузере или ждать ответа оператора: бот хранит историю предыдущих заказов, напоминает о необходимости пополнить запасы протеина и уведомляет об акциях и скидках. Для владельцев магазина это решение – способ автоматизировать обработку заявок, снизить нагрузку на службу поддержки и ускорить выполнение заказов. В результате Телеграмм-бот «Фитнес-магазин» превращается в виртуального ассистента, дарящего комфорт и экономию времени как спортсменам-любителям, так и профессионалам.

## 1.2 Анализ аналогов

На данный момент аналогов нет. Были выбраны телеграмм-боты, в которых можно приобрести какие-либо услуги. Также в качестве примера был взят аналог сайта спортивного магазина. В данном пункте рассмотрим некоторые из них:

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.49.001 ПЗ | Лист |
|      |      |          |         |      |                       | 7    |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

- TuzShop;
- PAY2GAME;
- Спортмастер.

#### 1.2.1 Телеграмм–бот «TuzShop»

Интерфейс бота представлен на рисунках 1.1 и 1.2.



Рисунок 1.1 – Основное меню



Рисунок 1.2. – Меню товаров

Удобны бот для покупки игровой валюты.

Достоинства:

- удобство. У бота минималистичный интерфейс, который легок в использовании;
- рассылка. Можно включить и выключить рассылки, чтобы не отключать уведомления бота и всегда быть в курсе новостей и анонсов любой игры, и скидков магазина;
- отзывы. В один клик можно перейти и прочитать отзывы.

Из недостатков поддержка долго отвечает.

### 1.2.2 Телеграмм–бот «PAY2GAME»

Интерфейс бота представлен на рисунках 1.3 и 1.4.

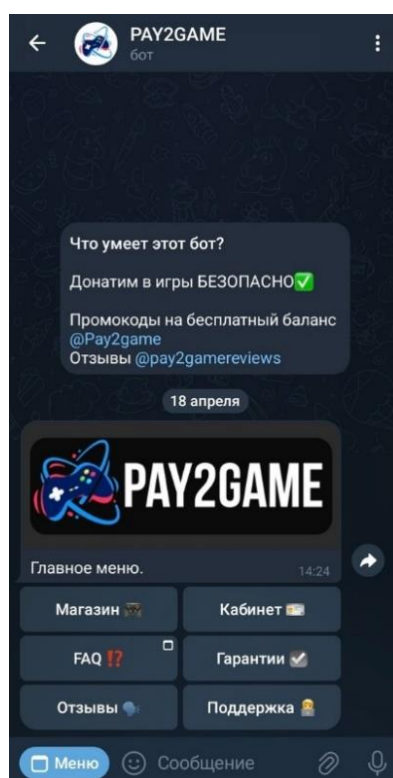


Рисунок 1.3 – Основное меню

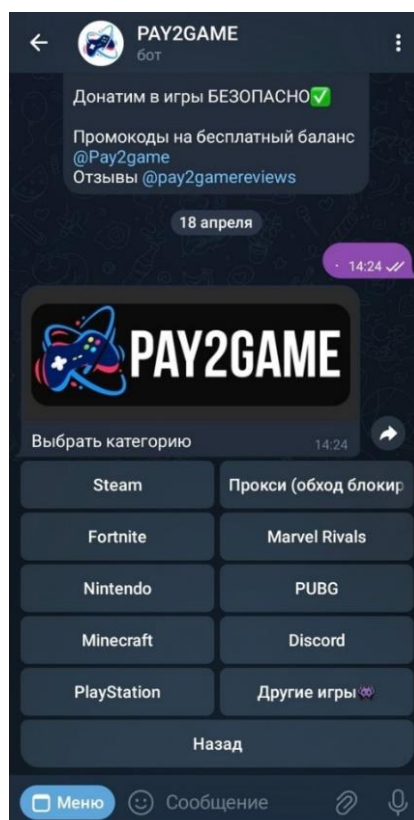


Рисунок 1.4 – Меню товаров

Данный бот удобен для покупки игр и игровой валюты.

Достоинства:

- сертификат. Можно приобрести подарочный сертификат для своего друга;
- витрина. Обширный выбор игр;
- баланс. Пополнить баланс можно любым удобным способом.

Недостатки рассылки. Нельзя отключить рассылку.

### 1.2.3 Веб–сайт «Спортмастер»

Интерфейс сайта представлен на рисунках 1.5 и 1.6.

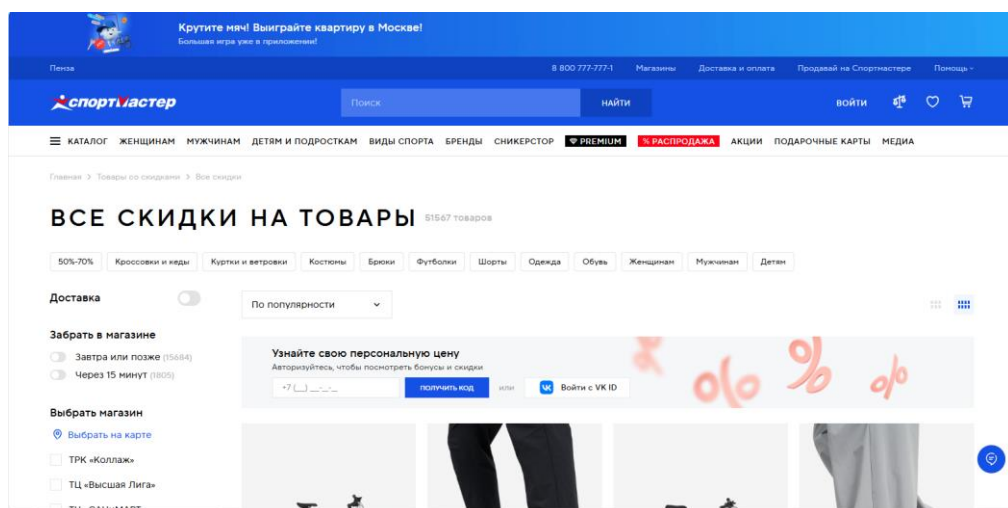


Рисунок 1.5 – Основное меню

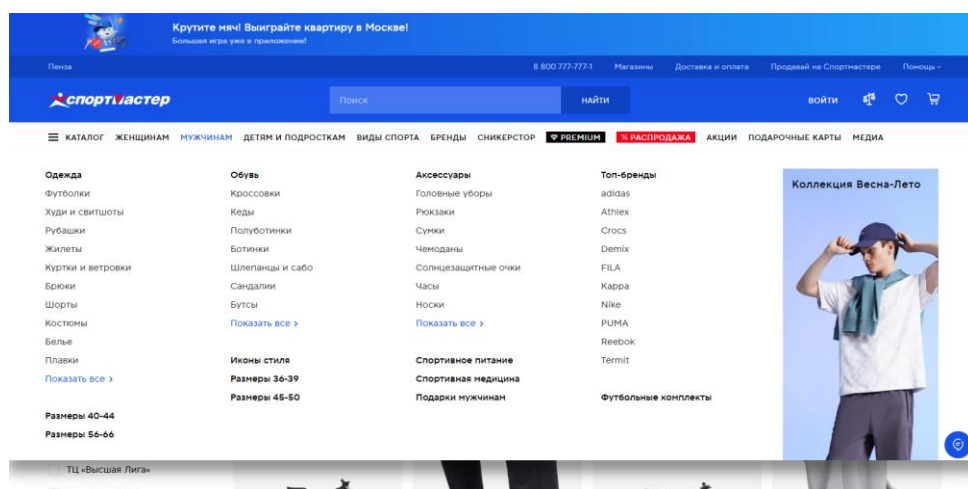


Рисунок 1.6 – Меню товаров

Сайт для покупки спортивных товаров.

Достоинства – бонусная система. При покупке вы получаете бонусы, которые можете потратить при следующем заказе.

Неудобства – стабильность. Сайт иногда подвисает.

Telegram–бот «Фитнес–магазин» выигрывает у классического сайта своей простотой и скоростью:

– всегда под рукой в привычном Telegram, без входа в аккаунт и загрузки приложений;

|      |      |          |         |      |
|------|------|----------|---------|------|
|      |      |          |         |      |
| Изм. | Лист | № докум. | Подпись | Дата |

ПГУ1.090301.49.001 ПЗ

Лист

11

- заказ в два клика, мгновенные уведомления о скидках и статусе доставки;
- персональные рекомендации на основе истории покупок и встроенные платежи без редиректов;
- автоматизация учёта товара, накладных и отчётов без участия менеджера.

В итоге вы получаете лёгкий, быстрый и персонализированный канал продаж прямо в чате.

### 1.3 Постановка задачи

В рамках проекта необходимо разработать Телеграмм-бота под рабочим названием SportShopBot, который станет удобным «магазином в чате» для пользователей, увлекающихся спортом и здоровым образом жизни. Бот должен обеспечить полный цикл онлайн-покупки: от регистрации до просмотра истории заказов.

#### 1.3.1 Регистрация пользователя

При первом запуске SportShopBot попросит нажать на кнопку СТАРТ или прописать /start, тем самым пользователь автоматически зарегистрируется, через встроенный механизм Telegram-авторизации – это позволит без проблем связать профиль с чатом и сразу начать покупки.

#### 1.3.2. Витрина товаров

Бот должен отображать каталог спортивной продукции по категориям (питание, одежда, инвентарь), чтобы пользователь мог быстро найти нужный товар.

#### 1.3.3. Корзина

Любой выбранный товар добавляется в корзину одним нажатием. В разделе «Корзина» пользователь видит список позиций, может очистить корзину, а также быстро оплатить.

#### 1.3.4. История покупок

В профиле каждого пользователя сохраняется история заказов: дата, время и сумма на которую была сделана покупка.

Минимальные системные требования для работы SportShopBot:

- установлен мессенджер Telegram (смартфон, планшет или ПК);
- наличие стабильного интернет-соединения.

С этими элементами бот сможет обеспечить пользователю интуитивно понятный, быстрый и безопасный процесс покупки спортивных товаров прямо в привычном чате.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.49.001 ПЗ | Лист |
|      |      |          |         |      |                       | 13   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

## 2 Выбор инструментов разработки

Для написания Телеграмм-бота я использовал язык программирования C# и базу данных PostgreSQL.

### 2.1 C#

C# – это современный язык программирования, разработанный компанией Microsoft в рамках экосистемы .NET. Он сочетает в себе удобство объектно-ориентированного подхода и высокую производительность, что делает его востребованным во многих сферах разработки.

Основные преимущества C#:

- объектно-ориентированность – поддержка ключевых принципов ООП, таких как инкапсуляция, наследование и полиморфизм, что помогает строить структурированные и легко расширяемые приложения;

- безопасность типов – строгая типизация позволяет выявлять ошибки на этапе компиляции, снижая вероятность некорректных операций с данными;

- автоматическое управление памятью – встроенный механизм сборки мусора помогает избегать утечек памяти и упрощает работу с динамическими объектами;

- асинхронное программирование – поддержка ``async/await`` даёт возможность работать с длительными операциями, такими как обработка сетевых запросов, без блокировки потоков;

- гибкость и широкие возможности – язык применяется для создания веб-приложений (ASP.NET), настольных программ (Windows Forms, WPF), мобильных решений (Xamarin) и игровых проектов (Unity).

Благодаря своей простоте, мощным инструментам и развитой экосистеме, C# остаётся одним из самых популярных языков программирования и используется для разработки разнообразного программного обеспечения.

## 2.2 PostgreSQL

PostgreSQL – это мощная объектно–реляционная система управления базами данных (СУБД), которая предоставляет широкий набор инструментов для хранения, обработки и анализа данных. Эта СУБД работает по принципу клиент–серверной архитектуры, где приложение клиента соединяется с сервером PostgreSQL, чтобы выполнять операции с данными. Сам сервер состоит из множества процессов, включая главный процесс, процессы управления соединениями и рабочие процессы, занимающиеся обработкой запросов.

Основные возможности PostgreSQL:

- язык запросов – PostgreSQL поддерживает стандартный SQL, что делает его универсальным инструментом для работы с базами данных. Кроме того, система предоставляет возможность использовать PL/SQL для написания процедур и функций, а также поддерживает такие языки, как Perl, Python, Java и C, что расширяет возможности работы с данными;

- хранение данных – PostgreSQL использует объектно–реляционную модель, при которой информация хранится в таблицах, содержащих строки, столбцы и индексы. Кроме традиционного хранения, СУБД поддерживает JSON–документы, позволяя работать со сложными и неструктурированными данными;

- транзакции – PostgreSQL позволяет выполнять несколько операций как единый процесс, что обеспечивает целостность данных. В случае ошибки одной из операций транзакция отменяется, предотвращая повреждение информации.

Благодаря своим возможностям PostgreSQL широко применяется в разработке корпоративных систем, веб–приложений и аналитических платформ, обеспечивая стабильную работу и высокую гибкость управления данными.

## 2.3 Rest

REST (Representational State Transfer) – это подход к созданию веб-сервисов, который основан на использовании стандартных HTTP-запросов для взаимодействия с данными. Он позволяет клиентам обращаться к ресурсам при помощи таких методов, как GET (получение данных), POST (создание нового элемента), PUT (обновление) и DELETE (удаление). REST обеспечивает простую и удобную передачу информации между клиентом и сервером, сохраняя понятную структуру и легкость интеграции в различные приложения. Благодаря этому REST широко применяется для разработки API, которые работают с веб-приложениями и мобильными сервисами.

|      |      |          |         |      |                              |      |
|------|------|----------|---------|------|------------------------------|------|
|      |      |          |         |      | <i>ПГУ1.090301.49.001 ПЗ</i> | Лист |
|      |      |          |         |      |                              | 16   |
| Изм. | Лист | № докум. | Подпись | Дата |                              |      |

### 3 Архитектура

Трёхуровневая архитектура – это способ организации программного обеспечения, при котором система разделяется на три отдельных слоя: клиент, сервер и база данных.

Клиентское приложение отвечает за взаимодействие с пользователем, предоставляя интерфейс для ввода и вывода информации.

Серверное приложение принимает запросы от клиента, обрабатывает их, выполняет логику программы и передаёт данные.

Сервер базы данных хранит всю необходимую информацию и предоставляет её серверу при необходимости.

Такой подход улучшает масштабируемость, безопасность и управляемость системы, поскольку каждый уровень выполняет свою специализированную задачу. Диаграмма архитектуры представлена на рисунке 3.1.

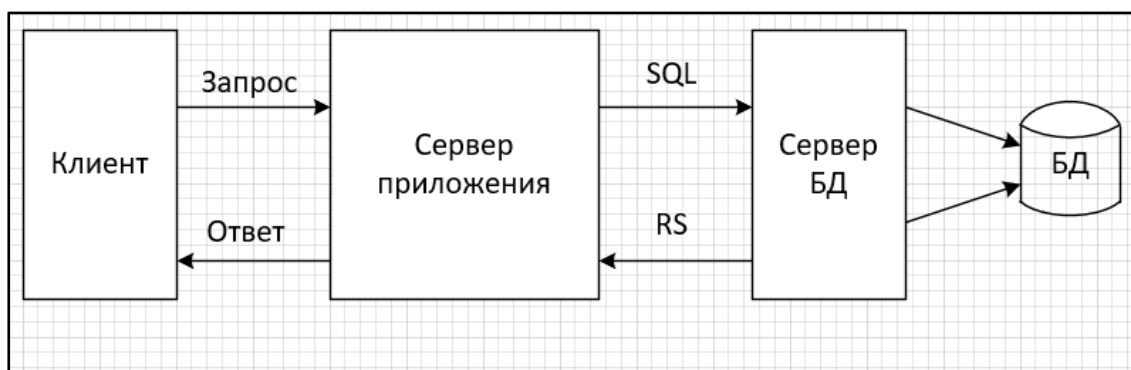


Рисунок 3.1 – Диаграмма архитектуры

Диаграмма прецедентов детализирует взаимодействие пользователя с Телеграмм-ботом, демонстрируя его функциональные возможности и удобство использования. Данная диаграмма представлена на рисунке 3.2.

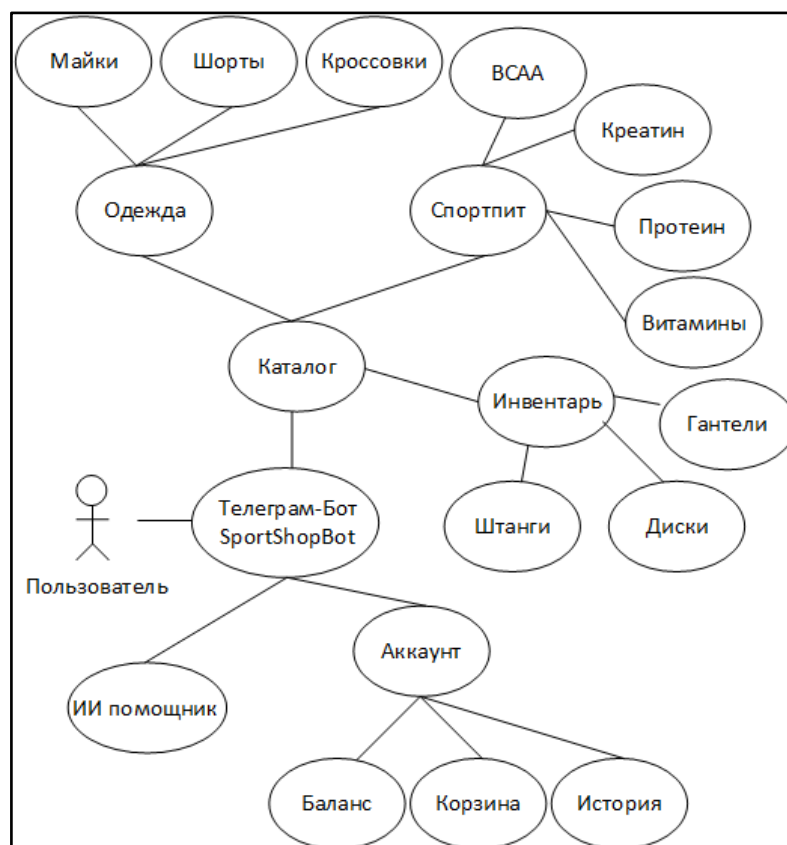


Рисунок 3.2 – Диаграмм прецедентов

Эта диаграмма детализирует взаимодействие пользователя с Телеграмм–ботом, демонстрируя его функциональные возможности и удобство использования.

В центре диаграммы находится сам бот, который является связующим звеном между пользователем и доступными товарами. Каталог товаров разделен на три главные категории:

- одежда – включает майки, шорты, кроссовки и другие элементы спортивного гардероба;
- спортивное питание – предлагает добавки, такие как ВСАА, креатин, протеин и витамины, необходимые для эффективных тренировок и восстановления;
- инвентарь – охватывает оборудование для занятий спортом, например гантели, штанги и диски.

Кроме того, пользователь может управлять своим аккаунтом, где представлены:

- баланс – отслеживание доступных средств для покупок;
- корзина – место, где хранятся выбранные товары перед их оформлением;
- история покупок – отображает список ранее приобретенных товаров.

Дополнительной функцией является ИИ–помощник, который может предоставить ответ на нужный вопрос пользователя.

Диаграмма последовательности визуализирует взаимодействие между пользователем, клиентом и сервером в процессе работы с ботом для покупок. Она организована в виде последовательности шагов, где каждый элемент системы выполняет свою функцию, обеспечивая плавный процесс взаимодействия. Данная диаграмма представлена на рисунке 3.2.

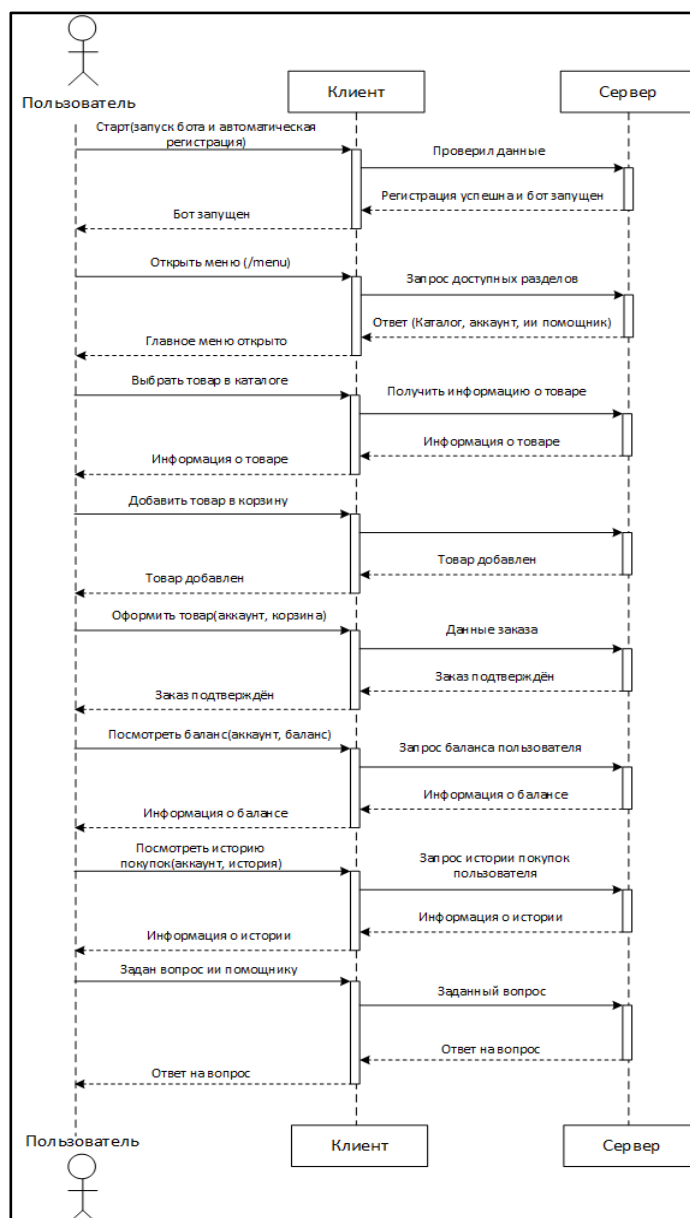


Рисунок 3.3 – Диаграмма последовательности

Основные участники диаграммы:

- пользователь – инициатор взаимодействия, использует бота для регистрации, выбора товаров, проверки баланса и истории покупок;
- клиент (бот) – принимает запросы от пользователя, направляет их на сервер, обрабатывает ответы и предоставляет данные в удобном виде;
- сервер – основная часть системы, где хранятся и обрабатываются данные о пользователях, товарах, балансе и покупках.

|      |      |          |         |      |
|------|------|----------|---------|------|
|      |      |          |         |      |
| Изм. | Лист | № докум. | Подпись | Дата |

Основные процессы расписаны ниже.

#### Регистрация:

- пользователь запускает бота и отправляет запрос на регистрацию;
- клиент передаёт запрос серверу;
- сервер проверяет данные, создаёт аккаунт, отправляет подтверждение обратно клиенту;
- бот уведомляет пользователя об успешной регистрации, предоставляя доступ к меню.

#### Выбор товаров:

- пользователь выбирает товары, отправляя запрос боту;
- клиент передаёт запрос серверу для получения актуальной информации о товарах;
- сервер отправляет данные о наличии, ценах и характеристиках;
- бот отображает информацию пользователю.

#### Оформление заказа:

- пользователь подтверждает заказ;
- клиент передаёт данные серверу;
- сервер фиксирует заказ, списывает средства с баланса и отправляет подтверждение;
- бот уведомляет пользователя об успешном оформлении.

#### Просмотр баланса и истории покупок:

- пользователь запрашивает данные;
- бот отправляет запрос серверу;
- сервер возвращает текущий баланс и список предыдущих покупок;
- бот отображает информацию пользователю.

#### Обратная связь с помощником:

- пользователь задаёт вопрос;

- бот направляет запрос серверу;
- сервер возвращает ответ;
- бот передаёт ответ пользователю.

Эта диаграмма помогает понять, как устроен процесс взаимодействия между пользователем и системой. Она логично структурирована, содержит последовательные шаги и обеспечивает наглядность всех ключевых этапов.

### 3.1 Структура БД

Диаграмма представляет структуру базы данных, где есть несколько сущностей (таблиц) и связи между ними.

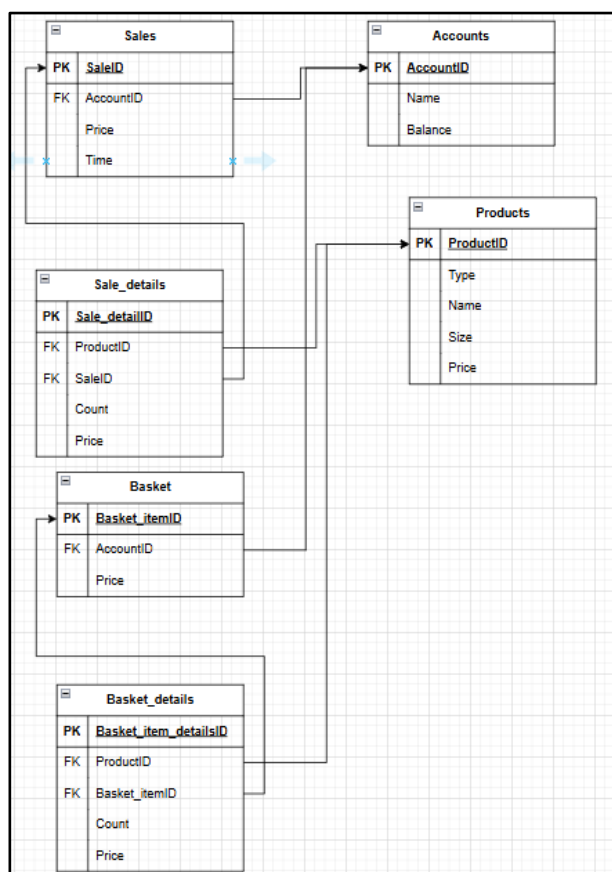


Рисунок 3.4 – Диаграмма БД

Описание таблиц и их взаимосвязей представлены ниже.

Sales (Продажи) – таблица содержащая информацию о продажах:

- SaleID – первичный ключ (PK), уникальный идентификатор продажи;
- AccountID – внешний ключ (FK), связывающий продажу с аккаунтом покупателя;

- Price – общая стоимость продажи;
- Time – дата и время совершения продажи.

Accounts (Аккаунты) – содержит данные о пользователях системы:

- AccountID – первичный ключ (PK), идентификатор аккаунта;
- Name – имя владельца аккаунта;
- Balance – баланс средств на аккаунте.

Products (Товары) – хранит данные о продуктах:

- ProductID – первичный ключ (PK), уникальный идентификатор товара;
- Type – тип товара;
- Name – название товара;
- Size – размер товара;
- Price – цена товара.

Sale\_details (Детали продаж) – связывает продажу с конкретными товарами:

- Sale\_detailID – первичный ключ (PK), идентификатор строки с деталями;
- ProductID – внешний ключ (FK), идентификатор товара;
- SaleID – внешний ключ (FK), идентификатор продажи;
- Count – количество товара в продаже;
- Price – стоимость за товарную позицию.

Basket (Корзина) – таблица временного хранения товаров, выбранных пользователем перед покупкой:

- Basket\_itemID – первичный ключ (PK), идентификатор корзины;
- AccountID – внешний ключ (FK), связывающий корзину с пользователем;

- Price – общая стоимость товаров в корзине.

Basket\_details (Детали корзины) – связь между товарами и корзиной.

- Basket\_item\_detailID – первичный ключ (PK), уникальный идентификатор позиции в корзине;
- ProductID – внешний ключ (FK), идентификатор товара;
- Basket\_itemID – внешний ключ (FK), связывающий товар с конкретной корзиной;
- Count – количество данного товара в корзине;
- Price – стоимость за товарную позицию.

Логика работы базы данных:

- пользователь регистрируется → создается запись в таблице Accounts;
- пользователь добавляет товары в корзину → создаются записи в Basket и Basket\_details;
- пользователь оформляет заказ → создается запись в Sales. Товары из корзины перемещаются в Sale\_details.

## 4 Тестирование

Для начала нужно нажать кнопку СТАРТ или написать /start. Тем самым вы запустите бота и автоматически регистрируетесь. Демонстрация запуска представлена на рисунках 4.1 и 4.2.



Рисунок 4.1 – Выполнение команды старт

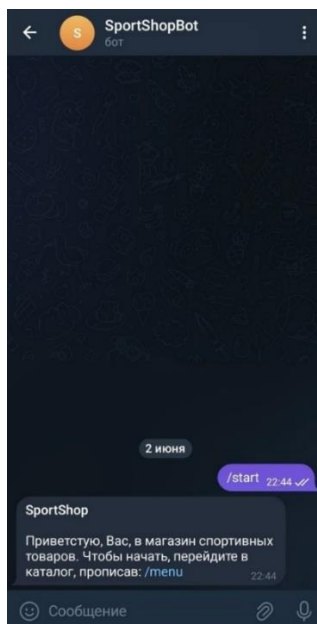


Рисунок 4.2 – Результат выполнения команды старт

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.49.001 ПЗ | Лист |
|      |      |          |         |      |                       | 25   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

Далее для открытия основного меню вы должны написать /menu. Выполненная команда представлена на рисунке 4.3.

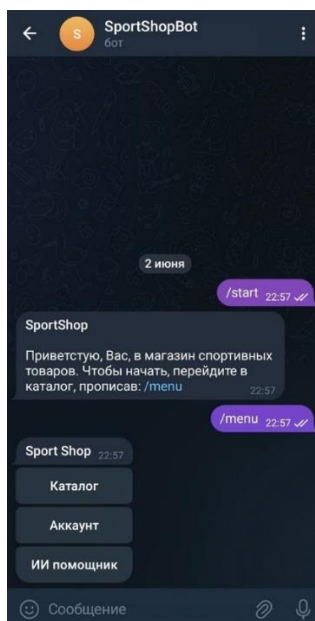


Рисунок 4.3 – Основное меню

Перед нами появляется основное меню магазина, в котором мы можем перейти в каталог, аккаунт и ИИ помощник. Нажимаем на каталог, и мы видим категории товар которые есть в магазине. Каталог представлен на рисунке 4.4. Так же у нас есть кнопка назад, которая нас возвращает в основное меню.

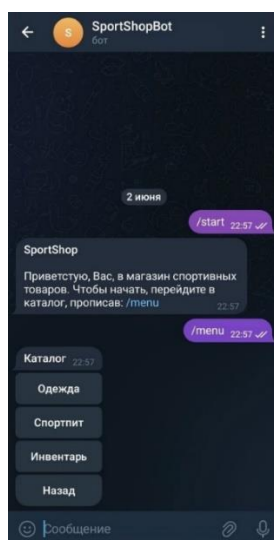


Рисунок 4.4 – Каталог

Выбираем нам нужную категорию и нажимаем на неё. Данное действие продемонстрировано на рисунке 4.5.

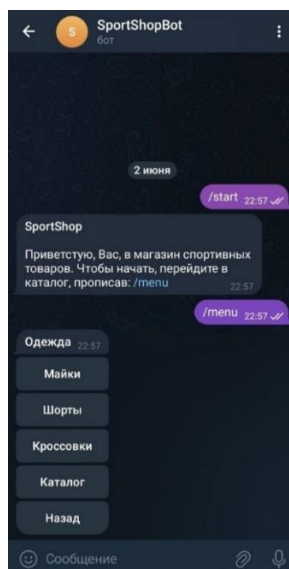


Рисунок 4.5 – Категория товаров

Здесь мы можем выбрать нужный товар в данной категории. Также у нас есть кнопка каталог, которая нас возвращает в меню каталога и кнопка назад, которая возвращает нас в основное меню магазина. Выбранная категория товара представлена на рисунке 4.6.

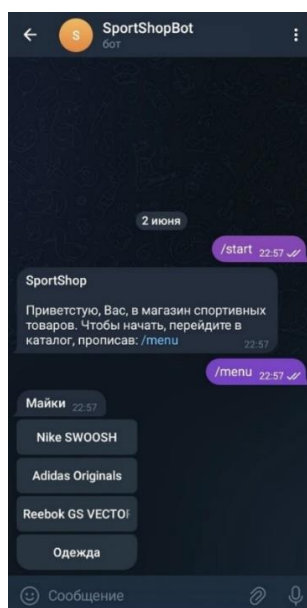


Рисунок 4.6 – Выбранная категория

После выбора нужного товара, перед нами появляется список с фирмами, а также кнопка для возврата для выбора товара. Выбираем нужную фирму. Данное действие представлено на рисунке 4.7.

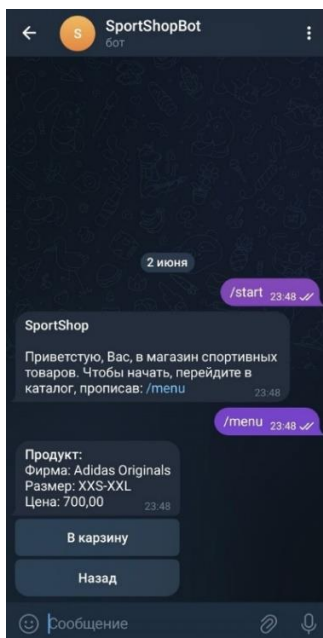


Рисунок 4.7 – Описание выбранного продукта

Появляется окно, в котором мы видим название фирмы, размер и цену. Также есть кнопка для добавления товара в корзину и кнопка назад для перехода в основное меню.

Возвращаемся в основное меню и переходим в аккаунт. Здесь мы видим меню аккаунта: баланс, корзина, история, которые представлены на рисунке 4.8.

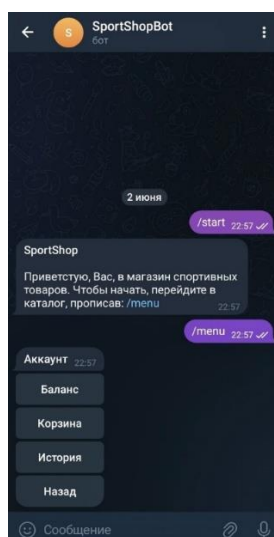


Рисунок 4.8 – Меню аккаунта

Заходим в раздел баланс, представленный на рисунке 4.9.

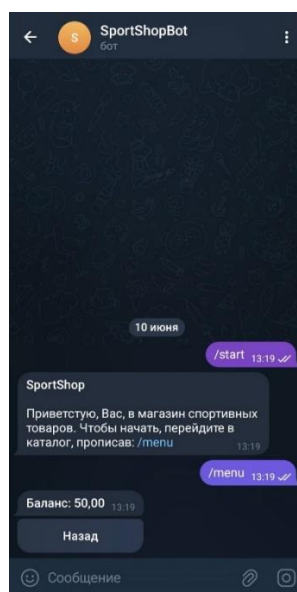


Рисунок 4.9 – Баланс

Возвращаемся и заходим в корзину, представленную на рисунке 4.10. Там мы видим все добавленные товары и информацию о них. Также мы можем оплатить и удалить корзину и вернуться обратно в меню.

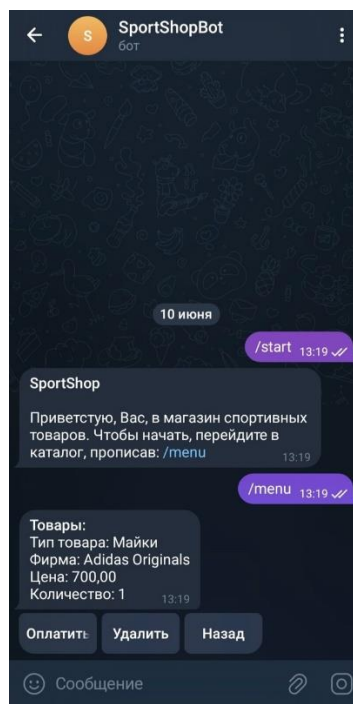


Рисунок 4.9 – Корзина

Теперь заходим в историю, представленную на рисунке 4.11. В которой видим сумму, дату и время сделки.

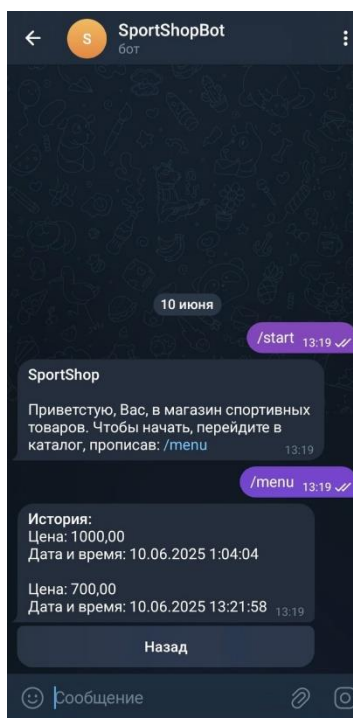


Рисунок 4.11 – История сделок

Возвращаемся в основное меню и переходим в ИИ помощник. В нём мы можем задать нужный нам вопрос, прописав /ai и написав вопрос. Работа помощника продемонстрирована на рисунке 4.12 и 4.13.

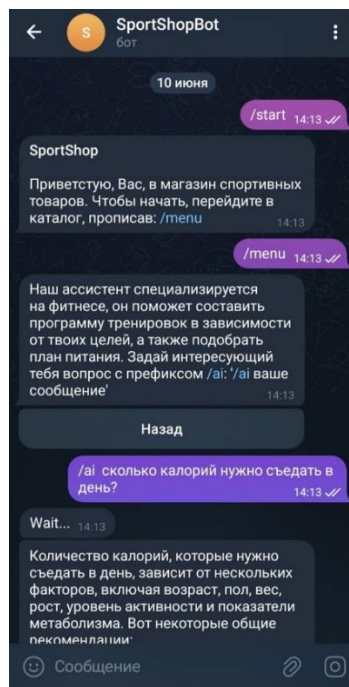


Рисунок 4.12 – Меню ии-помощника

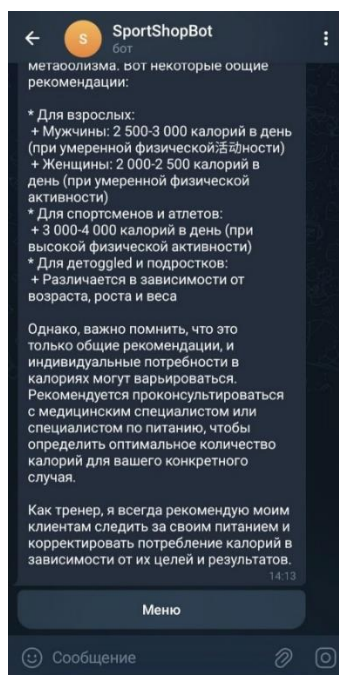


Рисунок 4.13 – Ответ ИИ-помощника

Получаем ответ и нажимаем кнопку меню для открытия основного меню, которое изображено на рисунке 4.14.

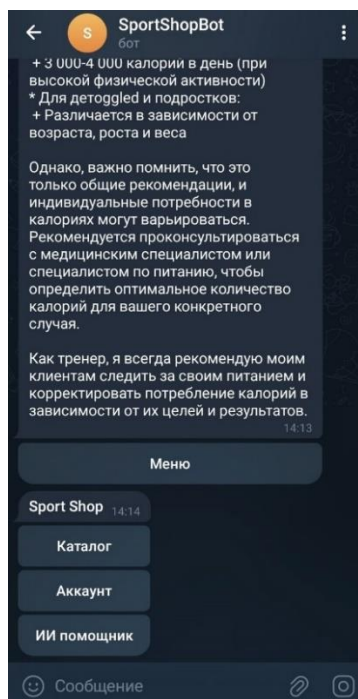


Рисунок 4.14 – Открытие основного меню

## 5 Безопасность жизнедеятельности

Работа за компьютером требует внимания к вопросам безопасности и здоровья, особенно для разработчиков и других специалистов, которые проводят за экраном много времени. Несоблюдение простых правил может привести к проблемам со зрением, нарушению осанки, а также повышенному риску развития карпального туннельного синдрома и других заболеваний. В этом разделе будут рассмотрены ключевые аспекты безопасной работы за компьютером и полезные рекомендации для разработчиков.

### 5.1 Требование к помещению и рабочему месту.

Требования к помещению для работы с компьютером:

- ориентация и площадь: окна на север или северо–восток, не менее 4,5 м<sup>2</sup> на одно рабочее место с ЖК–монитором;
- освещение: комбинированное – естественное и равномерное искусственное. Основной свет должен падать сбоку (слева), рядом с окном – жалюзи или плотные шторы для защиты от бликов;
- вентиляция и чистота воздуха: приточно–вытяжная система, скорость движения воздуха  $\leq 0,1$  м/с. Концентрация формальдегида и других ЛОС не выше ПДК по гигиеническим нормам;
- климат–контроль: температура 20–24 °С (оптимум 22 °С), относительная влажность 40–60 %;
- шумовой уровень: не выше 50 дБ, чтобы не отвлекаться и снижать стресс;
- электробезопасность: обязательное защитное заземление, качественные кабели и розетки – чтобы минимизировать электромагнитные излучения и ударный ток.

Требования к рабочему месту:

- стол: высота 70–75 см, глубина и ширина – не менее 120×60 см, чтобы разместить монитор, клавиатуру и документы;
- кресло: с регулируемой высотой, глубиной сиденья и поясничной опорой; стопы – полностью на полу или на подставке;
- монитор: экран на расстоянии 50–70 см от глаз, верхняя кромка чуть ниже уровня глаз (5–10° вниз); антибликовое покрытие или фильтр синего света;
- расположение периферии: клавиатура и мышь на одном уровне с локтями, запястья – прямо, без изгибов;
- освещение рабочего стола: настольная лампа с рассеянным светом для бумаги и заметок;
- перерывы и упражнения: каждые 45–60 мин – 5–минутная разминка (потягивания, вращение плечами, гимнастика для глаз).

Соблюдение этих простых, но важных требований поможет сохранить зрение, осанку и общее самочувствие, а ещё–повысить продуктивность и настроение в течение рабочего дня.

## 5.2 Требование к освещению.

Освещение на рабочем месте разработчика должно быть продуманным и многослойным: одним лишь потолочным светильником тут не обойтись. Во–первых, комбинируйте естественный и искусственный свет – дневные лучи наполняют пространство мягким, равномерным свечением, а лампы восполняют недостачу в пасмурную погоду или вечером. При этом главное – избежать резких контрастов: если монитор светится «ярче, чем осветитель», глаза быстро устанут.

Ключевые параметры освещённости:

- общее фоновой свет – примерно 300–400 лк на уровне пола (или 500–750 лк на рабочей поверхности);
- локальное освещение для бумаг и заметок – до 500–750 лк, организованное настольной лампой с рассеянным светом.

Чтобы не было бликов и отражений на экране, следите за расположением источников света:

- потолочные светильники с матовым рассеивателем обеспечивают мягкий «заполняющий» свет без резких теней;
- настольная лампа ставится сбоку (лучше – с левой), а не прямо за монитором и не прямо перед ним;
- окна прикрывайте жалюзи или рулонными шторами – так вы сохраните естественный свет, но защититесь от прямых солнечных лучей.

Важные нюансы:

- цветовая температура ламп около 4000–4500 К – это золотая середина: свет не слишком «холодный», но и не вызывает сонливости, как тёплые лампы;
- индекс цветопередачи (CRI) не ниже 80 – чтобы цвета на бумаге и экране выглядели естественно;
- антибликовые экраны и матовые покрытия предметов на столе минимизируют излишние отблески;
- по возможности используйте лампы с регулировкой яркости и без мерцания – это снижает нагрузку на сетчатку и нервную систему.

Правильно спроектированное освещение не только спасёт глаза от перенапряжения, но и повысит настроение и продуктивность, сделав работу за компьютером комфортнее и безопаснее.

### 5.3 Требования к режиму труда и отдыха.

Требования к режиму труда и отдыха при работе за компьютером.

Основной рабочий цикл:

- интервалы работы – 50–60 минут, затем 5–10–минутный перерыв;
- после каждых 3–4 циклов планируйте более длинный отдых – 15–20 минут.

Глазная разгрузка:

- правило 20–20–20: каждые 20 минут смотреть на отдалённый предмет в течение 20 секунд на расстоянии не менее 6 метров;
- в короткие паузы переходите к окну или выходите на балкон – смена фокуса и свежий воздух помогают глазам.

Двигательная активность:

- вставайте каждые 50–60 минут хотя бы на 1–2 минуты: пройдитесь по офису, потянитесь, сделайте пару приседаний;
- для спины и шеи полезна лёгкая разминка: круговые движения плечами, наклоны головы вперёд–назад и в стороны.

Обеденный перерыв:

- минимум 30–45 минут вдали от экрана – поешьте, пройдитесь, поговорите с коллегами;
- если есть возможность, выйдите на короткую прогулку: она «перезагрузит» мозг и улучшит кровообращение.

Водный баланс и питание:

- держите под рукой бутылку воды – пейте по глотку каждые 20–30 минут;
- лёгкие перекусы (орехи, фрукты, йогурты) не дают голоду отвлекать и поддерживают энергию.

#### Завершение рабочего дня:

- за 30–60 минут до конца смены выключите уведомления, запишите список дел на завтра и сделайте пару растяжек;
- если вы постоянно «зависаете» в задачах, заведите приложение–таймер (Pomodoro, Swimlanes и т.п.) – оно напомнит вовремя отдохнуть.

#### Общие принципы:

- слушайте свой организм: при сильной усталости делайте вынужденный отдых, даже если график плотный;
- меняйте тип активности: после «тонкой» умственной работы переключайтесь на «черновые» задачи – и наоборот;
- старайтесь ложиться спать и вставать в одно и то же время: регулярность сна напрямую влияет на продуктивность днём.

Соблюдение этих простых правил поможет сохранить здоровье глаз и спины, снизить нервное напряжение и работать в комфортном, безопасном режиме.

## Заключение

В ходе выполнения выпускной квалификационной работы был реализован бот для системы Telegram, с использованием языка C#, как основного языка программирования. Также были выполнены следующие задачи:

- полный анализ темы и аналогов;
- выбор инструментов разработки;
- анализ архитектуры;
- непосредственная разработка бота;
- тестирование бота.

Реализованный бот успешно решает задачу упрощения и автоматизации процесса покупки фитнес товаров. В результате проделанной работы пользователь получает интуитивный интерфейс: легко находит нужную категорию товаров и добавляет позицию в корзину, не выходя из знакомого мессенджера.

В перспективе функционал бота можно будет усовершенствовать за счёт внедрения новых возможностей и уникальных решений.

### Список использованных источников

1. Моргунов Е.П. PostgreSQL. Основы языка SQL: учеб. Пособие / Е.П. Моргунов; под ред. Е.В. Рогова, П.В. Лузанова. – СПб.: БХВ–Петербург, 2018. – 336 с.: ил.
2. core.telegram.org: [Электронный ресурс]. URL: <https://core.telegram.org/#tdlib-build-your-own-telegram> (дата обращения: 10.05.2025)
3. ru.m.wikipedia.org: [Электронный ресурс]. URL: [https://ru.m.wikipedia.org/wiki/Data\\_Access\\_Object](https://ru.m.wikipedia.org/wiki/Data_Access_Object) (дата обращения: 12.05.2025)
4. Ржеуцкая С.Ю. Базы данных. Язык SQL: учеб. пособие / С.Ю. Ржеуцкая. – Вологда: ВоГТУ, 2010. – 17–22 с.
5. metanit.com: [Электронный ресурс]. URL: <https://metanit.com/sharp/tutorial/> (дата обращения: 13.05.2025)
6. metanit.com: [Электронный ресурс]. URL: <https://metanit.com/sql/postgresql/> (дата обращения: 13.05.2025)
7. www.consultant.ru: [Электронный ресурс]. URL: [https://www.consultant.ru/document/cons\\_doc\\_LAW\\_401279/e58bf5df0f196f985fceed4354f3e084cf40eddb/](https://www.consultant.ru/document/cons_doc_LAW_401279/e58bf5df0f196f985fceed4354f3e084cf40eddb/) (дата обращения: 23.05.2025)
8. oxrana-bez.ru: [Электронный ресурс]. URL: <https://oxrana-bez.ru/stati/tehnika-bezopasnosti-pri-rabote-s-personalnym-kompyuterom/> (дата обращения: 23.05.2025)

## Приложение А

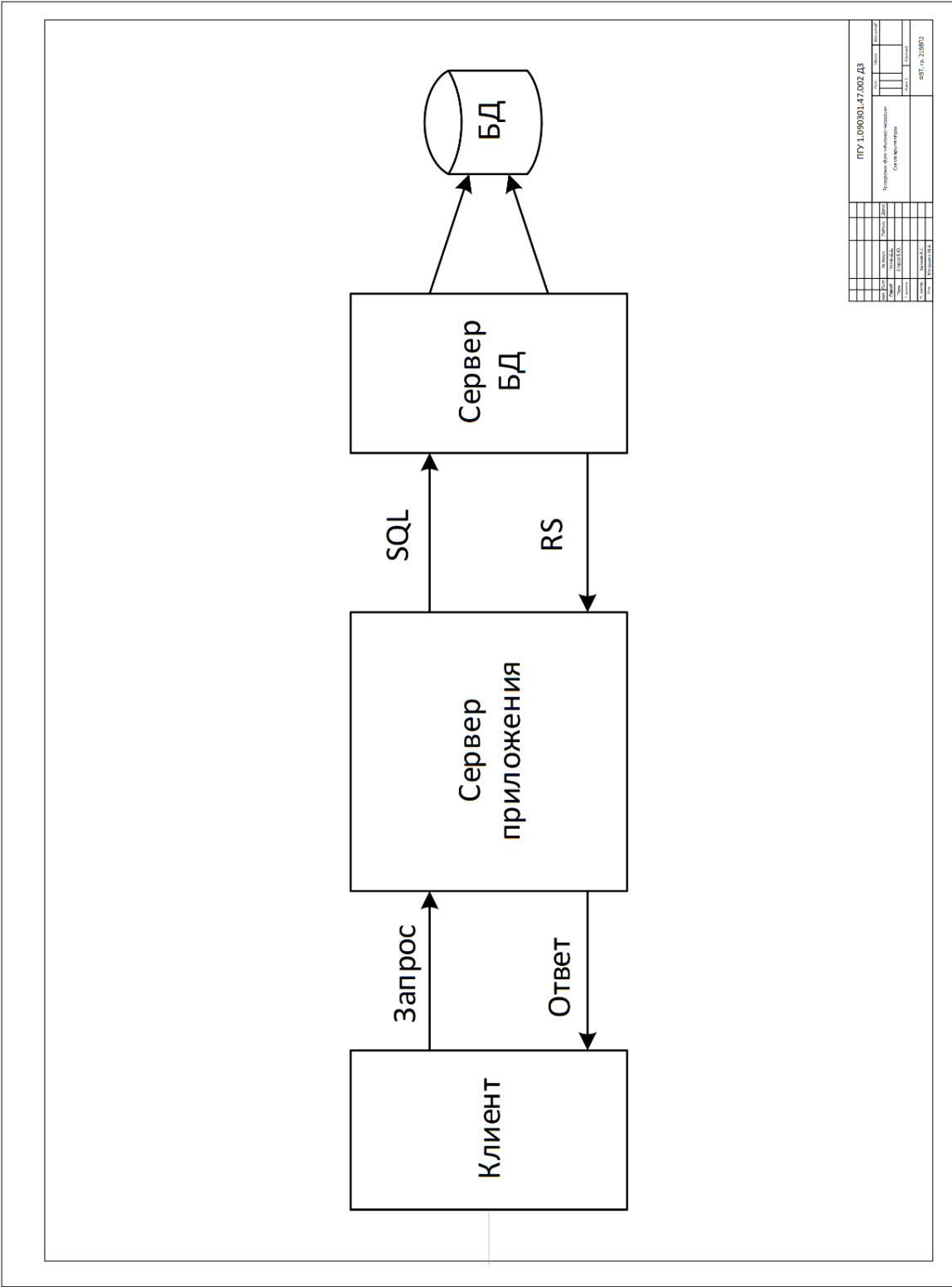
(Обязательное)

Телеграмм–бот «Фитнес–магазин»

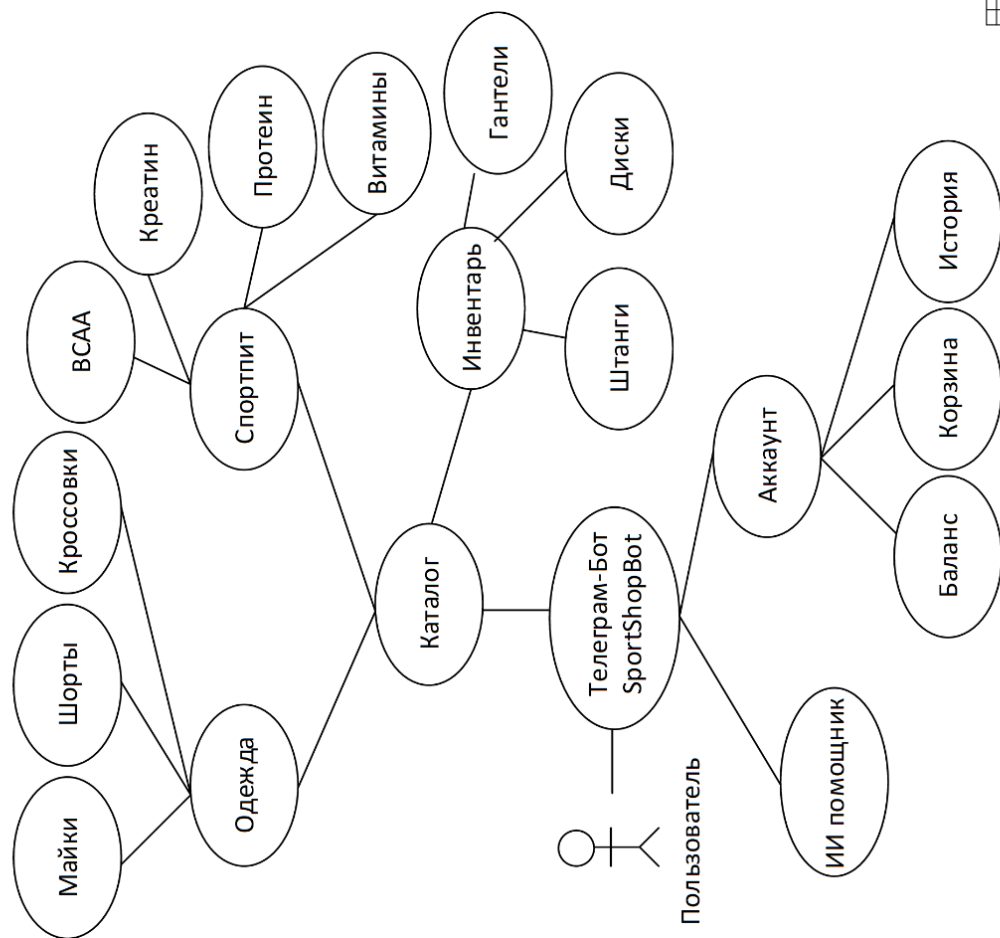
UML–диаграммы

|      |      |          |         |      |                              |      |
|------|------|----------|---------|------|------------------------------|------|
|      |      |          |         |      | <i>ПГУ1.090301.49.001 ПЗ</i> | Лист |
|      |      |          |         |      |                              | 40   |
| Изм. | Лист | № докум. | Подпись | Дата |                              |      |

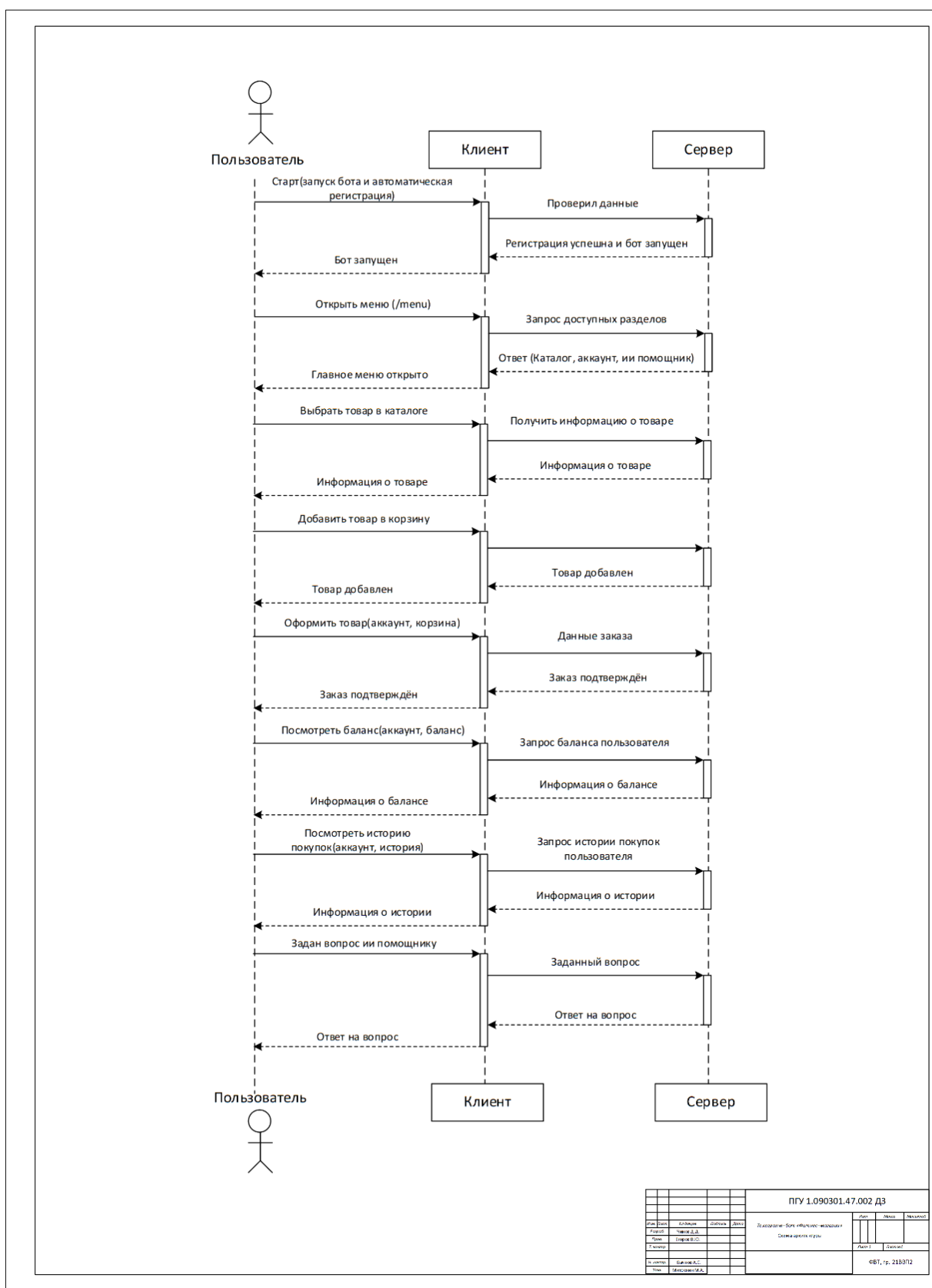
ПриложениеА.1 – Схема работы системы



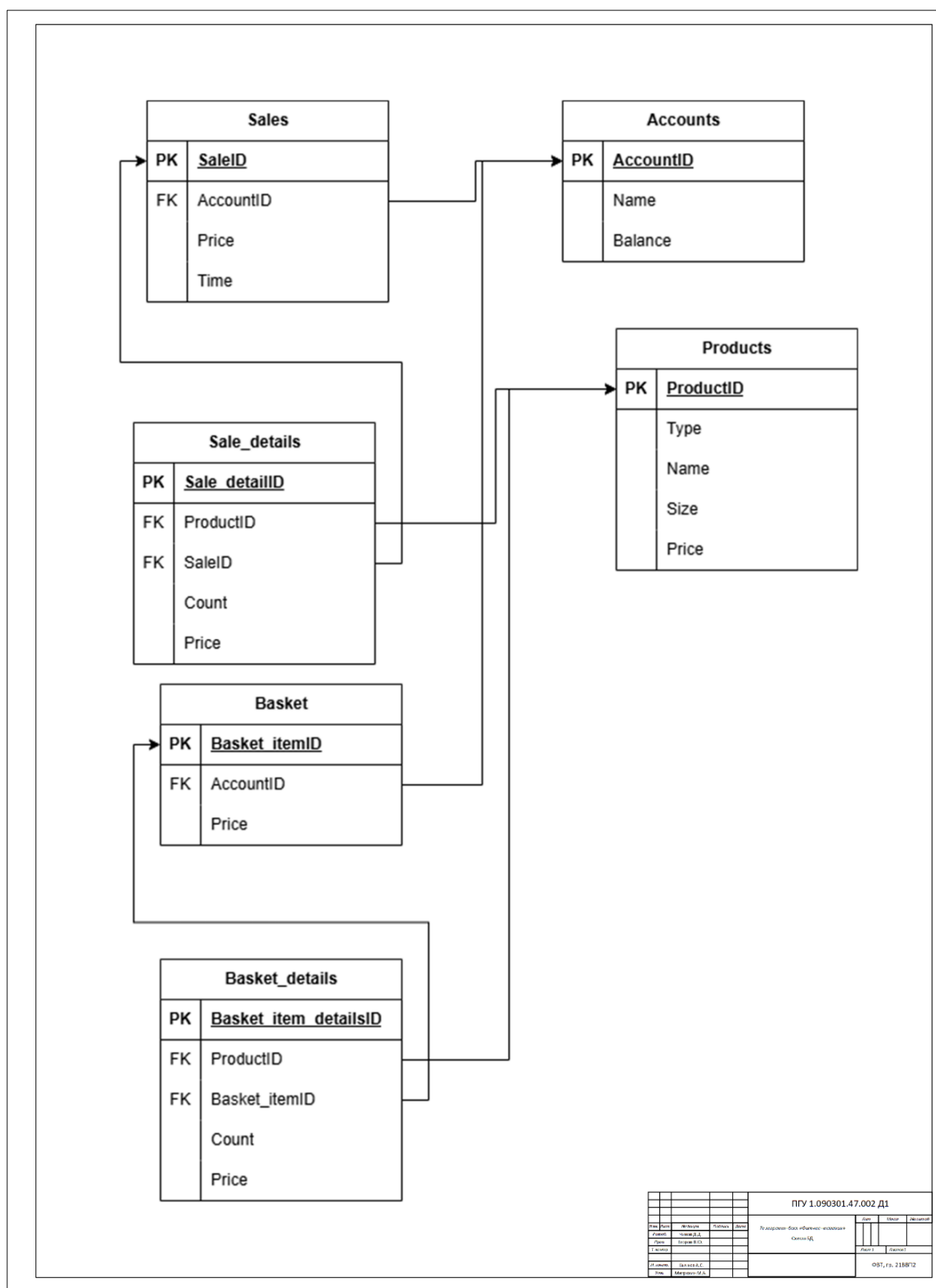
## ПриложениеА.2 – Диаграмма прецедентов

[illegible]

## Приложение А.3 – Диаграмма последовательности



## Приложение А.4 – Схема БД



|                        |         |          |     |      |                                     |  |  |  |  |
|------------------------|---------|----------|-----|------|-------------------------------------|--|--|--|--|
| ПГУ 1.090301.47.002 Д1 |         |          |     |      |                                     |  |  |  |  |
| Имя                    | Фамилия | Отчество | Пол | Дата | Информация для бухгалтерского учета |  |  |  |  |
| Имя                    | Фамилия | Отчество | Пол | Дата | Счетчик                             |  |  |  |  |
| Имя                    | Фамилия | Отчество | Пол | Дата | Лист 1                              |  |  |  |  |
| Имя                    | Фамилия | Отчество | Пол | Дата | 057, р. 215872                      |  |  |  |  |

|      |      |          |         |      |
|------|------|----------|---------|------|
| Изм. | Лист | № докум. | Подпись | Дата |
|      |      |          |         |      |

ПГУ1.090301.49.001 ПЗ

Лист

44

## Приложение Б

(Обязательное)

Телеграмм–бот «Фитнес–магазин»

Листинг программы

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.49.001 ПЗ | Лист |
|      |      |          |         |      |                       | 45   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

### Файл «Program.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Threading;
using Telegram.Bot;
using Telegram.Bot.Types;
using Telegram.Bot.Types.Enums;
using Telegram.Bot.Types.ReplyMarkups;
using SportShopBot.Entities;
using System.Net.Http;

namespace SportShopBot
{
    internal class Program
    {
        static void Main(string[] args)
        {
            const int MAX_LENGTH_MESSAGE = 4096;

            // Дескриптор для взаимодействия с базой данных
            IDatabaseHandler dbHandler = new
            DatabaseHandler("Host=localhost;Port=5432;Username=postgres;Passw
            ord=123;Database=sportshop");

            // Дескриптор для общения с ботом
            var bot = new
            TelegramBotClient("7977150259:AAFiLOXV3GtLNUXqGhjlmOmMljKfi6j6zSY
            ");

            var cts = new CancellationTokenSource();

            // StartReceiving не блокирует вызывающий поток.
            Прием выполняется в пуле потоков, поэтому мы используем токен отмены
            bot.StartReceiving(
                updateHandler: HandleUpdate,
                errorHandler: HandleError,
                cancellationToken: cts.Token
            );

            // Сообщите пользователю, что бот подключен к сети
            Console.WriteLine("Start listening for updates.
            Press enter to stop");
            Console.ReadLine();

            // Отправьте запрос на отмену, чтобы остановить
            бота
            cts.Cancel();
        }
    }
}
```

```

        // Каждый раз, когда пользователь взаимодействует
        с ботом, вызывается этот метод
        async Task HandleUpdate(ITelegramBotClient _,
Update update, CancellationToken cancellationToken)
        {
            switch (update.Type)
            {
                // Было получено сообщение
                case UpdateType.Message:
                    await HandleMessage(update.Message);
                    break;

                // Была нажата кнопка
                case UpdateType.CallbackQuery:
                    await
HandleButton(update.CallbackQuery);
                    break;
            }
        }

        async Task HandleError(ITelegramBotClient _,
Exception exception, CancellationToken cancellationToken)
        {
            await
Console.Error.WriteLineAsync(exception.Message);
        }

        async Task HandleMessage(Message msg)
        {
            var user = msg.From;
            var text = msg.Text ?? string.Empty;

            if (user is null)
                return;

            if (text == "/start")
            {
                Var                userData                =
dbHandler.SelectWithConditionRow("accounts",                "*",
$"id={user.Id}");

                if (userData.Count == 0)
                {
                    dbHandler.InsertData("accounts", new
Dictionary<string, string>
                    {
                        { "id", user.Id.ToString() },
                        {
                            "name",
user.FirstName.ToString() },
                        { "balance", "0" }
                    }
                }
            }
        }
    }
}

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.49.001 ПЗ | Лист |
|      |      |          |         |      |                       | 47   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

        });
    }
    else if (userData["name"].ToString() !=
user.FirstName)
    {
        dbHandler.UpdateData("accounts",
$"name='{user.FirstName}'", $"id='{user.Id}'");
    }

    await bot.SendMessage(msg.Chat.Id,
Keyboard.startMenu, ParseMode.Html);
}

Console.WriteLine($"{user.FirstName} wrote
{text}");

// Когда мы получаем команду, мы реагируем
соответствующим образом
if (text.StartsWith("/"))
{
    await HandleCommand(msg, text);
}
else if (text.Length > 0)
{
    // Чтобы сохранить разметку, мы выделяем
объекты (жирным шрифтом, курсивом..)
    await bot.SendMessage(user.Id,
text.ToUpper(), entities: msg.Entities);
}
else
{
    // Это эквивалентно пересылке без указания
имени отправителя
    await bot.CopyMessage(user.Id, user.Id,
msg.MessageId);
}
}

async Task HandleCommand(Message msg, string
command)
{
    switch (command.Split(' ')[0])
    {
        case "/menu":
            await SendMenu(msg.From.Id);
            break;

        case "/ai":
            Task.Run(() => AIRequest(msg.Chat.Id,
String.Join(" ", command.Split(' ').Skip(1))));

```

```

        await bot.SendMessage(msg.From.Id,
"Wait...");
        break;
    }

    await Task.CompletedTask;
}

async Task SendMenu(long userId)
{
    await bot.SendMessage(
        userId,
        Keyboard.mainMenu,
        ParseMode.Html,
        replyMarkup: Keyboard.mainMenuMarkup
    );
}

// Получение списка продуктов определенного типа
void GetProducts(string type, string backBtn,
string productsMenu, ref string text, ref InlineKeyboardMarkup
markup)
{
    var products = dbHandler.GetProducts(type,
backBtn);

    if (products is null)
    {
        text = Keyboard.catalogMenu;
        markup = Keyboard.catalogMenuMarkup;
    }
    else
    {
        text = productsMenu;
        markup = products;
    }
}

// Обращение к модели искусственного интеллекта
async void AIRequest(long userId, string question)
{
    var client = new AIHelper(new HttpClient(),
"sk-or-v1-
8c8024adf62f8ab628524fb7e6c2526c92629d70250dbe5f02fe2751c3abdacd"
);

    var text = await client.GetAnswer(question);

    if (text.Count() > MAX_LENGTH_MESSAGE)
    {

```

```

MAX_LENHT_MESSAGE)
    for (int i = 0; i < text.Length; i +=
MAX_LENHT_MESSAGE)
    {
        string subtext;

        if (text.Length - i >=
MAX_LENHT_MESSAGE)
        {
            subtext = text.Substring(i,
MAX_LENHT_MESSAGE);

            await bot.SendMessage(
                userId,
                subtext,
                ParseMode.Html
            );
        }
        else
        {
            subtext = text.Substring(i,
text.Length - i);

            await bot.SendMessage(
                userId,
                subtext,
                ParseMode.Html,
                replyMarkup: new
InlineKeyboardMarkup(new[] {
                    new[]
InlineKeyboardButton.WithCallbackData(Keyboard.menuBtn) } })
            );
        }
    }
    else if (text.Count() <= MAX_LENHT_MESSAGE &&
text != string.Empty)
    {
        await bot.SendMessage(
            userId,
            text,
            ParseMode.Html,
            replyMarkup: new
InlineKeyboardMarkup(new[] {
                new[]
InlineKeyboardButton.WithCallbackData(Keyboard.menuBtn) } })
            );
    }
    else
    {
        await bot.SendMessage(
            userId,

```

```

        "If you didn't get the desired result,
try again.",

        ParseMode.Html,
        replyMarkup: new
InlineKeyboardMarkup(new[] {
    new[] {
        InlineKeyboardButton.WithCallbackData(Keyboard.menuBtn) } })
    });
}

async Task HandleButton(CallbackQuery query)
{
    var user = query.From;
    string text = string.Empty;
    var markup = new
InlineKeyboardMarkup(Array.Empty<InlineKeyboardButton>());

    if (query.Data == Keyboard.menuBtn)
    {
        text = Keyboard.mainMenu;
        markup = Keyboard.mainMenuMarkup;
    }
    // Основное меню
    else if (query.Data == Keyboard.backButton)
    {
        text = Keyboard.mainMenu;
        markup = Keyboard.mainMenuMarkup;
    }
    else if (query.Data == Keyboard.catalogBtn)
    {
        text = Keyboard.catalogMenu;
        markup = Keyboard.catalogMenuMarkup;
    }
    else if (query.Data == Keyboard.accountBtn)
    {
        text = Keyboard.accountMenu;
        markup = Keyboard.accountMenuMarkup;
    }
    // Каталог
    else if (query.Data == Keyboard.clothesBtn)
    {
        text = Keyboard.clothesMenu;
        markup = Keyboard.clothesMenuMarkup;
    }
    else if (query.Data == Keyboard.nutritionBtn)
    {
        text = Keyboard.nutritionMenu;
        markup = Keyboard.nutritionMenuMarkup;
    }
}

```

```

else if (query.Data == Keyboard.inventoryBtn)
{
    text = Keyboard.inventoryMenu;
    markup = Keyboard.inventoryMenuMarkup;
}
// Одежда
else if (query.Data == Keyboard.shirtBtn)
{
    GetProducts(Keyboard.shirtBtn,
Keyboard.clothesBtn, Keyboard.shirtMenu, ref text, ref markup);
}
else if (query.Data == Keyboard.sneakersBtn)
{
    GetProducts(Keyboard.sneakersBtn,
Keyboard.clothesBtn, Keyboard.sneakersMenu, ref text, ref markup);
}
else if (query.Data == Keyboard.shortBtn)
{
    GetProducts(Keyboard.shortBtn,
Keyboard.clothesBtn, Keyboard.shortsMenu, ref text, ref markup);
}
// Питание
else if (query.Data == Keyboard.bcaaBtn)
{
    GetProducts(Keyboard.bcaaBtn,
Keyboard.nutritionBtn, Keyboard.bcaaMenu, ref text, ref markup);
}
else if (query.Data == Keyboard.creatineBtn)
{
    GetProducts(Keyboard.creatineBtn,
Keyboard.nutritionBtn, Keyboard.creatineMenu, ref text, ref
markup);
}
else if (query.Data == Keyboard.proteinBtn)
{
    GetProducts(Keyboard.proteinBtn,
Keyboard.nutritionBtn, Keyboard.proteinMenu, ref text, ref markup);
}
else if (query.Data == Keyboard.vitaminsBtn)
{
    GetProducts(Keyboard.vitaminsBtn,
Keyboard.nutritionBtn, Keyboard.vitaminsMenu, ref text, ref
markup);
}
// Спортивный инвентарь
else if (query.Data == Keyboard.barbellsBtn)
{
    GetProducts(Keyboard.barbellsBtn,
Keyboard.inventoryBtn, Keyboard.barbellsMenu, ref text, ref
markup);
}

```

```

    }
    else if (query.Data == Keyboard.discsBtn)
    {
        GetProducts(Keyboard.discsBtn,
Keyboard.inventoryBtn, Keyboard.discMenu, ref text, ref markup);
    }
    else if (query.Data == Keyboard.dumbbellsBtn)
    {
        GetProducts(Keyboard.dumbbellsBtn,
Keyboard.inventoryBtn, Keyboard.dumbbellsMenu, ref text, ref
markup);
    }
    // Данные аккаунта
    else if (query.Data == Keyboard.balanceBtn)
    {
        text =
dbHandler.GetBalance(user.FirstName);
        markup = Keyboard.balanceMenuMarkup;
    }
    else if (query.Data == Keyboard.basketBtn)
    {
        text =
dbHandler.GetBasket(user.Id.ToString());
        markup = Keyboard.basketMenuMarkup;
    }
    else if (query.Data == Keyboard.historyBtn)
    {
        text =
dbHandler.GetDeals(user.Id.ToString());
        markup = new InlineKeyboardMarkup(new[] {
            new[] {
InlineKeyboardButton.WithCallbackData(Keyboard.backButton) } });
    }
    else if (query.Data == Keyboard.payBtn)
    {
        if (dbHandler.SetDeal(user.Id.ToString())
== 1)
        {
            text =
dbHandler.GetBasket(user.Id.ToString());
            markup = Keyboard.basketMenuMarkup;
        }
    }
    // Карточка товара
    else if (query.Data == Keyboard.buyBtn)
    {
        dbHandler.SetBasketProduct(user.Id.ToString(),
query.Message.Text);
    }
}

```

```

        else if (query.Data == Keyboard.delBtn)
        {
            if (dbHandler.DeleteData("basket",
$"account_id='{user.Id}'") > 0)
            {
                text =
dbHandler.GetBasket(user.Id.ToString());
                markup = Keyboard.basketMenuMarkup;
            }
        }
        else if (query.Data == Keyboard.helperButton)
        {
            text = Keyboard.helperMenu;
            markup = new InlineKeyboardMarkup(new[] {
                new[] {
InlineKeyboardButton.WithCallbackData(Keyboard.backButton) } });
        }
        else
        {
            var product =
dbHandler.SelectWithConditionRow("products", "*",
$"name='{query.Data}'");

            text = $"<b>Продукт: </b>\n" +
                $"Фирма: {product["name"]}\n" +
                $"Размер: {product["size"]}\n" +
                $"Цена: {product["price"]}";

            markup = Keyboard.productMenuMarkup;
        }

        // Закройте запрос, чтобы завершить анимацию
загрузки на стороне клиента
        await bot.AnswerCallbackQuery(query.Id);

        // Заменить текст меню и клавиатуру
        if (query.Data == Keyboard.menuBtn)
            await bot.SendMessage(
                query.Message.Chat.Id,
                text,
                ParseMode.Html,
                replyMarkup: markup
            );
        else if (text != string.Empty &&
markup.InlineKeyboard.Count() != 0)
            await bot.EditMessageText(
                query.Message.Chat.Id,
                query.Message.MessageId,
                text,
                ParseMode.Html,

```

```

        replyMarkup: markup
    );

    Task.WaitAll();
}
}
}
}
}

```

### Файл «Keyboard.cs»

```

using System;
using Telegram.Bot.Types.ReplyMarkups;

namespace SportShopBot.Entities
{
    internal class Keyboard
    {
        #region Pre-assign menu text

        public const string startMenu = "<b>SportShop</b>\n\n"
+
        "Приветствую, Вас, в магазин
спортивных товаров. " +
        "Чтобы начать, перейдите в
каталог, прописав: /menu";

        public const string helperMenu = "Наш ассистент
специализируется на фитнесе," +
        " он поможет составить программу
тренировок в зависимости от твоих целей," +
        " а также подобрать план питания." +
        " Задай интересующий тебя вопрос с
префиксом /ai: '/ai ваше сообщение'";

        public const string mainMenu = "<b>Sport Shop</b>";
        public const string catalogMenu = "<b>Каталог</b>";
        public const string accountMenu = "<b>Аккаунт</b>";
        public const string clothesMenu = "<b>Одежда</b>";
        public const string nutritionMenu = "<b>Спортивное
питание</b>";
        public const string inventoryMenu =
"<b>Инвентарь</b>";
        public const string shirtMenu = "<b>Майки</b>";
        public const string sneakersMenu = "<b>Кроссовки</b>";
        public const string shortsMenu = "<b>Шорты</b>";
        public const string bcaaMenu = "<b>BCAA</b>";
        public const string creatineMenu = "<b>Креатин</b>";
        public const string proteinMenu = "<b>Протеин</b>";
        public const string vitaminsMenu = "<b>Витамины</b>";
        public const string barbellsMenu = "<b>Штанги</b>";
    }
}

```

```

public const string dumbbellsMenu = "<b>Гантели</b>";
public const string discMenu = "<b>Диски</b>";
public const string noProducts = "<b>No products</b>";

#endregion

#region Pre-assign button text

public const string nextButton = "Next";
public const string mainButton = "Main";
public const string backButton = "Назад";
public const string menuBtn = "Меню";
public const string catalogBtn = "Каталог";
public const string commentsBtn = "Comments";
public const string accountBtn = "Аккаунт";
public const string balanceBtn = "Баланс";
public const string supportBtn = "Support";
public const string basketBtn = "Корзина";
public const string clothesBtn = "Одежда";
public const string nutritionBtn = "Спортпит";
public const string inventoryBtn = "Инвентарь";
public const string accessoriesBtn = "Accessories";
public const string tutorialButton = "Tutorial";
public const string helperButton = "ИИ помощник";

// Кнопки раздела "Clothes"
public const string shirtBtn = "Майки";
public const string shortBtn = "Шорты";
public const string sneakersBtn = "Кроссовки";

// Кнопки раздела "Nutritions"
public const string bcaaBtn = "BCAA";
public const string creatineBtn = "Креатин";
public const string proteinBtn = "Протеин";
public const string vitaminsBtn = "Витамины";

// Кнопки раздела "Inventory"
public const string barbellsBtn = "Штанги";
public const string discsBtn = "Диски";
public const string dumbbellsBtn = "Гантели";

// Кнопки раздела "Account"
public const string addBalanceBtn = "Append";
public const string historyBtn = "История";

// Кнопки раздела "Product card"
public const string buyBtn = "В корзину";

// Кнопки раздела "Basket"
public const string payBtn = "Оплатить";

```

```

        public const string delBtn = "Удалить";

        #endregion

        #region Main keyboards

        public static InlineKeyboardMarkup mainMenuMarkup =
new InlineKeyboardMarkup(
    new[] {
        new[] {
InlineKeyboardButton.WithCallbackData(catalogBtn) },
        new[] {
InlineKeyboardButton.WithCallbackData(accountBtn) },
        new[] {
InlineKeyboardButton.WithCallbackData(helperButton) }
    }
);

        public static InlineKeyboardMarkup catalogMenuMarkup
= new InlineKeyboardMarkup(
    new[] {
        new[] {
InlineKeyboardButton.WithCallbackData(clothesBtn) },
        new[] {
InlineKeyboardButton.WithCallbackData(nutritionBtn) },
        new[] {
InlineKeyboardButton.WithCallbackData(inventoryBtn) },
        new[] {
InlineKeyboardButton.WithCallbackData(backButton) }
    }
);

        public static InlineKeyboardMarkup accountMenuMarkup
= new InlineKeyboardMarkup(
    new[] {
        new[] {
InlineKeyboardButton.WithCallbackData(balanceBtn) },
        new[] {
InlineKeyboardButton.WithCallbackData(basketBtn) },
        new[] {
InlineKeyboardButton.WithCallbackData(historyBtn) },
        new[] {
InlineKeyboardButton.WithCallbackData(backButton) }
    }
);

        #endregion

        #region Account keyboards

```

```

        public static InlineKeyboardMarkup balanceMenuMarkup
= new InlineKeyboardMarkup(
            new[] {
                new[] {
InlineKeyboardButton.WithCallbackData(backButton) }
                }
            );

        public static InlineKeyboardMarkup basketMenuMarkup =
new InlineKeyboardMarkup(
            new[] {
                new[] {
InlineKeyboardButton.WithCallbackData(payBtn),
InlineKeyboardButton.WithCallbackData(delBtn),
InlineKeyboardButton.WithCallbackData(backButton) },
                }
            );

        #endregion

        #region Catalog keyboards

        public static InlineKeyboardMarkup clothesMenuMarkup
= new InlineKeyboardMarkup(
            new[] {
                new[] {
InlineKeyboardButton.WithCallbackData(shirtBtn) },
                new[] {
InlineKeyboardButton.WithCallbackData(shortBtn) },
                new[] {
InlineKeyboardButton.WithCallbackData(sneakersBtn) },
                new[] {
InlineKeyboardButton.WithCallbackData(catalogBtn) },
                new[] {
InlineKeyboardButton.WithCallbackData(backButton) }
            }
        );

        public static InlineKeyboardMarkup
nutritionMenuMarkup = new InlineKeyboardMarkup(
            new[] {
                new[] {
InlineKeyboardButton.WithCallbackData(bcaaBtn) },
                new[] {
InlineKeyboardButton.WithCallbackData(creatineBtn) },
                new[] {
InlineKeyboardButton.WithCallbackData(proteinBtn) },
            }
        );

```

```

        new[]
InlineKeyboardButton.WithCallbackData(vitaminsBtn) },
        new[]
InlineKeyboardButton.WithCallbackData(catalogBtn) },
        new[]
InlineKeyboardButton.WithCallbackData(backButton) }
    }
    );

    public static InlineKeyboardMarkup
inventoryMenuMarkup = new InlineKeyboardMarkup(
        new[] {
            new[]
InlineKeyboardButton.WithCallbackData(barbellsBtn) },
            new[]
InlineKeyboardButton.WithCallbackData(discsBtn) },
            new[]
InlineKeyboardButton.WithCallbackData(dumbbellsBtn) },
            new[]
InlineKeyboardButton.WithCallbackData(catalogBtn) },
            new[]
InlineKeyboardButton.WithCallbackData(backButton) }
        }
    );

    public static InlineKeyboardMarkup productMenuMarkup
= new InlineKeyboardMarkup(
        new[] {
            new[]
InlineKeyboardButton.WithCallbackData(buyBtn) },
            new[]
InlineKeyboardButton.WithCallbackData(backButton) }
        }
    );

    #endregion
}
}

```

Файл «IDatabaseHandler.cs»

```

using System;
using System.Collections.Generic;
using Telegram.Bot.Types.ReplyMarkups;

```

```

namespace SportShopBot.Entities
{
    internal interface IDatabaseHandler
    {
        /// <summary>
        /// Добавление новых записей в таблице
    }
}

```

```

        /// </summary>
        /// <param name="table">Имя таблицы</param>
        /// <param name="args">Поля таблицы</param>
        /// <param name="values">Значения полей
таблицы</param>
        void InsertData(string table, Dictionary<string,
string> args);

        /// <summary>
        /// Обновление данных в записях таблицы
        /// </summary>
        /// <param name="table">Имя таблицы</param>
        /// <param name="editColumn">Выбор колонки для
изменения</param>
        /// <param name="condition">Условие для изменения
данных в записи</param>
        void UpdateData(string table, string editColumn,
string condition);

        /// <summary>
        /// Удаление данных из таблицы
        /// </summary>
        /// <param name="table">Имя таблицы</param>
        /// <param name="condition">Условие для удаления
записи</param>
        int DeleteData(string table, string condition);

        /// <summary>
        /// Получение данных с условием из таблицы
        /// </summary>
        /// <param name="table">Имя таблицы</param>
        /// <param name="value">Тип получаемых данных</param>
        /// <param name="condition">Условие для удаления
записи</param>
        List<Dictionary<string, object>>
SelectWithConditionData(string table, string value, string
condition);

        /// <summary>
        /// Получение данных с условием из таблицы
        /// </summary>
        /// <param name="table">Имя таблицы</param>
        /// <param name="value">Тип получаемых данных</param>
        /// <param name="condition">Условие для удаления
записи</param>
        Dictionary<string, object>
SelectWithConditionRow(string table, string value, string
condition);

        /// <summary>

```

```

        /// Получение данных из таблицы
        /// </summary>
        /// <param name="table">Имя таблицы</param>
        /// <param name="value">Тип получаемых данных</param>
        /// <param name="condition">Условие для удаления
записи</param>
        List<Dictionary<string, object>> SelectData(string
table, string col);

        /// <summary>
        /// Получение данных о балансе
        /// </summary>
        /// <param name="name"></param>
        /// <returns></returns>
        string GetBalance(string name);

        /// <summary>
        /// Получение данных о продукции в корзине
        /// </summary>
        /// <param name="name"></param>
        /// <returns></returns>
        string GetBasket(string userId);

        /// <summary>
        /// Получение данных о продуктах определенного типа
        /// </summary>
        /// <param name="type"></param>
        /// <returns></returns>
        InlineKeyboardMarkup GetProducts(string type, string
backBtn);

        /// <summary>
        /// Добавление продукта в корзину
        /// </summary>
        /// <param name="userId"></param>
        /// <param name="message"></param>
        void SetBasketProduct(string userId, string message);

        /// <summary>
        /// Покупка продукции из корзины
        /// </summary>
        /// <param name="userId"></param>
        int SetDeal(string userId);

        /// <summary>
        /// Запрос истории сделок покупателя
        /// </summary>
        /// <param name="userId"></param>
        /// <returns></returns>
        string GetDeals(string userId);

```

```

    }
}

```

### Файл «DatabaseHandler.cs»

```

using Npgsql;
using System;
using System.Collections.Generic;
using System.Linq;
using Telegram.Bot.Types.ReplyMarkups;

namespace SportShopBot.Entities
{
    internal class DatabaseHandler : IDatabaseHandler
    {
        /// <summary>
        /// Дескриптор для работы с базой данных
        /// </summary>
        private NpgsqlConnectionStringBuilder _connStr;

        /// <summary>
        /// Конструктор
        /// </summary>
        /// <param name="connStr">Данные для подключения к
базе данных</param>
        public DatabaseHandler(string connStr) => _connStr =
new NpgsqlConnectionStringBuilder(connStr);

        #region Main requests

        public void InsertData(string table,
Dictionary<string, string> args)
        {
            var argsStr = String.Join(",", args.Keys);
            var valuesStr = String.Join("'", args.Values);

            using (var conn = new
NpgsqlConnection(_connStr.ToString()))
            {
                Console.Out.WriteLine("Opening connection");
                conn.Open();

                using (var command = new
NpgsqlCommand($"INSERT INTO {table} ({argsStr}) VALUES
('{valuesStr}')" , conn))
                {
                    int nRows = command.ExecuteNonQuery();

                    Console.Out.WriteLine(String.Format($"Number of rows
inserted={nRows}"));
                }
            }
        }
    }
}

```

```

    }
}

public void UpdateData(string table, string
editColumn, string condition)
{
    using (var conn = new
NpgsqlConnection(_connStr.ToString()))
    {
        Console.Out.WriteLine("Opening connection");
        conn.Open();

        using (var command = new
NpgsqlCommand($"UPDATE {table} SET {editColumn} WHERE {condition}",
conn))
        {
            int nRows = command.ExecuteNonQuery();

            Console.Out.WriteLine(String.Format($"Number of rows
update={nRows}"));
        }
    }

    public int DeleteData(string table, string condition)
    {
        using (var conn = new
NpgsqlConnection(_connStr.ToString()))
        {
            Console.Out.WriteLine("Opening connection");
            conn.Open();

            using (var command = new
NpgsqlCommand($"DELETE FROM {table} WHERE {condition}", conn))
            {
                int nRows = command.ExecuteNonQuery();

                Console.Out.WriteLine(String.Format($"Number of rows
delete={nRows}"));

                return nRows;
            }
        }
    }

    #endregion

    #region Requests with a condition

```

```

        public List<Dictionary<string, object>>
SelectWithConditionData(string table, string value, string
condition)
    {
        using (var conn = new
NpgsqlConnection(_connStr.ToString()))
        {
            Console.Out.WriteLine("Opening connection");
            conn.Open();

            using (var command = new
NpgsqlCommand($"SELECT {value} FROM {table} WHERE {condition}",
conn))
            {
                var listRow = new List<Dictionary<string,
object>>();
                var reader = command.ExecuteReader();

                while (reader.Read())
                {
                    var row = new Dictionary<string,
object>();
                    for (int i = 0; i < reader.FieldCount;
i++)
                    {
                        var name = reader.GetName(i);
                        var res = reader.GetValue(i);
                        row.Add(name, res);
                    }
                    listRow.Add(row);
                }

                return listRow;
            }
        }
    }

    public Dictionary<string, object>
SelectWithConditionRow(string table, string value, string
condition)
    {
        using (var conn = new
NpgsqlConnection(_connStr.ToString()))
        {
            Console.Out.WriteLine("Opening connection");
            conn.Open();

            using (var command = new
NpgsqlCommand($"SELECT {value} FROM {table} WHERE {condition}",
conn))

```

```

        {
            var listRow = new List<Dictionary<string,
object>>>();

            var reader = command.ExecuteReader();
            var row = new Dictionary<string,

object>();

            while (reader.Read())
            {
                for (int i = 0; i < reader.FieldCount;
i++)
                {
                    var name = reader.GetName(i);
                    var res = reader.GetValue(i);
                    row.Add(name, res);
                }
            }

            return row;
        }
    }

    public List<Dictionary<string, object>>
SelectData(string table, string col)
    {
        using (var conn = new
NpgsqlConnection(_connStr.ToString()))
        {
            Console.Out.WriteLine("Opening connection");
            conn.Open();

            using (var command = new
NpgsqlCommand($"SELECT {col} FROM {table}", conn))
            {
                var listRow = new List<Dictionary<string,
object>>>();

                var reader = command.ExecuteReader();

                while (reader.Read())
                {
                    var row = new Dictionary<string,

object>();

                    for (int i = 0; i < reader.FieldCount;
i++)
                    {
                        var name = reader.GetName(i);
                        var value = reader.GetValue(i);
                        row.Add(name, value);
                    }
                }
            }
        }
    }

```

```

        listRow.Add(row);
    }

    return listRow;
}

}

public string GetBalance(string name)
{
    var list = SelectWithConditionData("accounts",
"balance", $"name='{name}'");

    return $"<b>Баланс: {list[0]["balance"]}</b>";
}

public string GetBasket(string userId)
{
    var account = SelectWithConditionRow("accounts",
"*, $"id='{userId}'");
    var basket = SelectWithConditionRow("basket", "*",
$"account_id='{account["id"]}'");

    if (basket.Count == 0)
        return "<b>Корзина пуста</b>";

    var basketDetails =
SelectWithConditionData("basket_details",
"*,
$"basket_item_id='{basket["id"]}'");

    var res = "<b>Товары:</b>\n";
    foreach (var item in basketDetails)
    {
        var product =
SelectWithConditionRow("products",
"*,
$"id='{item["product_id"]}'");

        res += $"Тип товара: {product["type"]}\nФирма:
{product["name"]}\nЦена: {item["price"]}\nКоличество:
{item["count"]}\n\n";
    }

    return res ;
}

public InlineKeyboardMarkup GetProducts(string type,
string backBtn)
{
    var products =
SelectWithConditionData("products", "*", $"type='{type}'");

```

```

        if (products.Count == 0)
            return null;

        var listBtns = new List<InlineKeyboardButton[]>();
        foreach (var item in products)
        {
            listBtns.Add(new[] {
InlineKeyboardButton.WithCallbackData(item["name"].ToString()) });
        }
        listBtns.Add(new[] {
InlineKeyboardButton.WithCallbackData(backBtn) });

        return new InlineKeyboardMarkup(listBtns);
    }

    public void SetBasketProduct(string userId, string
message)
    {
        var data = message.Split('\n');
        var price = Convert.ToDouble(data.Last().Split('
').Last());

        // Проверка на наличие корзины у пользователя
        var basket = SelectWithConditionRow("basket", "*",
$"account_id='{userId}'");
        if (basket.Count == 0)
        {
            InsertData("basket", new Dictionary<string,
string>
            {
                { "account_id", userId },
                { "price", $"{price}" }
            });
        }
        else
        {
            var
                newPrice
            =
            Convert.ToDouble(basket["price"]) + price;
            UpdateData("basket", $"price='{newPrice}'",
$"account_id='{userId}'");
        }

        var
            name
        =
        data.First(p
            =>
            p.StartsWith("Фирма")).Split(':').Last().Trim();
        var product = SelectWithConditionRow("products",
        "*", $"name='{name}'");
        var basketItem = SelectWithConditionRow("basket",
        "*", $"account_id='{userId}'");
    }

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.49.001 ПЗ | Лист |
|      |      |          |         |      |                       | 67   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

        // Проверка на наличие в базе деталей корзины
        var basketDetailsItem =
SelectWithConditionRow("basket_details", "*",
$"product_id={product["id"]}" );
        if (basketDetailsItem.Count == 0)
        {
            // Создание записи о деталях товара в корзине
            InsertData("basket_details", new
Dictionary<string, string>
            {
                { "product_id", $"{product["id"]}" },
                { "basket_item_id",
$" {basketItem["id"]} ",
                { "count", "1" },
                { "price", $"{price}" }
            });
        }
        else
        {
            // Обновленме данных о деталях товара в
корзине
            var count =
Convert.ToDouble(basketDetailsItem["count"]) + 1;
            var newPrice =
Convert.ToDouble(basketDetailsItem["price"]) + price;
            UpdateData("basket_details",
$"count='{count} ', price='{newPrice} ',
$"id='{basketDetailsItem["id"]}'");
        }
    }

    public int SetDeal(string userId)
    {
        // Получение данных о корзине товаров
        var basket = SelectWithConditionRow("basket", "*",
$"account_id='{userId}'");

        if (basket.Count == 0) return 0;

        // Занесение данных в базу о сделке
        var time = DateTime.Now;
        InsertData("deals", new Dictionary<string,
string>
        {
            { "account_id", userId },
            { "price",
$" {Convert.ToDouble(basket["price"])} ",
            { "time_deal", $"{time}" }
        });
    }

```

```

        // Занесение данных в базу о деталях сделки
        var basketItems =
SelectWithConditionData("basket_details", "*",
$"basket_item_id='{basket["id"]}'");
        var deal = SelectWithConditionRow("deals", "*",
$"account_id='{userId}' AND time_deal='{time}'");
        foreach (var item in basketItems)
            InsertData("deal_details", new
Dictionary<string, string>
            {
                { "product_id", $"{item["product_id"]}"
            },
            { "deal_id", $"{deal["id"]}" },
            { "count", $"{item["count"]}" },
            { "price",
$" {Convert.ToDouble(item["price"])}" }
            });

        // Начисление бонусных баллов покупателю
        var user = SelectWithConditionRow("accounts", "*",
$"id='{userId}'");
        var balance = Convert.ToDouble(user["balance"]) +
Convert.ToDouble(basket["price"]) * 0.05;
        UpdateData("accounts", $"balance='{balance}'",
$"id='{userId}'");

        // Очистка корзины
        return DeleteData("basket",
$"id='{basket["id"]}'");
    }

    public string GetDeals(string userId)
    {
        var deals = SelectWithConditionData("deals", "*",
$"account_id='{userId}'");

        if (deals.Count == 0)
            return "<b>Нет истории</b>";

        var res = "<b>История:</b>\n";
        foreach (var deal in deals)
        {
            res += $"Цена: {deal["price"]}\nДата и время:
{deal["time_deal"]}\n\n";
        }

        return res;
    }
}

```

```

        #endregion
    }

    Файл «AIHelper.cs»
    using Newtonsoft.Json.Linq;
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Net;
    using System.Net.Http;
    using System.Text;
    using System.Threading.Tasks;

    namespace SportShopBot.Entities
    {
        internal class AIHelper
        {
            internal class JsonObject
            {
                public string model { get; set; }
                public Dictionary<string, string> messages { get;
set; }
            }

            private readonly HttpClient _httpClient;

            private readonly string _apiKey;

            public AIHelper(HttpClient httpClient, string apiKey)
            {
                _httpClient = httpClient;
                _apiKey = apiKey;
            }

            public async Task<string> GetAnswer(string question)
            {
                string questionTemplate = @"{
                    ""model"": ""meta-llama/llama-3.3-8b-instruct:free"",
                    ""messages"": [
                        {
                            ""role"": ""system"",
                            ""content"": ""You are a fitness
trainer.""
                        },
                        {
                            ""role"": ""user"",
                            ""content"": [
                                {

```

```

        ""type"": ""text"",
        ""text"": ""{0}""
    }
    ],
    },
    ],
    ""max_tokens"": 150
};
string jsonContent =
questionTemplate.Replace("{0}", question);

var request = new HttpRequestMessage
{
    Method = HttpMethod.Post,
    RequestUri = new
Uri("https://openrouter.ai/api/v1/chat/completions"),
    Headers =
    {
        { HttpRequestHeader.ContentType.ToString(),
"application/json" },
        { HttpRequestHeader.Authorization.ToString(),
"Bearer " + _apiKey},
    },
    Content = new StringContent(jsonContent,
Encoding.UTF8, "application/json")
};

var response = await
_httpClient.SendAsync(request);
string answer = await
GetClearAnswerFromResponse(response);

return answer;
}

public async Task<string>
GetClearAnswerFromResponse(HttpResponseMessage response)
{
    if (response.IsSuccessStatusCode)
    {
        var responseContent = await
response.Content.ReadAsStringAsync();
        var parsedResponse =
JsonObject.Parse(responseContent);

        var choices = parsedResponse["choices"];
        if (choices == null || choices.Count() == 0 ||
choices[0] == null)
            return string.Empty;
    }
}

```

```

        var message = choices?[0]?["message"];
        if (message == null || message["content"] ==
null)

            return string.Empty;

        var answer = message["content"]?.ToString();

        return answer;
    }

    var          errorContent          =          await
response.Content.ReadAsStringAsync();
    return $"Error: {response.StatusCode}, Content:
{errorContent}";
    }
}

```

#### Файл «SportShopTables.sql»

```

create table accounts (
    id bigint primary key,
    name varchar(50) not null,
    balance numeric(9,2)
);

create table products (
    id serial primary key,
    type varchar(50) not null,
    name varchar(50) not null,
    size varchar(50) not null,
    price numeric(9,2)
);

create table deals (
    id serial primary key,
    account_id integer references accounts(id),
    price numeric(9,2),
    time_deal timestamp
);

create table deal_details (
    id serial primary key,
    product_id integer references products(id),
    deal_id integer references deals(id) on delete cascade,
    count integer,
    price numeric(9,2)
);

create table basket (
    id serial primary key,

```

```

        account_id integer references accounts(id),
        price numeric(9,2)
    );

    create table basket_details (
        id serial primary key,
        product_id integer references products(id),
        basket_item_id integer references basket(id) on delete
cascade,
        count integer,
        price numeric(9,2)
    );

```

### Файл «SportShopInserData.sql»

```

        insert into products(type, name, size, price)
values('shirt', 'Nike SWOOSH', 'XXS-XXL', '500');
        insert into products(type, name, size, price) values('shirt',
'Adidas Originals', 'XXS-XXL', '700');
        insert into products(type, name, size, price) values('shirt',
'Reebok GS VECTOR', 'XXS-XXL', '600');

        insert into products(type, name, size, price)
values('sneakers', 'Adidas ECLYPTIX', '40-47', '6800');
        insert into products(type, name, size, price)
values('sneakers', 'Nike Air Max', '40-47', '9000');
        insert into products(type, name, size, price)
values('sneakers', 'ASICS JOLT5', '40-47', '8900');

        insert into products(type, name, size, price) values('short',
'Adidas D4T SHORT', 'XXS-XXL', '4000');
        insert into products(type, name, size, price) values('short',
'Nike TOTALITY', 'XXS-XXL', '5000');
        insert into products(type, name, size, price) values('short',
'Puma ESS 2 COLOR', 'XXS-XXL', '3900');

        insert into products(type, name, size, price) values('bcaa',
'BCAA aTech', '-', '850');
        insert into products(type, name, size, price) values('bcaa',
'BCAA Overvit', '-', '550');

        insert into products(type, name, size, price)
values('creatine', 'Creatine SOLAB', '-', '500');
        insert into products(type, name, size, price)
values('creatine', 'Creatine Overvit', '-', '700');

        insert into products(type, name, size, price)
values('protein', 'Protein SOLAB', '-', '500');
        insert into products(type, name, size, price)
values('protein', 'Protein 1WIN', '-', '520');

```

```

insert into products(type, name, size, price)
values('vitamins', 'NOW Foods Daily Vits', '-', '500');
insert into products(type, name, size, price)
values('vitamins', 'Maxler VitaMen', '-', '520');

insert into products(type, name, size, price)
values('barbell', 'Torneo 30x1200', '-', '3600');
insert into products(type, name, size, price)
values('barbell', 'KETTLER 30x1200', '-', '20000');

insert into products(type, name, size, price) values('disc',
'Torneo 2.5kg', '-', '600');
insert into products(type, name, size, price) values('disc',
'Torneo 5kg', '-', '1000');
insert into products(type, name, size, price) values('disc',
'Torneo 10kg', '-', '1600');
insert into products(type, name, size, price) values('disc',
'Torneo 15kg', '-', '2400');
insert into products(type, name, size, price) values('disc',
'Torneo 20kg', '-', '3000');

insert into products(type, name, size, price)
values('dumbbells', 'Demix 0.5kg', '-', '300');
insert into products(type, name, size, price)
values('dumbbells', 'Demix 1kg', '-', '500');
insert into products(type, name, size, price)
values('dumbbells', 'Demix 2kg', '-', '700');
insert into products(type, name, size, price)
values('dumbbells', 'KETTLER 5kg', '-', '2200');
insert into products(type, name, size, price)
values('dumbbells', 'KETTLER 10kg', '-', '6000');
insert into products(type, name, size, price)
values('dumbbells', 'KETTLER 15kg', '-', '7000');

```

## Приложение В

(Обязательное)

Телеграмм–бот «Фитнес–магазин»

Презентация

|      |      |          |         |      |                              |      |
|------|------|----------|---------|------|------------------------------|------|
|      |      |          |         |      | <i>ПГУ1.090301.49.001 ПЗ</i> | Лист |
|      |      |          |         |      |                              | 75   |
| Изм. | Лист | № докум. | Подпись | Дата |                              |      |

# Выпускная квалификационная работа на тему: Телеграмм-бот «Фитнес-магазин»

Студент гр. 21ВВП2: Чинов Даниил Дмитриевич  
Руководитель: к. т. н., доцент кафедры «Вычислительная Техника» -  
Егоров Валерий Юрьевич

ПГУ 2025

## Введение

Цель работы – разработка Телеграмм-бота на C#.

Задачи:

- Регистрация пользователя
- Витрина товаров
- Корзина
- История покупок

Объект исследования — разработка Телеграмм-бот

Предмет исследования — готовой бот разработанный на C#

2

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.49.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 76   |

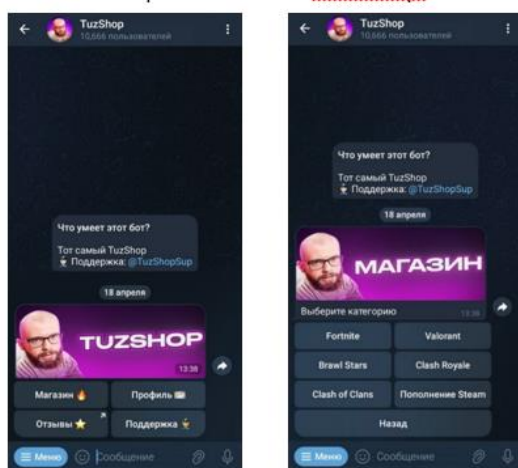
## Актуальность проекта

Современные технологии меняют подходы к ведению бизнеса, и одним из ярких примеров этого является использование мессенджеров для автоматизации взаимодействия с клиентами. Телеграм-боты становятся все более популярными, поскольку они предлагают удобный способ общения и покупок. В контексте фитнес-магазинов, такие боты могут значительно улучшить клиентский опыт и оптимизировать бизнес-процессы.

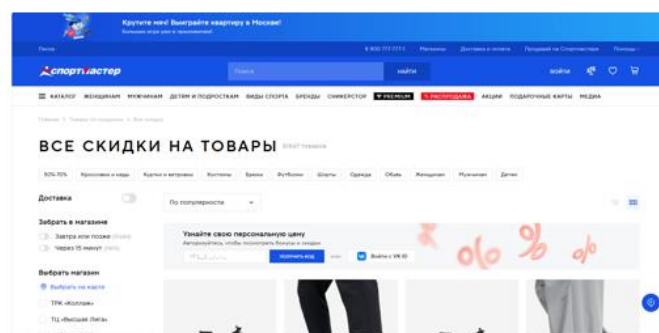
3

## Обзор аналогов

Телеграмм-бот «TuzShop»



Веб-сайт «Спортмастер»



4

|      |      |          |         |      |
|------|------|----------|---------|------|
|      |      |          |         |      |
| Изм. | Лист | № докум. | Подпись | Дата |

ПГУ1.090301.49.001 ПЗ

Лист

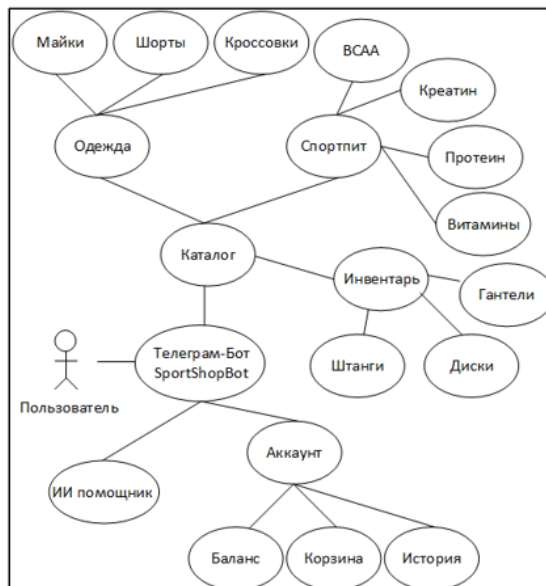
77

## Выбор инструментов разработки



5

## UML диаграмма прецедентов



6

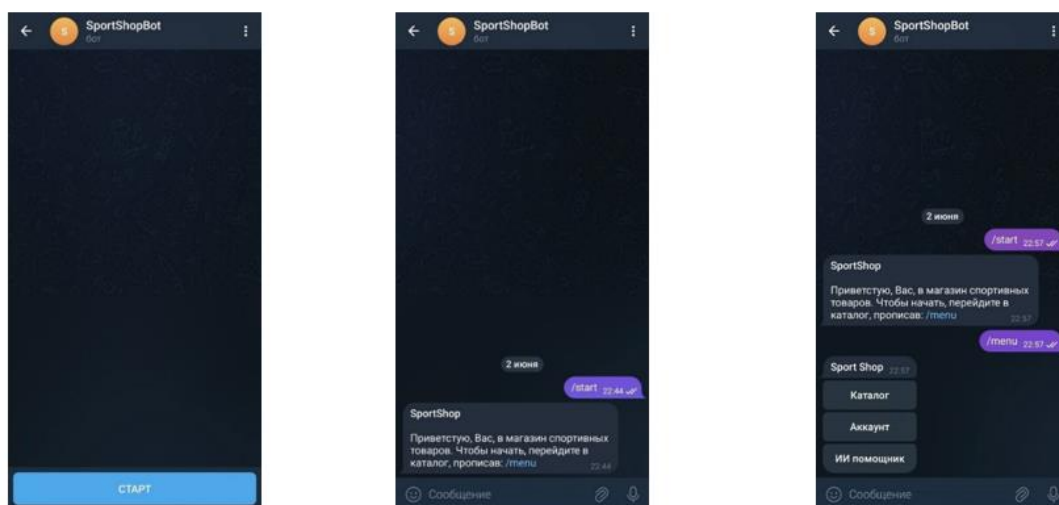
|      |      |          |         |      |
|------|------|----------|---------|------|
|      |      |          |         |      |
| Изм. | Лист | № докум. | Подпись | Дата |

ПГУ1.090301.49.001 ПЗ

Лист

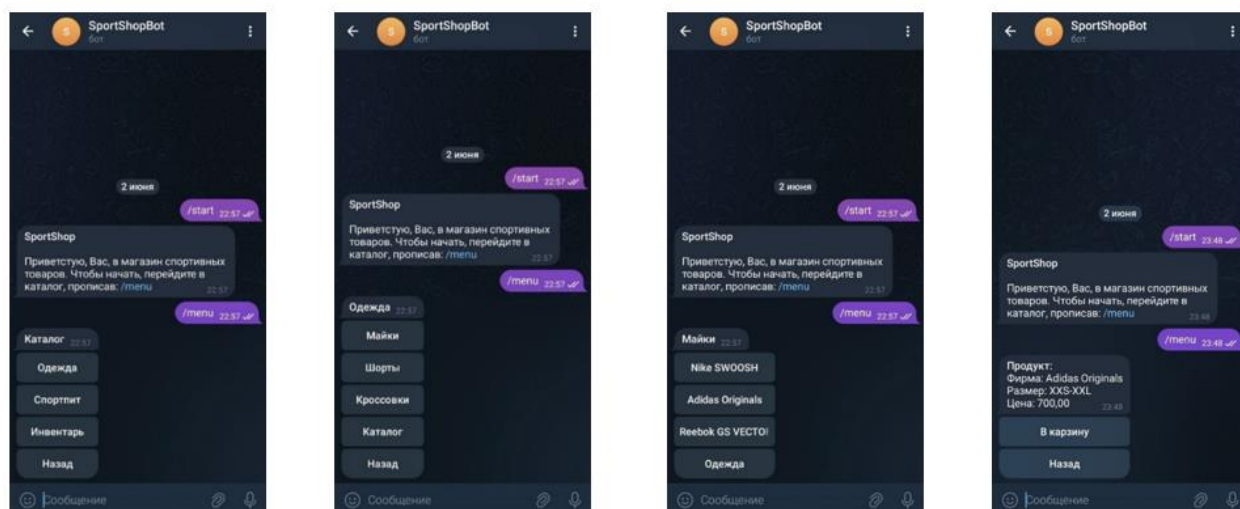
78

## Телеграмм-бот «Фитнес-магазин»



7

## Телеграмм-бот «Фитнес-магазин»



8

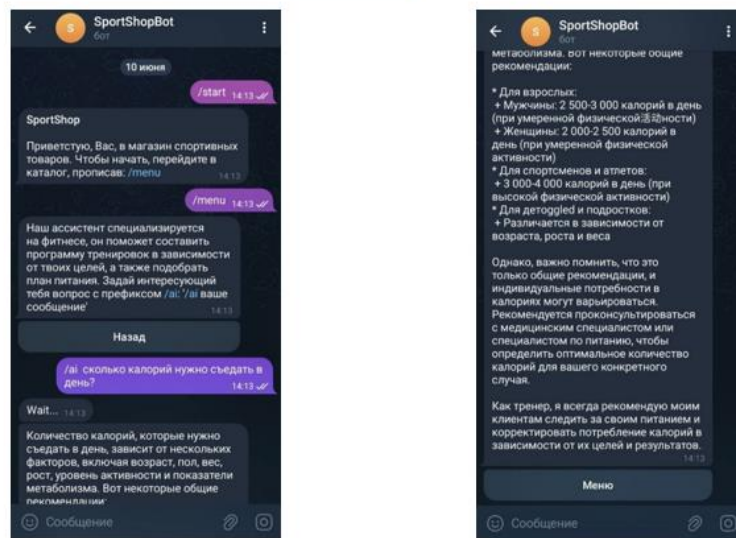
|      |      |          |         |      |
|------|------|----------|---------|------|
|      |      |          |         |      |
| Изм. | Лист | № докум. | Подпись | Дата |

ПГУ1.090301.49.001 ПЗ

Лист

79

## Телеграмм-бот «Фитнес-магазин»



9

## Заключение

- ✓ Проведён анализ предметной области и существующих аналогов.
- ✓ Выполнено проектирование структуры приложения.
- ✓ Осуществлена программная реализация.
- ✓ Проведено комплексное тестирование.

10

|      |      |          |         |      |
|------|------|----------|---------|------|
|      |      |          |         |      |
| Изм. | Лист | № докум. | Подпись | Дата |

ПГУ1.090301.49.001 ПЗ

Лист

80