Министерство образования и науки Республики Казахстан

Некоммерческое акционерное общество «Торайгыров университет»

**ДИПЛОМНЫЙ ПРОЕКТ**

**Создание информационно аналитической системы по потреблению основных продуктов питания разными слоями населения на основе открытых данных РК**

По образовательной программе 5В070300 – «Информационные системы»

Павлодар

2022Министерство образования и науки Республики Казахстан

Некоммерческое акционерное общество «Торайгыров университет»

Факультет Computer Science

Допущен(а) к защите

Руководитель ОП \_\_\_\_\_\_\_\_\_\_\_\_ Н. Н. Оспанова

(подпись)

\_\_ \_\_\_\_\_\_\_\_\_ 20\_\_года

**ДИПЛОМнЫЙ ПРОЕКТ**

**Создание информационно аналитической системы по потреблению основных продуктов питания разными слоями населения на основе открытых данных РК**

**по образовательно программе 5В070300 – Информационные системы**

**Выполнил(а)** \_\_\_\_\_\_\_\_\_\_\_\_ Д. В. Шухарт

(подпись)

Группа ИС-402

**Научный руководитель**

магистр, ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_ С. Н. Талипов

(подпись)

**Нормоконтроль**

Ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_ А. З. Даутова

(подпись)

Павлодар

2022

|  |  |  |
| --- | --- | --- |
| Задание  на выполнение ДР/П | лого 2020 | Ф МИ СМК 4.01.6/02 |

Министерство образования и науки Республики Казахстан

Некоммерческое акционерное общество «Торайгыров университет»

Факультет «Computer Science»

Специальность Информационные системы

**Утверждаю**

Руководитель образовательных программ канд. Пед. наук, профессор

Н. Н. Оспанова

20 года

**ЗАДАНИЕ**

на выполнение дипломной (го) работы/проекта

Обучающемуся Шухарт Диане Владимировне

Тема ДР/П Создание информационно аналитической системы по потреблению основных продуктов питания разными слоями населения на основе открытых данных РК

утверждена приказом по университету от 20 года №

Срок сдачи ДР/П 20 года

Исходные данные к ДР/П Методические рекомендации по выполнению ДР/П

Перечень подлежащих разработке в ДР/П вопросов или краткое содержание

выпускной работы (дипломной работы/дипломного проекта):

1 Описание и сравнительный анализ средств для разработки

2 Обзор возможностей TELEGRAM

3 Чат бот. Основные понятия. Функции. Виды

4 Выбор среды разработки и языка программирования

5 Общий алгоритм реализации проекта

6 Целевая аудитория

7 Постановка цели

8 Актуальность проекта

|  |  |  |
| --- | --- | --- |
| Задание  на выполнение ДР/П | лого 2020 | Ф МИ СМК 4.01.6/02 |

9 Разработка чат бота

10 Архитектура чат бота

11 Регистрация чат бота в TELEGRAM

12 Api

13 Добавление библиотек

14 Написание программы

Перечень графического материала с точным указанием обязательных чертежей:38

Рекомендуемая основная литература: 5

**ГРАФИК**

подготовки дипломной работы/проекта

|  |  |  |
| --- | --- | --- |
| Наименование разделов, перечень  разрабатываемых вопросов | Сроки представления научному  руководителю и консультантам | Подпись |
| Описание и сравнительный анализ средств для разработки |  |  |
| Постановка цели |  |  |
| Разработка чат бота |  |  |

**Подписи**

консультантов на законченную дипломную работу (проект), с указанием относящихся к ним разделов ДР/П

|  |  |  |  |
| --- | --- | --- | --- |
| Наименования разделов | Консультанты, Ф.И.О.  (уч. степень, звание) | Дата  подписания | Подпись |
| Описание и сравнительный анализ средств для разработки |  |  |  |
| Постановка цели |  |  |  |
| Разработка чат бота |  |  |  |

Научный руководитель С. Н. Талипов

(подпись) (инициалы, фамилия)

Задание принял к исполнению

обучающийся Д. В. Шухарт

(подпись) (инициалы, фамилия)

Дата 20\_\_года

**Содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
| 1 | Описание и сравнительный анализ средств для разработки | 6 |
| 1.1 | Обзор возможностей SPA | 6 |
| 1.2 | Технологии используемые в проекте | 7 |
| 1.3 | Выбор среды разработки | 7 |
| 1.4 | Общий алгоритм реализации проекта | 8 |
| 2. | Постановка цели | 9 |
| 2.1 | Актуальность проекта | 9 |
| 2.2 | Цель проекта | 9 |
| 2.3 | Целевая аудитория |  |
| 3 | Разработка сайта | 10 |
| 3.1 | Архитектура проекта | 10 |
| 3.2 | Создание базы данных для проекта | 10 |
| 3.3 | Api | 13 |
| 3.4 | Добавление библиотек (аксиос, роутер дом) | 14 |
| 3.5 | Написание программы | 16 |
|  | Заключение | 29 |
|  | Список использованных источников | 30 |
|  | Приложение А | 31 |

**Введение**

В наше время, сотни и тысячи предприятий в каждом городе создают сотни и тысячи рабочих мест. В связи с этим, иметь систему, которая бы содержала информацию о всех предприятиях, было бы крайне полезно для широкого круга пользователей. В данном дипломном проекте мы предлагаем разработать справочно-информационную систему для города Павлодар, в которой будут содержаться данные о предприятиях. Эта система сможет облегчить процесс поиска нужного предприятия, благодаря единому и удобному интерфейсу и хранению данных.

Цель данного проекта - создать единую справочную систему для города Павлодар, в которой будут храниться данные о всех предприятиях, такие как название, описание, имя руководителя и товары. Благодаря этой системе пользователи смогут быстро и удобно находить необходимые компании, ознакамливаться с их основными характеристиками и связываться с ними.

Для реализации проекта использовались следующие технологии: React - популярная библиотека JavaScript для создания пользовательских интерфейсов; TypeScript - язык программирования, предназначенный для разработки крупных проектов; JsonServer - простой сервер, который позволяет быстро создавать и запускать прототипы API.

Благодаря применению этих технологий, мы сможем создать удобную, быструю и надежную справочно-информационную систему для города Павлодар.

1. **Описание и сравнительный анализ средств для разработки**

**1.1 Обзор возможностей SPA**

SPA (Single Page Application) - это веб-приложение, которое работает в одной веб-странице, без перезагрузки страницы в ответ на действия пользователя. Вместо этого, SPA использует AJAX (Asynchronous JavaScript and XML) для загрузки данных и обновления содержимого на странице, без необходимости полной перезагрузки.

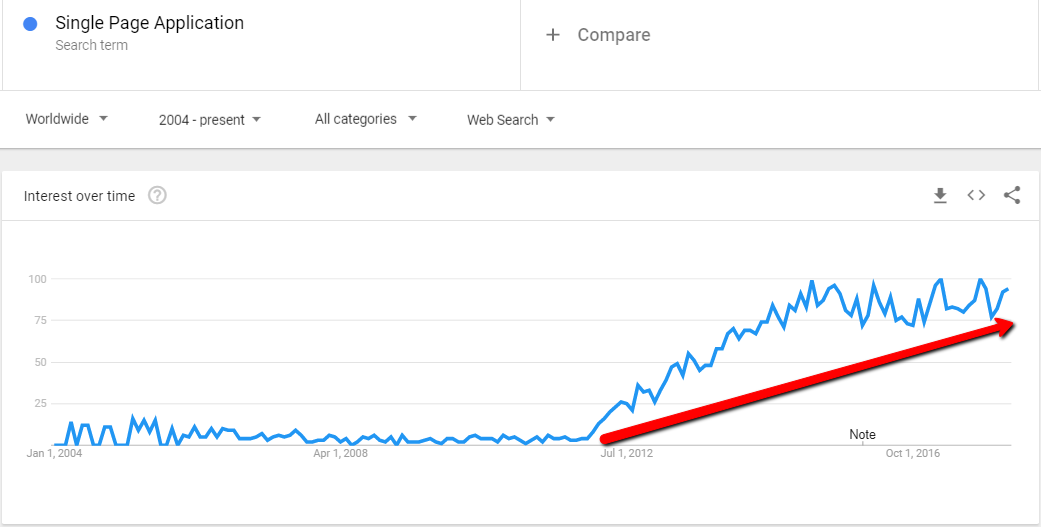


Рисунок 1 − График развития популярности SPA

SPA имеет множество преимуществ перед традиционными многостраничными приложениями. Вот некоторые из них:

- быстрая загрузка: SPA загружает только необходимые ресурсы при первоначальной загрузке страницы, что сокращает время загрузки страницы и улучшает производительность приложения.

- плавный пользовательский интерфейс: SPA использует AJAX для обновления содержимого на странице, без необходимости полной перезагрузки, что делает пользовательский интерфейс более плавным и реактивным.

- легкость разработки: SPA использует фронтенд-технологии, такие как JavaScript, для разработки пользовательского интерфейса, что делает его более простым в разработке и обслуживании.

- лучшее управление состоянием: SPA использует механизмы маршрутизации для управления состоянием приложения, что делает его более гибким и удобным для использования.

- лучшее взаимодействие с пользователем: SPA может использовать AJAX для взаимодействия с пользователем, что делает его более дружелюбным и удобным в использовании.

В целом, SPA предоставляет более современный, быстрый и гибкий подход к разработке веб-приложений. Он позволяет создавать более реактивные и дружелюбные пользовательские интерфейсы, обеспечивает лучшее управление состоянием и улучшает производительность приложения. Это делает SPA популярным выбором для разработки веб-приложений.

**1.2 Технологии используемые в проекте**

Для создания справочно-информационной системы для города Павлодар будут использоваться технологии React, TypeScript и Json-server.

React - это JavaScript-библиотека для разработки пользовательских интерфейсов. Она позволяет создавать динамические и переиспользуемые компоненты, что упрощает разработку и снижает сложность кода. React используется для построения фронтенда справочно-информационной системы, в котором будут отображаться данные о предприятиях.

TypeScript - это язык программирования, являющийся надмножеством JavaScript. Он добавляет статическую типизацию, что позволяет выявлять ошибки в коде на этапе компиляции, а не во время выполнения программы. TypeScript используется для улучшения качества кода и облегчения его поддержки.

Json-server - это утилита для создания фейкового сервера с REST API на основе файла с данными в формате JSON. Она позволяет быстро и легко создать сервер, не используя настоящую базу данных. Json-server используется для создания серверной части справочно-информационной системы, где будут храниться данные о предприятиях в формате JSON.

Таким образом, использование React и TypeScript позволит создать динамический и удобный пользовательский интерфейс, а Json-server обеспечит хранение данных и доступ к ним через API. Все эти технологии работают вместе для создания эффективной и удобной справочно-информационной системы

**1.3 Выбор среды разработки**

IntelliJ IDEA - это мощная и интуитивно понятная интегрированная среда разработки, предназначенная для разработки приложений на различных языках программирования, включая Java, JavaScript и Python. Среда разработки была разработана и выпущена компанией JetBrains еще в 2001 году, и с тех пор она стала одним из наиболее популярных инструментов разработки программного обеспечения.

Одной из главных причин выбора IntelliJ IDEA для данного проекта является его поддержка нескольких языков программирования, что обеспечивает гибкость и масштабируемость при разработке проекта. Кроме того, среда разработки обладает высокой производительностью и предоставляет разработчикам множество инструментов для улучшения их производительности и эффективности.

IntelliJ IDEA также имеет встроенную поддержку многих популярных фреймворков и библиотек, что значительно облегчает процесс разработки. Среда разработки также предоставляет широкий спектр функций, включая интеллектуальную помощь в написании кода, автодополнение, рефакторинг кода и тестирование, что обеспечивает высокую точность и надежность

* 1. **Общий алгоритм реализации проекта**

Перед началом работы нужно было определиться с языком программирования, в начале стоял выбор между PHP и React.

React TS (TypeScript) и PHP являются двумя различными языками программирования, которые могут использоваться для разработки веб-приложений. Они имеют разные принципы работы и обладают своими преимуществами и недостатками.

React TS предоставляет возможность строгой типизации, что позволяет уменьшить количество ошибок в процессе разработки и обеспечить более надежное приложение. PHP, с другой стороны, является слабо типизированным языком, что может привести к возникновению ошибок во время выполнения программы. Также React TS позволяет использовать современные технологии разработки, такие как компонентный подход, маршрутизация, серверный рендеринг и т.д. Поэтому был выбран этот язык программирования.

После нужно было изучить документацию, статьи и другие материалы, связанные с React, чтобы получить необходимые знания для реализации проекта.

После изучения можно приступать к построению логики и написанию самого кода.

**2 Постановка цели**

**2.1 Актуальность проекта**

Актуальность данного проекта заключается в том, что в настоящее время все больше людей и компаний используют Интернет для поиска информации о товарах и услугах. В связи с этим, важно иметь современную и удобную систему, которая позволяет быстро и эффективно находить необходимую информацию о компаниях и их продуктах.

Создание единой справочной системы для города Павлодар, в которой будут содержаться данные о предприятиях, поможет сделать процесс поиска компаний и их продуктов более удобным и быстрым для жителей и гостей города. Кроме того, такая система позволит предприятиям продвигать свои товары и услуги, улучшить свою видимость и привлечь новых клиентов.

Использование технологий React и TypeScript обеспечивает высокую производительность и надежность системы, а Json-server позволяет быстро и легко создавать и поддерживать базу данных предприятий. Кроме того, данные технологии позволяют создавать качественный пользовательский интерфейс и обеспечивать удобство использования системы.

**2.2 Цель проекта**

Цель проекта заключается в создании единой справочно-информационной системы для города Павлодар, в которой будут содержаться данные о предприятиях, такие как название, адрес, контакты, описание компании, имя руководителя и товары. Основная задача системы - облегчить процесс поиска компаний и их продуктов для жителей и гостей города, а также предоставить предприятиям возможность продвигать свои товары и услуги.

**2.3 Целевая аудитория**

Целевая аудитория проекта - это жители и гости города Павлодар, которые ищут информацию о компаниях и их продуктах. Также, целевой аудиторией являются предприятия города, которые хотят продвигать свои товары и услуги и улучшить свою видимость среди потенциальных клиентов.

**3 Разработка сайта**

**3.1 Архитектура проекта**

Архитектура проекта справочно-информационной системы преприятий включает в себя следующие компоненты:

Клиентская часть - это интерфейс, который видят пользователи системы. Она разработана с помощью фреймворка React и TypeScript. Клиентская часть обеспечивает удобный пользовательский интерфейс, включая поиск по предприятиям, просмотр детальной информации о каждой компании, а также возможность связаться с представителями предприятий.

Серверная часть - это RESTful API, созданный с помощью фреймворка Json-server. Она обеспечивает хранение и обработку данных, связанных с предприятиями. Серверная часть обеспечивает быстрый доступ к данным и возможность быстрой масштабируемости системы.

База данных - это JSON файл, содержащий информацию о предприятиях, такую как название, контакты, адрес, описание, имя руководителя и товары. База данных хранится на сервере и обновляется автоматически при добавлении или изменении информации о предприятиях.

Таким образом, архитектура проекта обеспечивает удобный пользовательский интерфейс, быстрый доступ к данным, а также защиту данных пользователей. Она является масштабируемой и может быть легко адаптирована для изменения требований и потребностей пользователей.

**3.2 Создание базы данных для проекта**

Перед началом создания проекта необходимо создать базу данных где будет храниться вся информация предприятий, для этого используется используется JSON файл. В качестве инструмента для создания и управления JSON-файлом используется фреймворк Json-server. Этот инструмент обеспечивает быстрый доступ к данным, а также предоставляет API для обработки запросов на чтение и запись данных в файл.

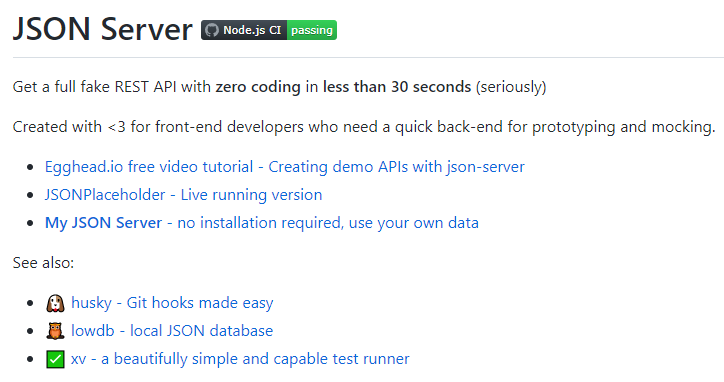


Рисунок 2 − Библиотека Json-server

Данный бот это самый простой и единственный способ регистрации, настройки и управления другими telegram – ботами на данной площадке. Работа с ним не требует особых навыков, с его помощью можно зарегистрировать неограниченное количество новых ботов

Данным ботом можно управлять с помощью команд, которые представлены на рисунке 3

Для создания нового бота нужно ввести команду /start, изображенную на рисунке 3.

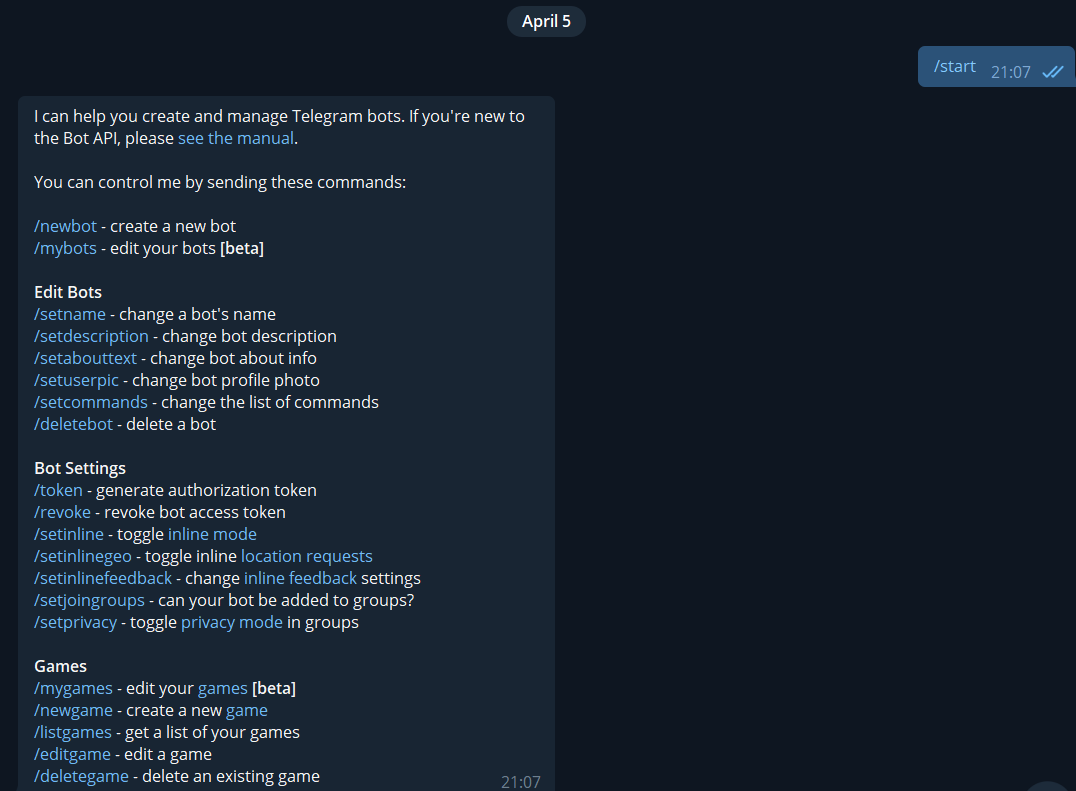


Рисунок 3 − Команда /start

Следующим шагом идет создание имени бота и его ник, это продемонстрировано на рисунке 4, который необходим для поиска среди других пользователей

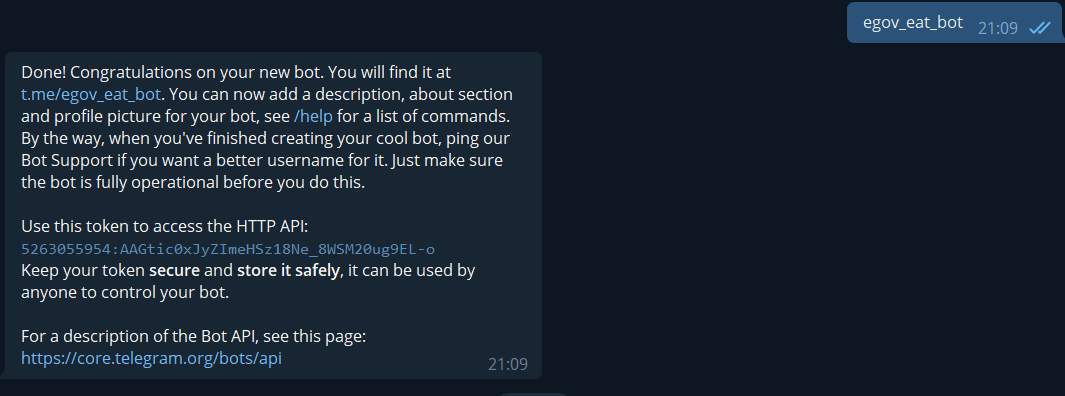


Рисунок 4 − Создание имени бота

Далее мы прописываем описание, при необходимости и команды, команды используемые в данном проекте представлены на рисунке 5

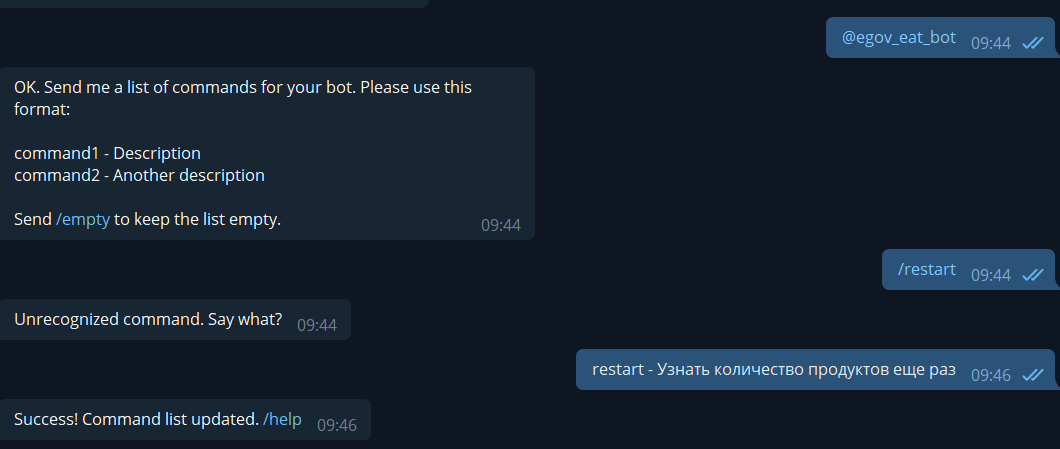


Рисунок 5 − Создание команды

Также бот дает уникальный ключ, изображенный на рисунке 6, который в дальнейшем будет осуществлять связь между пользователем и сервером

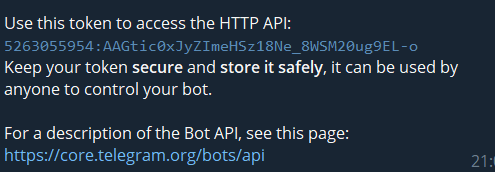


Рисунок 6 − Уникальный ключ

**3.3 API**

Application programming interface, или API это интерфейс, при помощи которого, сервер взаимодействуют с другим сервером, в данном случае это сервер egov с открытыми данными.

Открытые данные предоставленные на данном сайте, являются частью инициативы правительства Республики казахстан и являются отдельным элементом «Электронного правительства» Республики Казахстан.

При помощи «открытых данных», любой желающий может без проблем находить интересующую его информацию, которые разбиты по разным темам и разделам, и предоставляются государственными органами.

Были использованы данные по потреблению основных продуктов питания разными слоями населения на основе открытых данных Республики К

Это специальный набор способов и правил, с помощью которых программы обмениваются данными между собой, и в этом им помогают функции, классы, структуры, к которым обращаются программы.

В случае с данным проектом сервис Telegram обменивается данными с сервисом EGOV представленный на Рисунке 7



Рисунок 7 − Сервис EGOV

Telegram применяет личный протокол шифрования MTProto. MTProto API или же Telegram API − через данное API приложение Telegram связывается с сервером, за счет того, что Telegram API полностью открыт, любой разработчик может без проблем написать свой клиент мессенжер.

Для написания ботов в Telegram был создан специальный TelegramBot Api, чтобы воспользоваться им не нужно знать о том, как он работает, ведь протокол шифрования, указанный выше. Будет сам обрабатывать все шифрование и связь с Telegram API.

**3.4 Добавление библиотек**

Несмотря на то, что множество задач при написании кода можно решить при помощи стандартных средств InteliJIdea, в данном дипломном проекте использовались сторонние библиотеки, так как это было не просто полезно, а необходимо.

Библиотеки используются для упрощения написания кода, работы программы они избегают нагромождения функций в коде и использование сложных конструкций.

В данном случае можно было бы не использовать сторонние библиотеки, но в таком случае были бы сложности с написанием кода, а библиотеки упростили синтаксис программы, и тем самым всю работу.

В данном проекте использовалась библиотека Telegram Bot Java Library. В базовый класс, который реализовывает все взаимодействие бота с пользователем и с Telegram.

Работает данная библиотека с помощью технологии LongPolling, так как он не требует каких либо сертификатов шифрования и одним из его плюсов является запуск программного кода на любом устройстве.

Для корректной работы были использованы методы, представленные на рисунке 8, которые наследуются от класса TelegramLongPollingBot. Такие как:

- public void onUpdateReceived(Update update)

- public String getBotUsername()

- public String getBotToken()

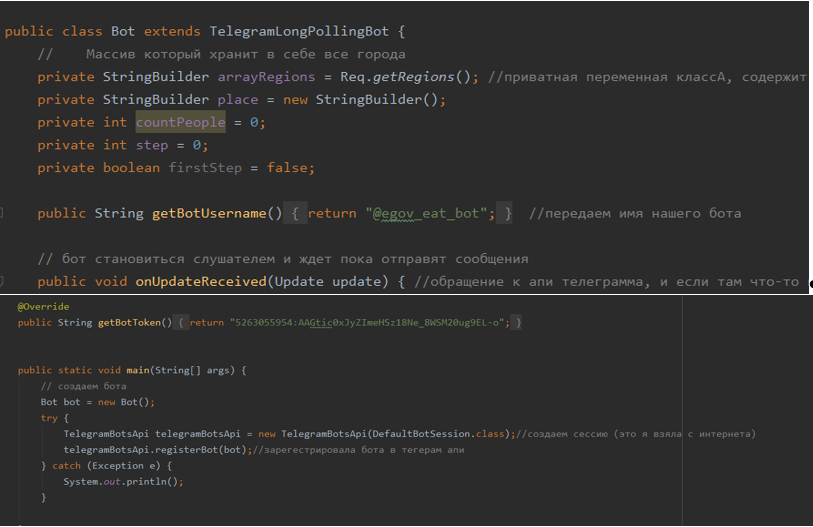


Рисунок 8 − методы класса

**3.5 Написание программы**

В начале написания кода был создан класс Bot, изображённый на рисунке 9, который наследуется от класса используемой библиотеки TelegramLongPollingBot



Рисунок 9 − Класс Bot

После данного этапа было установлено соединение с Telegram ботом, при помощи методов , которые среда попросила реализовать от материнского класса:

* public String getBotUsername()
* public String getBotToken()

Которые показаны на рисунках 10 и 11

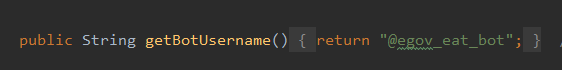


Рисунок 10 − Методы класса

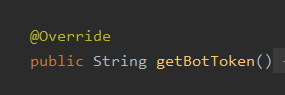


Рисунок 11 − Методы класса

После подключения к Telegram боту в данном проекте был добавлен так называемый «слушатель», через метод onUpdateReceived, изображенный на рисунке 12, который ждет отправки сообщения от пользователя, таких команд как /star или /restar

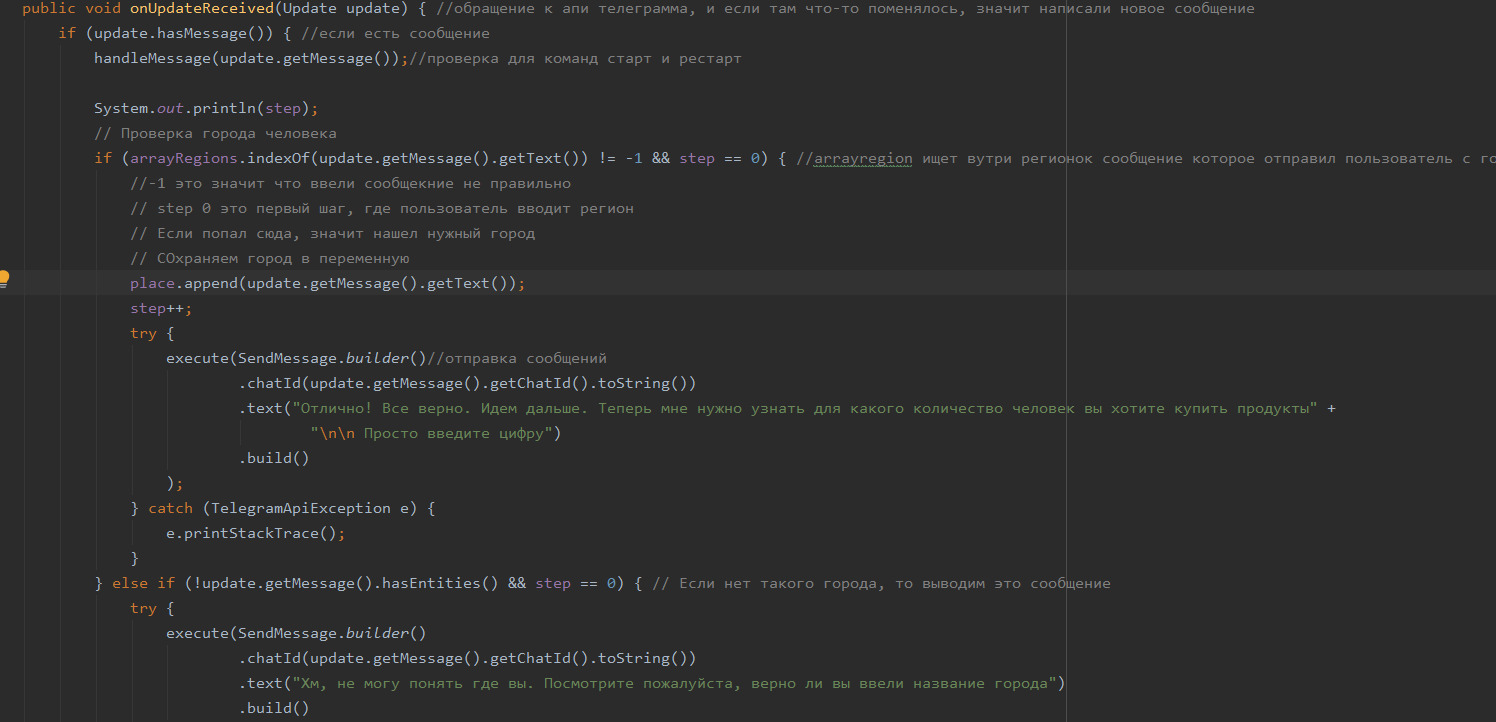


Рисунок 12 − Метод onUpdateReceived

После этого была реализация проверки на наличие команд от пользователя при помощи метода обработки сообщений handleMessage.

Данный метод показан на рисунке 13, также в данном методе присутствует условие с проверкой данного сообщения на текст или команду

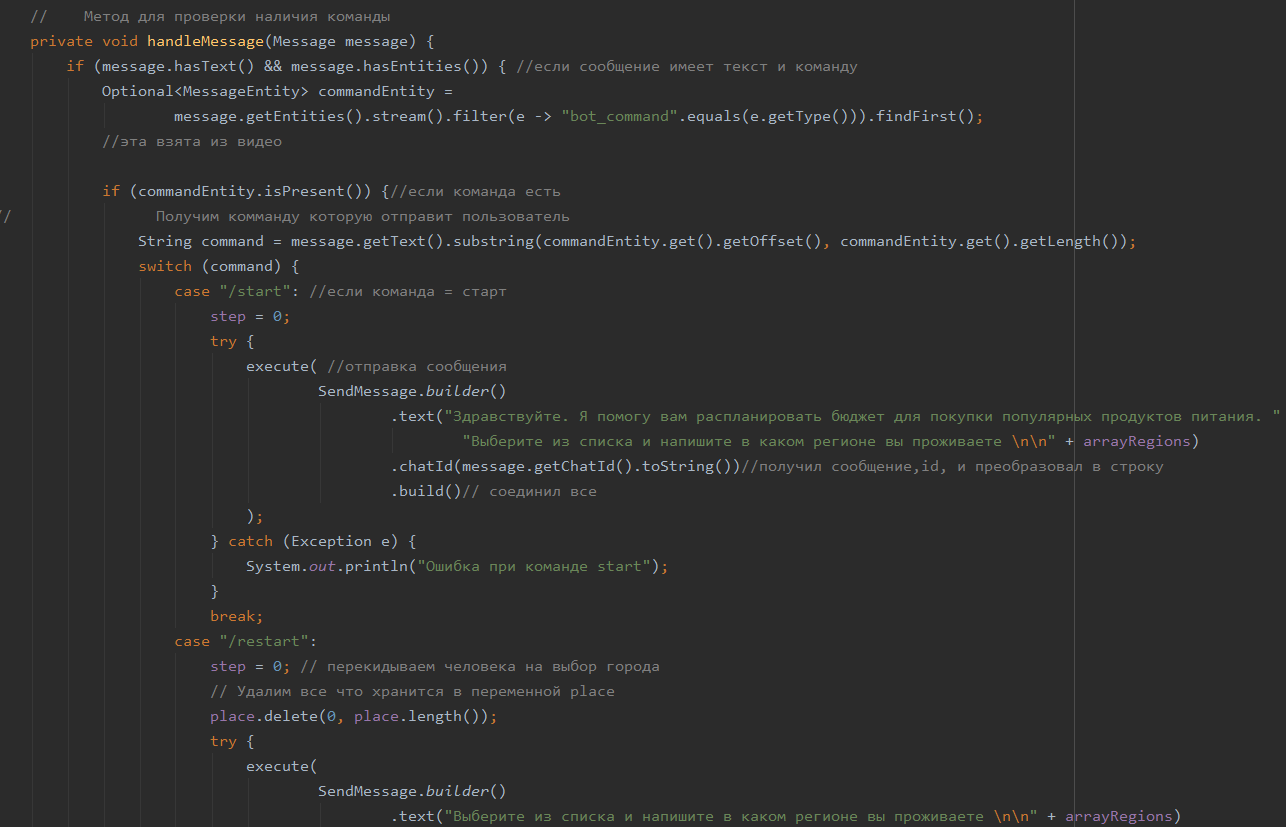


Рисунок 13 − Метод handleMessage

Кроме базовой команды /start, которая является стандартной в каждом боте Telegram, в данном проекте были добавлена команда /restart, через главный бот Telegram – BotFather, о котором было написано выше

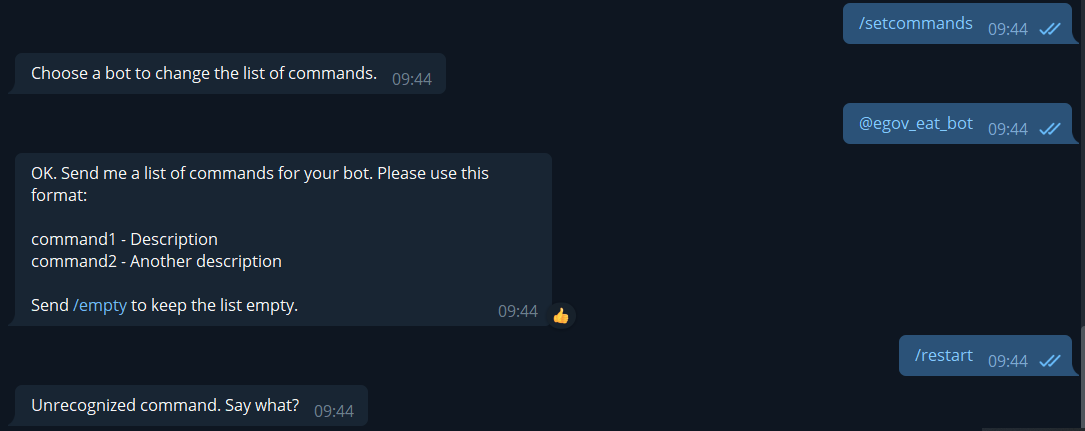


Рисунок 13 − Команда Restart

Команда /restart нужна для того чтобы пользователь мог повторно отправить запрос

Кода пользователь вводит сообщение идет проверка на содержание команды /start или /restart через оператор switch/

Switch это специальный оператор, который последовательно сравнивает введенные команды в чат Telegram, которые находятся после каждого ключевого слова case в коде. Реализация этого оператора изображена на рисунке 14



Рисунок 14 − Реализация оператора switch

Если была введена команда /start пользователю выводится приветственное сообщение и массив городов, которые получены из egov API и инструкция по использованию бота, которая была реализована с помощью метода SendMessage.builder().

В данном методе мы указываем текст сообщения и id чата с пользователем с помощью строк: text("Здравствуйте. Я помогу вам распланировать бюджет для покупки популярных продуктов питания. " +  
 "Выберите из списка и напишите в каком регионе вы проживаете \n\n" + arrayRegions) и chatId(message.getChatId().toString())

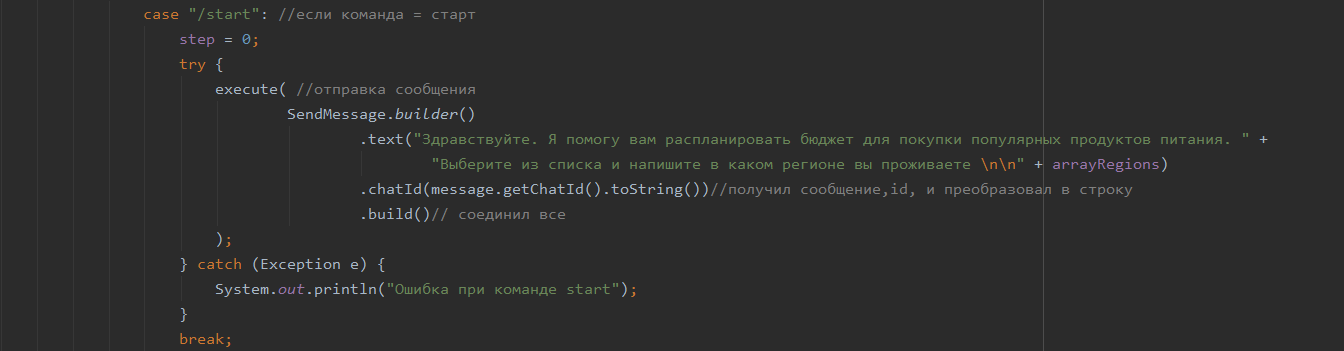


Рисунок 15 − Метод SendMessage.builder

Для того, чтобы подключиться к egov API был реализован класс Req, изображённый на рисунке 16



Рисунок 16 − Класс Req

В данном классе расположен главный метод, который получает данные.

Для этого создаётся защищенное соединение и открывается соединение, изображённое на рисунке 17

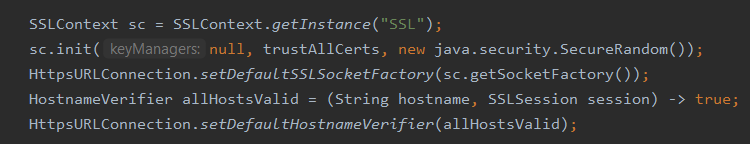


Рисунок 17 − Защищенное соединение

Далее формируется строка, с записью ответа от сервера, изображенная на рисунке 18 в которую записывается ответ, полученный от сервера egov

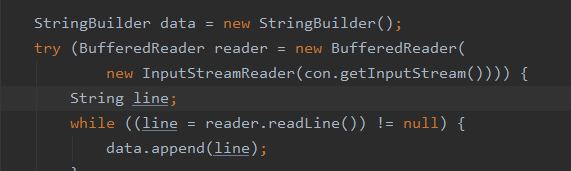


Рисунок 18 − Запись ответа

Для того чтобы вывести пользователю какие города доступны, реализован второй класс getRegions

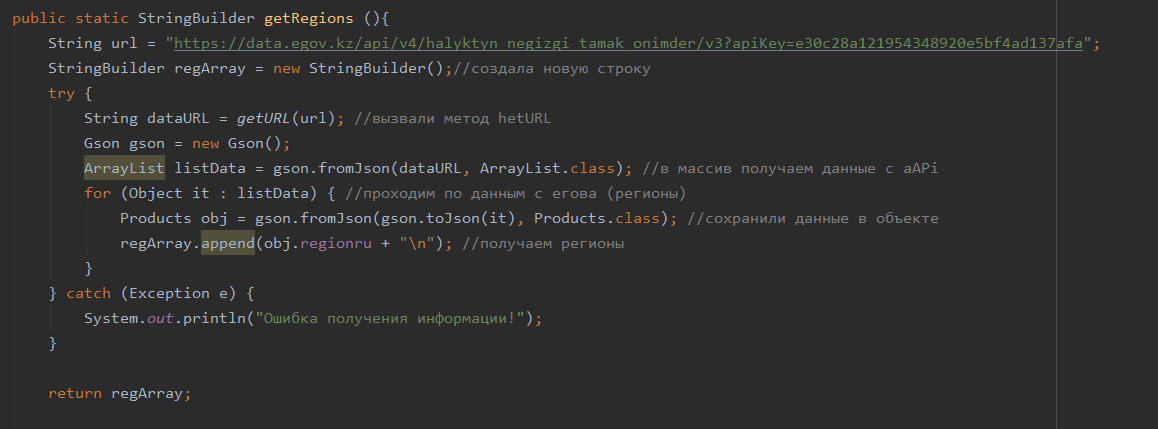


Рисунок 19 − Класс getRegions

Для более удобной разработки полученные данные были преобразованы в формат gson с помощью библиотеки gson, изображенной на рисунке 20



Рисунок 20 − Библиотека Gson

Gson – это библиотека на языке Java c открытым исходным кодом для перевода какой либо необходимой структуры данных в последовательность битов, и обратной данной операции.

Данная библиотека требует, чтобы для данных был создан специальный класс, который должен содержать такие же переменные, как и на сервере egov, как представлены данные на сервере egov, представлено на рисунке 21.

Для того чтобы данные на данном сайте приняли вид как на рисунке 21 необходимо было использовать специальное расширение браузера под название GsonViewer



Рисунок 21 − Egov

Поэтому был реализован класс products, изображенный на рисунке 22, в котором необходимо было прописать данные без ошибочно и точно, как в формате gson

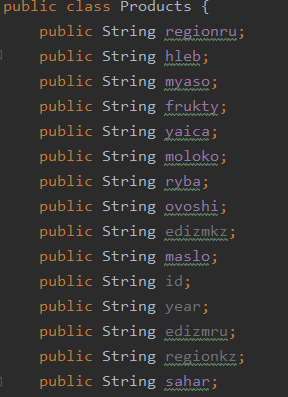


Рисунок 22 − Класс Products

Здесь все переменные имеют такое же наименование, как и в сервере egov.

Для сохранения всех данных был создан массив arrayList, в котором хранился ответ от сервера egov, массив представлен на рисунке 23

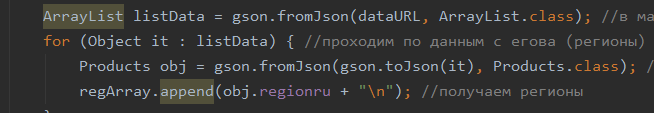


Рисунок 23 − Массив

Далее необходимо извлечь данные для каждого региона с помощью Products obj = gson.fromJson(gson.toJson(it), Products.class), который находится в методе getRegions, представленный на рисунке 24

Также в этом методе прописывается ссылка на Egov Api, в соответствии с полученным ключом Api при регистрации

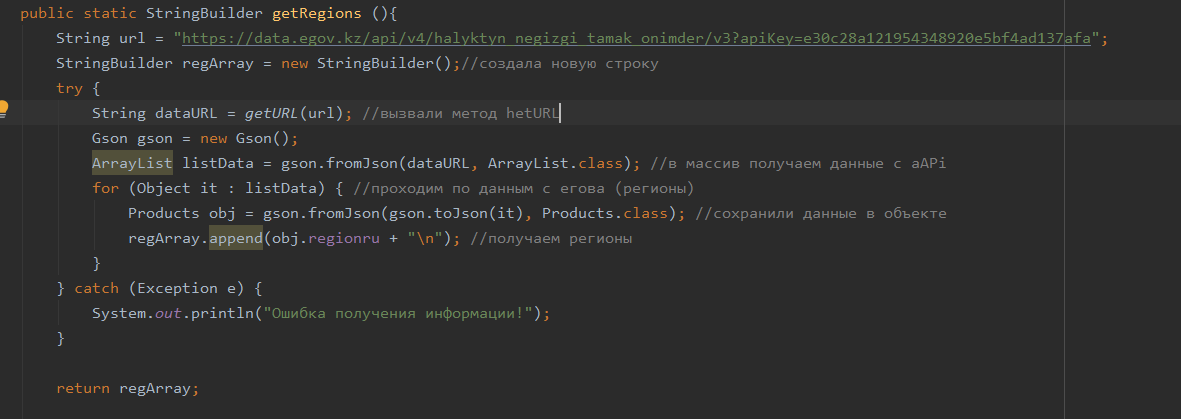


Рисунок 24 − Метод getRegions

После реализации все данные с сервера хранились в объектах obj.

Так как все это делалось для получения списков регионов в строку req.array определенную ранее, из каждого объекта obj было получено информация о регионах через obj.regionru



Рисунок 25 − Получение регионов

После работы цикла, изображенного на рисунке выше, было получено готовую строку со всеми городами, и данную строку мы отправляли пользователю в приветственном сообщении.

После этого необходимо было дождаться ответа пользователя чтобы он точно указал название города, для проверки было создано условие, изображенное на рисунке 26



Рисунок 26 − Условие

Если пользователь ввел не корректный ответ ему отправлялось сообщение, представленное на рисунке 27

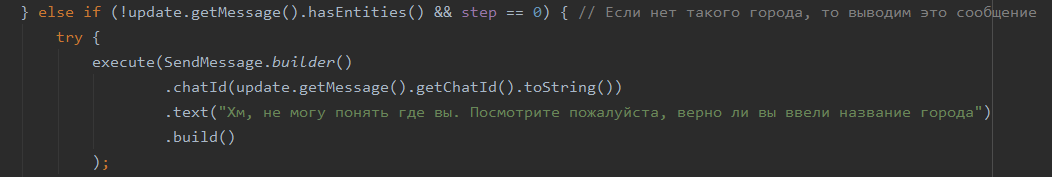


Рисунок 27 − Получение сообщения

В версии Telegram сообщение представлено в видео, показанном на рисунке 28, дальше бот ждет корректное ответное сообщение от пользователя, при введении некорректного ответа бот будет отправлять данное сообщение столько раз, сколько получит неверный ответ

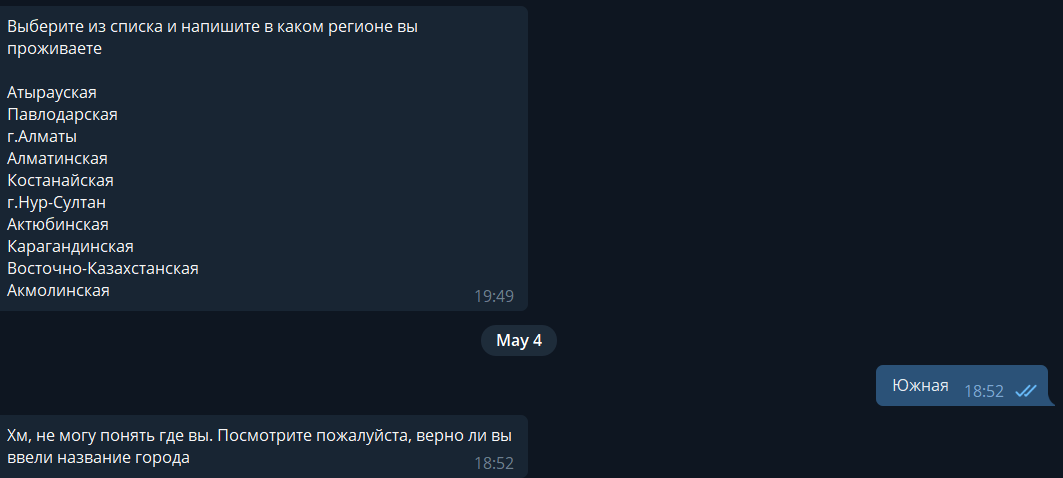


Рисунок 28 − Некорректное введение данных

Если данные введены корректно, переходим к следующему шагу. Следующий шаг — это вывод сообщения: «Отлично! Все верно. Теперь мне нужно узнать для какого количества человек вы хотите купить продукты», в котором бот узнает точное количество человек

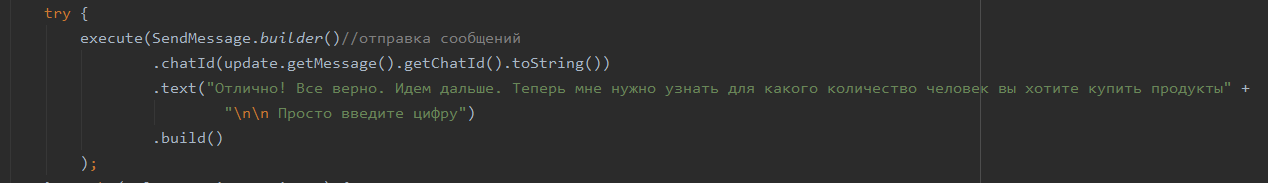


Рисунок 29 − Вывод сообщения

На этом этапе бот ждет чтобы пользователь ввел для какого количества человек необходимо купить продуктов.

Так же здесь реализована защита c помощью условия if, представленная на рисунке 30 , которое защищает бота от некорректного введения данных, например цифр меньше 0, или слов.

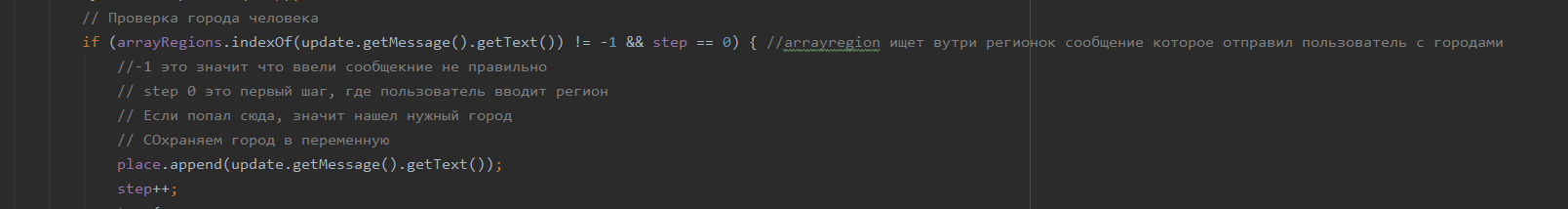


Рисунок 30 − Защита от некорректного введения

Если был введен текст, то ничего не происходит, если введено число равное или меньше 0, выводится сообщение, представленное на рисунке 31

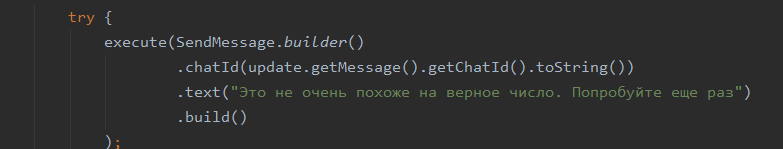


Рисунок 31 − Сообщение при некорректном введении данных

Если же данные корректны, идет следующий запрос на сервер. Для этого в классе req реализован метод getInfo.

Данный метод получает ответ от пользователя, регион и количество человек, он работает по такому же принципу что и getRegions, описанный выше, за исключением условия, представленного на рисунке 32

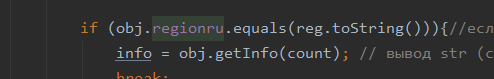


Рисунок 32 − Условие

Далее из всех регионов мы выбираем тот регион, который ввел пользователь и запрашиваем информацию про этот регион.

Полную информацию про этот регион подготавливает класс products с помощью метода getInfo, представленного на рисунке 33

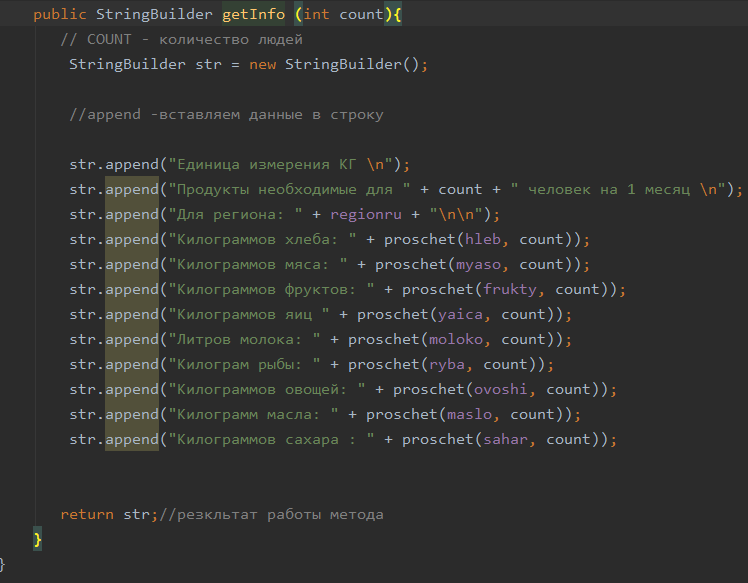


Рисунок 33 − Метод getInfo

Так как внутри объекта хранятся данные от сервера, то нам эти данные необходимо вывести для пользователя в понятном для него формате, что было сделано с помощью StringBuilder и пояснению к каждому из полей. Пояснение к каждому из полей берется с сайта открытых данных egov, в данном случае это продукты, необходимо ввести данные также как они представлены на сайте.

После данного этапа, необходимо вернуться в класс bot

Здесь отправляется пользователю сообщение об обработке данных, представленный на рисунке 34



Рисунок 34 − Сообщение об обработке данных

Когда был получен ответ от сервера пользователю сразу отправляется сообщение, которое было получено от метода getInfo, представленный на рисунке 35

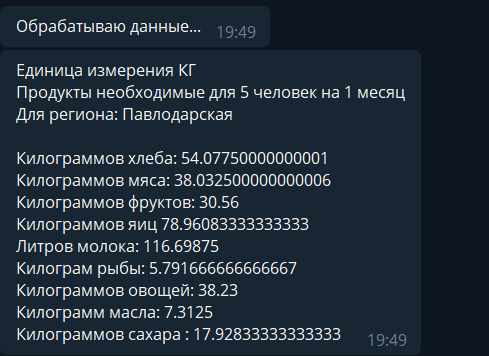


Рисунок 35 − Вывод данных

В случае если пользователь хочет воспользоваться ботом повторно, ему предлагается ввести команду /restart, бот также выводит это сообщение, как пользователь видит сообщение представлено на рисунке 36

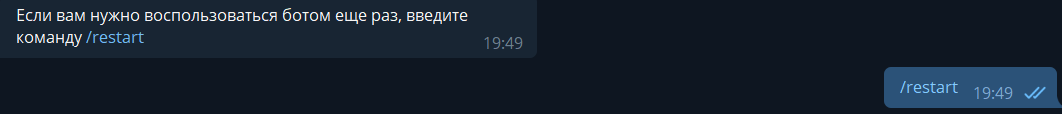


Рисунок 36 − Команда restart

Команда /restart реализуется с помощью метода execute, представленный на рисунке 37

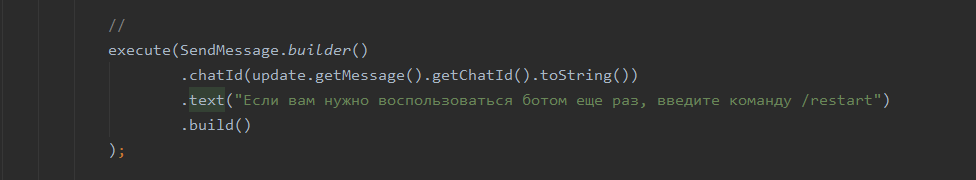


Рисунок 37 − Метод execute

Когда пользователь нажимает команду /restart, ему заново отправляется список со всеми городами и работа бота повторяется заново, код данной команды представлен на рисунке 38

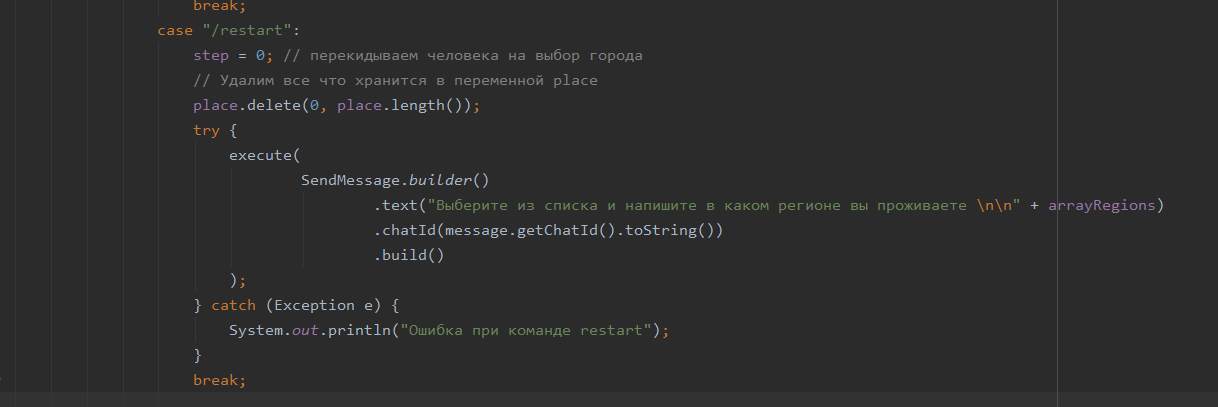


Рисунок 38 − Обновление работы бота

**Заключение**

В настоящее время популярность мессенджером растет. Также стоит отметить рост популярности Telegram, как приводилось выше мессенджер насчитывает более 100 миллионов пользователей, и соответственно Telegram ботов, которые работают на данной платформе.

Благодаря социальным сетям мы можем не только общаться, делиться фотографиями, но и автоматизировать процессы и передавать рутинную работу службам автоматизации, в нашем случае это Telegram боты.

В рамках данной работы были выполнены поставлены задачи. В начале работы были изучены материалы и ресурсы, которые помогли в построении и реализации Telegram бота, также были выявлены возможности применения, функции и технологии ботов в Telegram, и в конечном итоге спроектирован и разработан чат бот.

Тема данного дипломного проекта является наиболее острой и актуальной, ведь это не просто получение данных из открытых источников, которое предоставляет государство, а возможность сэкономить средства как для людей с разным уровнем достатка, так и для благотворительных фондов, которые являются основной целевой аудиторией данного TELEGRAM бота.

В настоящее время идет тенденция на благотворительность, сохранение природы и экологии, и несмотря на то, что данный проект разрабатывался в сфере IT, он может помочь решить такие проблемы как:

- проблема экономической доступности продуктов питания

- расчёт количества продуктов питания, необходимых для покупки в благотворительных фондах

Открытые данные мы можем использовать не только для аналитической деятельности, но и для практического использования. Также сейчас стоит остро проблема автоматизации процессов в различных социально экономических областях. В результате автоматизации процессов освобождается человеческий ресурс, который в свою очередь может быть реализован в других областях, тем самым это повышает эффективность работы.

Практическая значимость данного проекта, заключается в том, что данный TELEGRAM бот можно использовать как для личных целей, так и для социально значимых проектов

**Список использованных источников**

1 Н. А. Вязовик. Программирование на Java Курс лекций, Интернет университет информационных технологий, 2003. − 592 с.

2 О. В. Герман. − Санкт-Петербург.: Программирование на Java и C#, 2012. − 512 c.

3 Берд, Барри Java для чайников / Барри Берд. − М. : Диалектика / Вильямс, 2013. − 521c.

4 Эккель, Брюс Философия Java / Брюс Эккель. − М. : Питер, 2016. − 809c.

5 Создание Telegram бота на Java: от идеи до деплоя [https://javarush.ru/groups/posts/504-sozdanie-telegram-bota-na-java-ot-idei-do-deploja]

**Приложение A**

import org.telegram.telegrambots.bots.DefaultBotOptions;  
import org.telegram.telegrambots.bots.TelegramLongPollingBot;  
import org.telegram.telegrambots.meta.TelegramBotsApi;  
import org.telegram.telegrambots.meta.api.methods.send.SendMessage;  
import org.telegram.telegrambots.meta.api.objects.Message;  
import org.telegram.telegrambots.meta.api.objects.MessageEntity;  
import org.telegram.telegrambots.meta.api.objects.Update;  
import org.telegram.telegrambots.meta.exceptions.TelegramApiException;  
import org.telegram.telegrambots.updatesreceivers.DefaultBotSession;  
  
import java.util.ArrayList;  
import java.util.Optional;  
  
  
public class Bot extends TelegramLongPollingBot {  
 // Массив который хранит в себе все города  
 private StringBuilder arrayRegions = Req.getRegions(); //приватная переменная классА, содержит в себе все города  
 private StringBuilder place = new StringBuilder();  
 private int countPeople = 0;  
 private int step = 0;  
 private boolean firstStep = false;  
  
 public String getBotUsername() {  
 return "@egov\_eat\_bot";  
 } //передаем имя нашего бота  
  
 // бот становиться слушателем и ждет пока отправят сообщения  
 public void onUpdateReceived(Update update) { //обращение к апи телеграмма, и если там что-то поменялось, значит написали новое сообщение  
 if (update.hasMessage()) { //если есть сообщение  
 handleMessage(update.getMessage());//проверка для команд старт и рестарт  
  
 System.out.println(step);  
 // Проверка города человека  
 if (arrayRegions.indexOf(update.getMessage().getText()) != -1 && step == 0) { //arrayregion ищет вутри регионок сообщение которое отправил пользователь с городами  
 //-1 это значит что ввели сообщекние не правильно  
 // step 0 это первый шаг, где пользователь вводит регион  
 // Если попал сюда, значит нашел нужный город  
 // СОхраняем город в переменную  
 place.append(update.getMessage().getText());  
 step++;  
 try {  
 execute(SendMessage.builder()//отправка сообщений  
 .chatId(update.getMessage().getChatId().toString())  
 .text("Отлично! Все верно. Идем дальше. Теперь мне нужно узнать для какого количество человек вы хотите купить продукты" +  
 "\n\n Просто введите цифру")  
 .build()  
 );  
 } catch (TelegramApiException e) {  
 e.printStackTrace();  
 }  
 } else if (!update.getMessage().hasEntities() && step == 0) { // Если нет такого города, то выводим это сообщение  
 try {  
 execute(SendMessage.builder()  
 .chatId(update.getMessage().getChatId().toString())  
 .text("Хм, не могу понять где вы. Посмотрите пожалуйста, верно ли вы ввели название города")  
 .build()  
 );  
 } catch (TelegramApiException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // Проверка количества человек  
 if (Integer.parseInt(update.getMessage().getText()) > 0 && step == 1) {//проверка на число  
 countPeople = Integer.parseInt(update.getMessage().getText());//сохраняем кол-во человек в переменную  
 step++;  
 try {  
 execute(SendMessage.builder()  
 .chatId(update.getMessage().getChatId().toString())  
 .text("Обрабатываю данные...")  
 .build()  
 );  
  
// ЗАПРОС К ЕГОВ И ОТРАВКА ДАННЫХ ПОЛЬЗОВАТЕЛЮ  
 execute(SendMessage.builder()  
 .chatId(update.getMessage().getChatId().toString())  
 .text(Req.getInfo(place, countPeople).toString()) //передали текст с req (str) и бот сам вывел  
 .build()  
 );  
  
 //  
 execute(SendMessage.builder()  
 .chatId(update.getMessage().getChatId().toString())  
 .text("Если вам нужно воспользоваться ботом еще раз, введите команду /restart")  
 .build()  
 );  
  
  
 } catch (TelegramApiException e) {  
 e.printStackTrace();  
 }  
 } else if (step == 1) { // Если это не число, то выводим это сообщение  
 try {  
 execute(SendMessage.builder()  
 .chatId(update.getMessage().getChatId().toString())  
 .text("Это не очень похоже на верное число. Попробуйте еще раз")  
 .build()  
 );  
 } catch (TelegramApiException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
  
 // Метод для проверки наличия команды  
 private void handleMessage(Message message) {  
 if (message.hasText() && message.hasEntities()) { //если сообщение имеет текст и команду  
 Optional<MessageEntity> commandEntity =  
 message.getEntities().stream().filter(e -> "bot\_command".equals(e.getType())).findFirst();  
 //эта взята из видео  
  
 if (commandEntity.isPresent()) {//если команда есть  
// Получим комманду которую отправит пользователь  
 String command = message.getText().substring(commandEntity.get().getOffset(), commandEntity.get().getLength());  
 switch (command) {  
 case "/start": //если команда = старт  
 step = 0;  
 try {  
 execute( //отправка сообщения  
 SendMessage.builder()  
 .text("Здравствуйте. Я помогу вам распланировать бюджет для покупки популярных продуктов питания. " +  
 "Выберите из списка и напишите в каком регионе вы проживаете \n\n" + arrayRegions)  
 .chatId(message.getChatId().toString())//получил сообщение,id, и преобразовал в строку  
 .build()// соединил все  
 );  
 } catch (Exception e) {  
 System.out.println("Ошибка при команде start");  
 }  
 break;  
 case "/restart":  
 step = 0; // перекидываем человека на выбор города  
 // Удалим все что хранится в переменной place  
 place.delete(0, place.length());  
 try {  
 execute(  
 SendMessage.builder()  
 .text("Выберите из списка и напишите в каком регионе вы проживаете \n\n" + arrayRegions)  
 .chatId(message.getChatId().toString())  
 .build()  
 );  
 } catch (Exception e) {  
 System.out.println("Ошибка при команде restart");  
 }  
 break;  
  
 }  
 }  
 }  
 }  
  
 @Override  
 public String getBotToken() {  
 return "5263055954:AAGtic0xJyZImeHSz18Ne\_8WSM20ug9EL-o";  
 }  
  
  
 public static void main(String[] args) {  
 // создаем бота  
 Bot bot = new Bot();  
 try {  
 TelegramBotsApi telegramBotsApi = new TelegramBotsApi(DefaultBotSession.class);//создаем сессию (это я взяла с интернета)  
 telegramBotsApi.registerBot(bot);//зарегестрировала бота в тегерам апи  
 } catch (Exception e) {  
 System.out.println();  
 }  
  
 }  
  
  
}

public class Products {  
 public String regionru;  
 public String hleb;  
 public String myaso;  
 public String frukty;  
 public String yaica;  
 public String moloko;  
 public String ryba;  
 public String ovoshi;  
 public String edizmkz;  
 public String maslo;  
 public String id;  
 public String year;  
 public String edizmru;  
 public String regionkz;  
 public String sahar;  
  
 @Override  
 public String toString() {  
 return "Регион = " + regionru;  
 }  
  
 private String proschet (String a, int b){  
 return String.valueOf(Double.parseDouble(a) \* b / 12) + "\n";  
 }  
  
 public StringBuilder getInfo (int count){  
 // COUNT - количество людей  
 StringBuilder str = new StringBuilder();  
  
 //append -вставляем данные в строку  
  
 str.append("Единица измерения КГ \n");  
 str.append("Продукты необходимые для " + count + " человек на 1 месяц \n");  
 str.append("Для региона: " + regionru + "\n\n");  
 str.append("Килограммов хлеба: " + proschet(hleb, count));  
 str.append("Килограммов мяса: " + proschet(myaso, count));  
 str.append("Килограммов фруктов: " + proschet(frukty, count));  
 str.append("Килограммов яиц " + proschet(yaica, count));  
 str.append("Литров молока: " + proschet(moloko, count));  
 str.append("Килограм рыбы: " + proschet(ryba, count));  
 str.append("Килограммов овощей: " + proschet(ovoshi, count));  
 str.append("Килограмм масла: " + proschet(maslo, count));  
 str.append("Килограммов сахара : " + proschet(sahar, count));  
  
  
 return str;//резкльтат работы метода  
 }  
}

import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.net.URLConnection;  
import java.security.KeyManagementException;  
import java.security.NoSuchAlgorithmException;  
import java.security.cert.X509Certificate;  
import java.util.ArrayList;  
import javax.net.ssl.HostnameVerifier;  
import javax.net.ssl.HttpsURLConnection;  
import javax.net.ssl.SSLContext;  
import javax.net.ssl.SSLSession;  
import javax.net.ssl.TrustManager;  
import javax.net.ssl.X509TrustManager;  
import com.google.gson.Gson;  
  
public class Req {  
//главный метод класса, он получает данные  
 public static String getURL(String urlStr) throws MalformedURLException, IOException, NoSuchAlgorithmException, KeyManagementException {  
  
 URL url = new URL(urlStr);  
  
 TrustManager[] trustAllCerts = new TrustManager[]{new X509TrustManager() {  
 @Override  
 public java.security.cert.X509Certificate[] getAcceptedIssuers() { return null; }  
 @Override  
 public void checkClientTrusted(X509Certificate[] certs, String authType) { }  
 @Override  
 public void checkServerTrusted(X509Certificate[] certs, String authType) { }  
 }};  
  
 SSLContext sc = SSLContext.getInstance("SSL");  
 sc.init(null, trustAllCerts, new java.security.SecureRandom());  
 HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());  
 HostnameVerifier allHostsValid = (String hostname, SSLSession session) -> true;  
 HttpsURLConnection.setDefaultHostnameVerifier(allHostsValid);  
  
 URLConnection con = url.openConnection();  
  
 StringBuilder data = new StringBuilder();  
 try (BufferedReader reader = new BufferedReader(  
 new InputStreamReader(con.getInputStream()))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 data.append(line);  
 }  
 } catch (IOException e) {  
 }  
 return data.toString();  
 }  
  
  
 public static StringBuilder getRegions (){  
 String url = "https://data.egov.kz/api/v4/halyktyn\_negizgi\_tamak\_onimder/v3?apiKey=e30c28a121954348920e5bf4ad137afa";  
 StringBuilder regArray = new StringBuilder();//создала новую строку  
 try {  
 String dataURL = getURL(url); //вызвали метод hetURL  
 Gson gson = new Gson();  
 ArrayList listData = gson.fromJson(dataURL, ArrayList.class); //в массив получаем данные с аAPi  
 for (Object it : listData) { //проходим по данным с егова (регионы)  
 Products obj = gson.fromJson(gson.toJson(it), Products.class); //сохранили данные в объекте  
 regArray.append(obj.regionru + "\n"); //получаем регионы  
 }  
 } catch (Exception e) {  
 System.out.println("Ошибка получения информации!");  
 }  
  
 return regArray;  
  
 }  
  
// Метод для получения инфы от сервера с учетеом полученных данных от пользователя  
 public static StringBuilder getInfo (StringBuilder reg, int count){  
 String url = "https://data.egov.kz/api/v4/halyktyn\_negizgi\_tamak\_onimder/v3?apiKey=e30c28a121954348920e5bf4ad137afa";  
 StringBuilder info = new StringBuilder();  
 try {  
 String dataURL = getURL(url); //вызвал метод  
 Gson gson = new Gson();  
 ArrayList listData = gson.fromJson(dataURL, ArrayList.class); //в массив получаем данные с апи  
 for (Object it : listData) {//проходим по данным егова  
 Products obj = gson.fromJson(gson.toJson(it), Products.class);// сохранили данные в объекте,которые в классе product.java  
  
 if (obj.regionru.equals(reg.toString())){//если регион в массиве и регион в сообщении одинаковые, то переходим к рассчетам  
 info = obj.getInfo(count); // вывод str (сообзщение с кг мяса)  
 break;  
 }  
 }  
 } catch (Exception e) {  
 System.out.println("Ошибка получения информации!");  
 }  
  
 return info;  
  
 }  
}