|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |
| Институт информационных технологий |
| Кафедра инструментального и прикладного программного обеспечения |



|  |  |
| --- | --- |
|  | |
| **Проект “Погодный бот-сервис”** | |
| **по дисциплине** | |
| **«Шаблоны программных платформ языка Джава»** | |
| Выполнили студенты группы **ИКБО-36-22** | ***Ярош В.Э., Утенков Ю.Ю, Захаров А.А, Аскаров Т.Е*** |
| Принял ассистент кафедры ИиППО | ***Ермаков С.Р.*** |

|  |  |  |
| --- | --- | --- |
| Проект выполнили | « » 2024 г. |  |
| «Зачтено» | « » 2024 г. |  |

Москва 2024

**Введение**

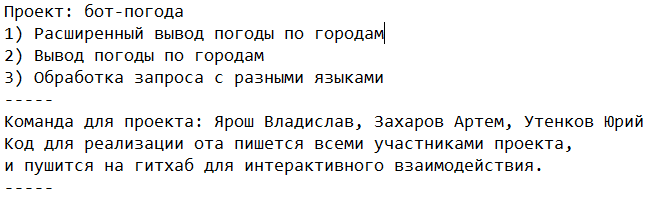
Целью данного отчета является описание разработки погодного бот-сервиса на Java с использованием фреймворка Spring Boot. В наше время использование ботов для получения информации о погоде становится все более популярным и удобным способом получения актуальных метеорологических данных. Этот отчет представляет собой обзор процесса проектирования и реализации такого бота, а также включает описание основных технологий и решений, использованных в ходе разработки.

В рамках работы были рассмотрены следующие аспекты:

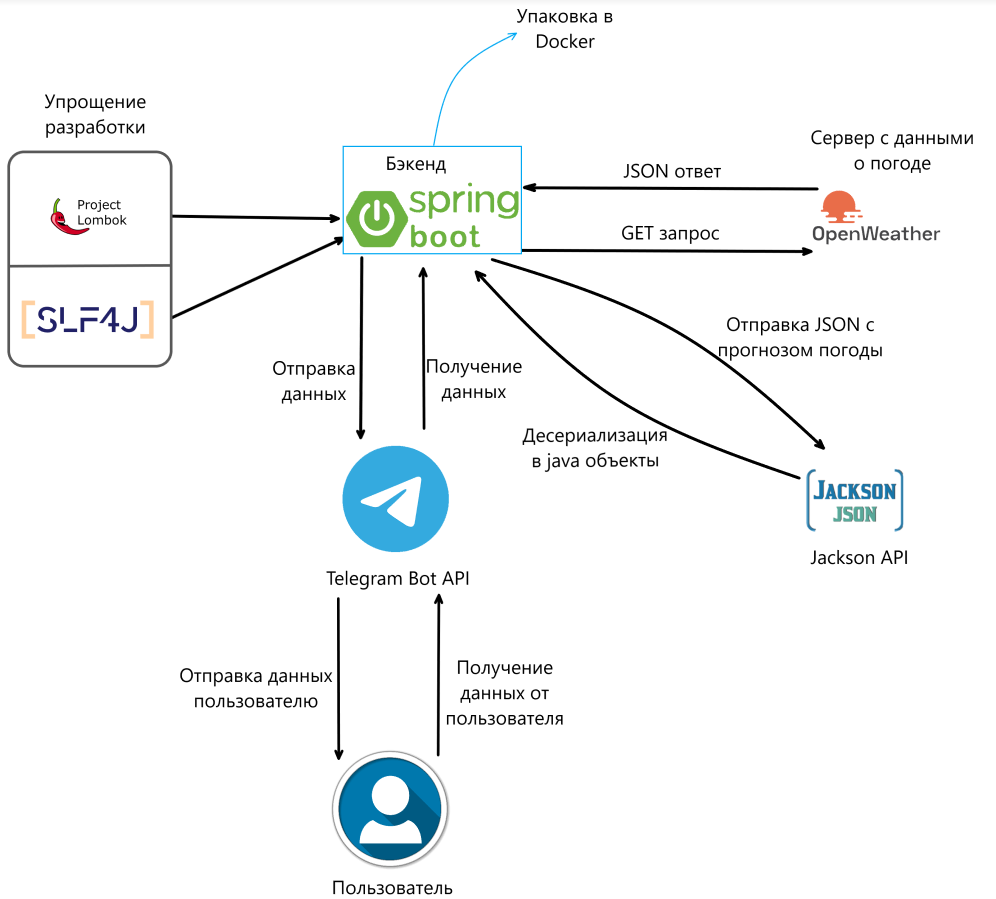
1. Техническое задание и функциональные требования: Определены основные требования к боту, включая способы взаимодействия с пользователем, формат предоставляемой информации о погоде и технологические ограничения.
2. Проектирование архитектуры: Разработана архитектура приложения, включая определение модулей, их взаимосвязей и структуру базы данных для хранения данных о запросах и результатов.
3. Интеграция с внешними сервисами: Реализовано взаимодействие с внешними API для получения данных о погоде, а также привязка сервиса к мессенджерам или другим каналам связи для обработки запросов пользователей.
4. Реализация функционала: Описан процесс создания основного функционала бота, включая логику обработки запросов, взаимодействия с базой данных и передачи информации пользователю.
5. Дальнейшее развитие: Описаны перспективы по дальнейшему улучшению и развитию сервиса.

Целью данного отчета является детальное описание каждого из указанных аспектов процесса разработки погодного бот-сервиса на Java Spring Boot с целью создания эффективного и удобного инструмента для получения информации о погоде.

**Техническое задание и функциональные требования**



**Проектирование архитектуры**



Используемые технологии:

1. Telegram Bot API: Для взаимодействия с Telegram платформой и создания бота я бы использовал Telegram Bot API. Это позволяет создавать ботов и обрабатывать сообщения от пользователей.
2. Java Spring Boot: Для реализации бота как веб-приложения на языке Java я бы выбрал Spring Boot. Spring Boot обеспечит простоту настройки и развертывания приложения, а также предоставит мощные инструменты для разработки REST.
3. OpenWeatherMap API: Для получения информации о погоде по запросу пользователя я бы воспользовался OpenWeatherMap API или аналогичным сервисом. Это API предоставляет доступ к актуальным метеорологическим данным, которые могут быть интегрированы в бота.
4. HTTP клиент (например, Apache HttpClient): Для отправки HTTP запросов к OpenWeatherMap API я бы использовал HTTP клиент на Java, такой как Apache HttpClient. Эти инструменты облегчают взаимодействие с внешними веб-сервисами.
5. Dependency Injection Framework (например, Spring DI): Для управления зависимостями и инверсии управления в приложении я бы использовал Dependency Injection фреймворк, например, встроенный в Spring Framework.
6. JUnit или другие фреймворки для тестирования: Для тестирования функционала бота я бы написал модульные тесты с использованием JUnit или других фреймворков для тестирования Java приложений.
7. Логирование (например, SLF4J) [в процессе разработки]: Для логирования работы бота я бы воспользовался библиотеками логирования, такими как SLF4J с реализацией Logback, чтобы обеспечить контроль и отслеживание работы приложения.

**Интеграция с внешними сервисами**

1. Регистрация и получение API ключа: Необходимо зарегистрироваться на OpenWeatherMap, чтобы получить уникальный API ключ, который будет использоваться для доступа к их сервисам.
2. Настройка HTTP клиента: Используя HTTP клиент в Java (Apache HttpClient), настраиваем запросы к API OpenWeatherMap с использованием полученного API ключа.
3. Валидация параметров URL запросов: Проверка отправленной пользователем команды, для получения точного и корректного ответа от бот-сервиса.
4. Формирование запросов: Создаем запросы к OpenWeatherMap API, указывая необходимые параметры, такие как город или координаты, чтобы получить информацию о погоде.
5. Обработка ответов: Обрабатываем ответы от OpenWeatherMap API, извлекаем необходимые данные о погоде (температура, погодные условия и т. д.), которые будут дальше использованы для формирования ответа пользователя.
6. Интеграция с логикой бота: Полученные данные о погоде интегрируются в логику бота, чтобы бот мог формировать информативные ответы на запросы пользователей о текущей или прогнозной погоде.

**Реализация функционала**

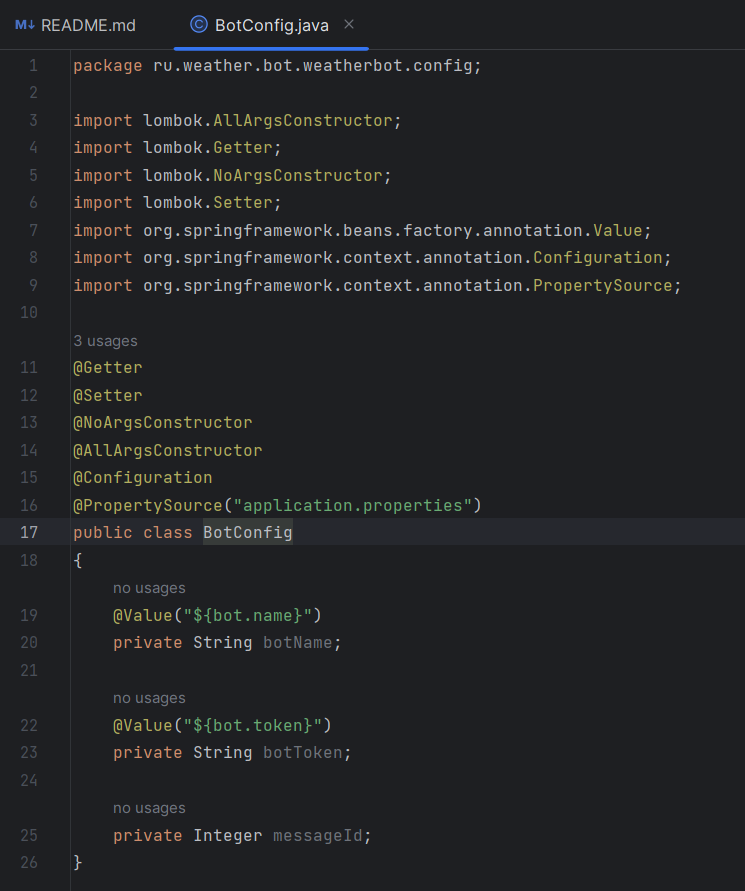
****

Рисунок 1 - Наполнение класса BotConfig

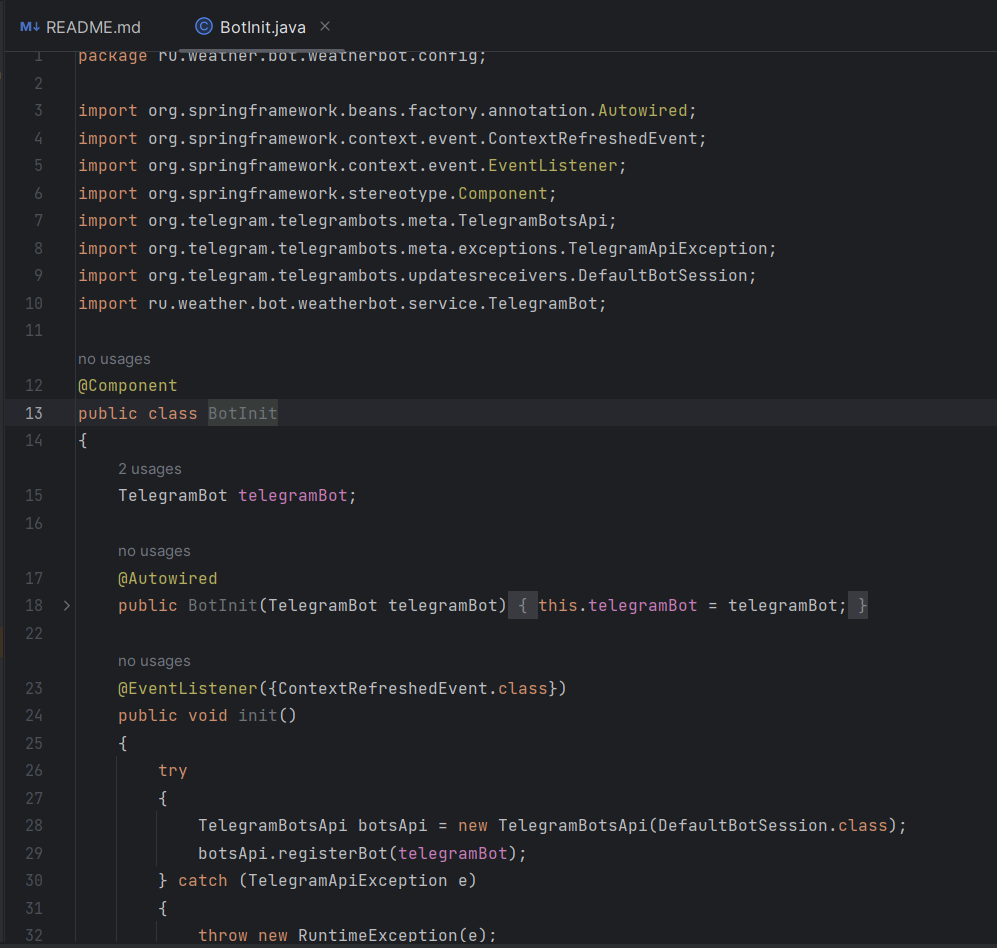


Рисунок 2 - Наполнение класса BotInit

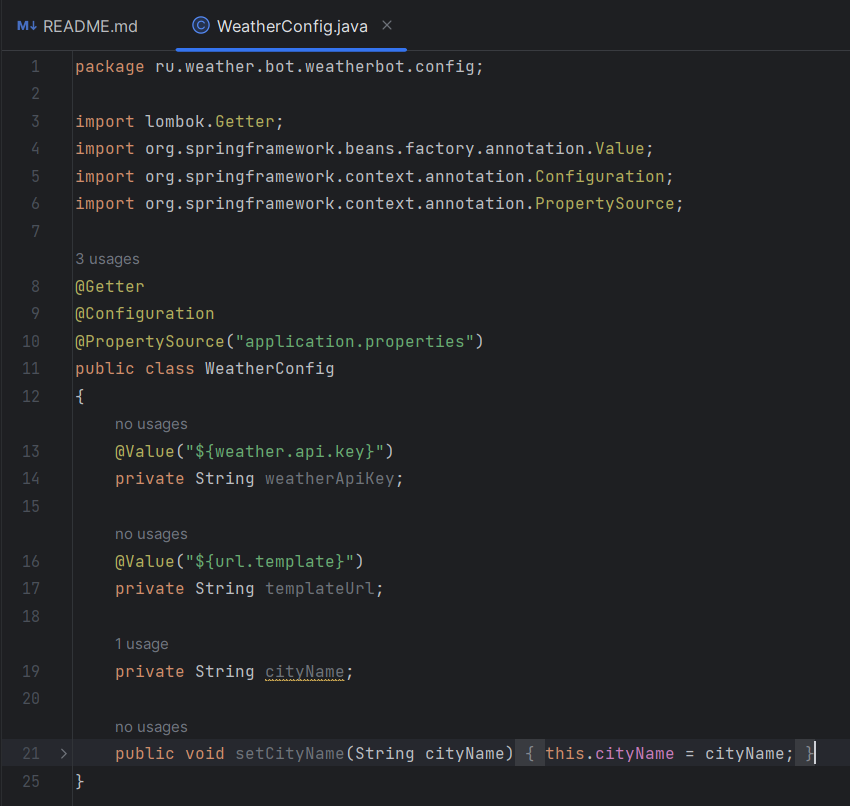


Рисунок 3 - Наполнение класса WeatherConfig



Рисунок 4 - Наполнение класса BotCommand



Рисунок 5 - Наполнение класса BotLanguage



Рисунок 6 - Наполнение класса WeatherData

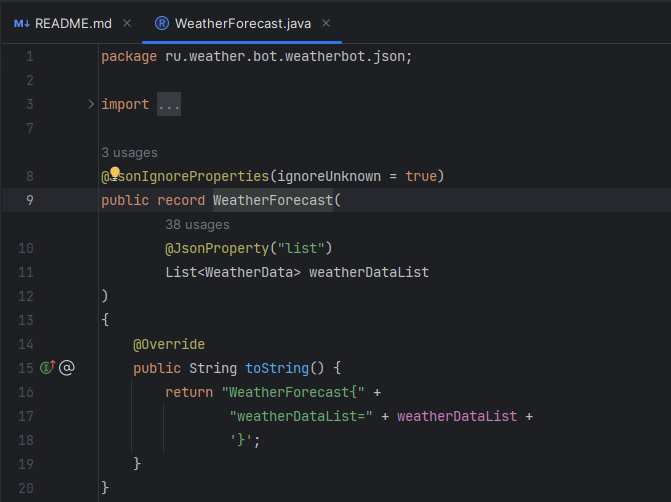


Рисунок 7 - Наполнение класса WeatherForecast

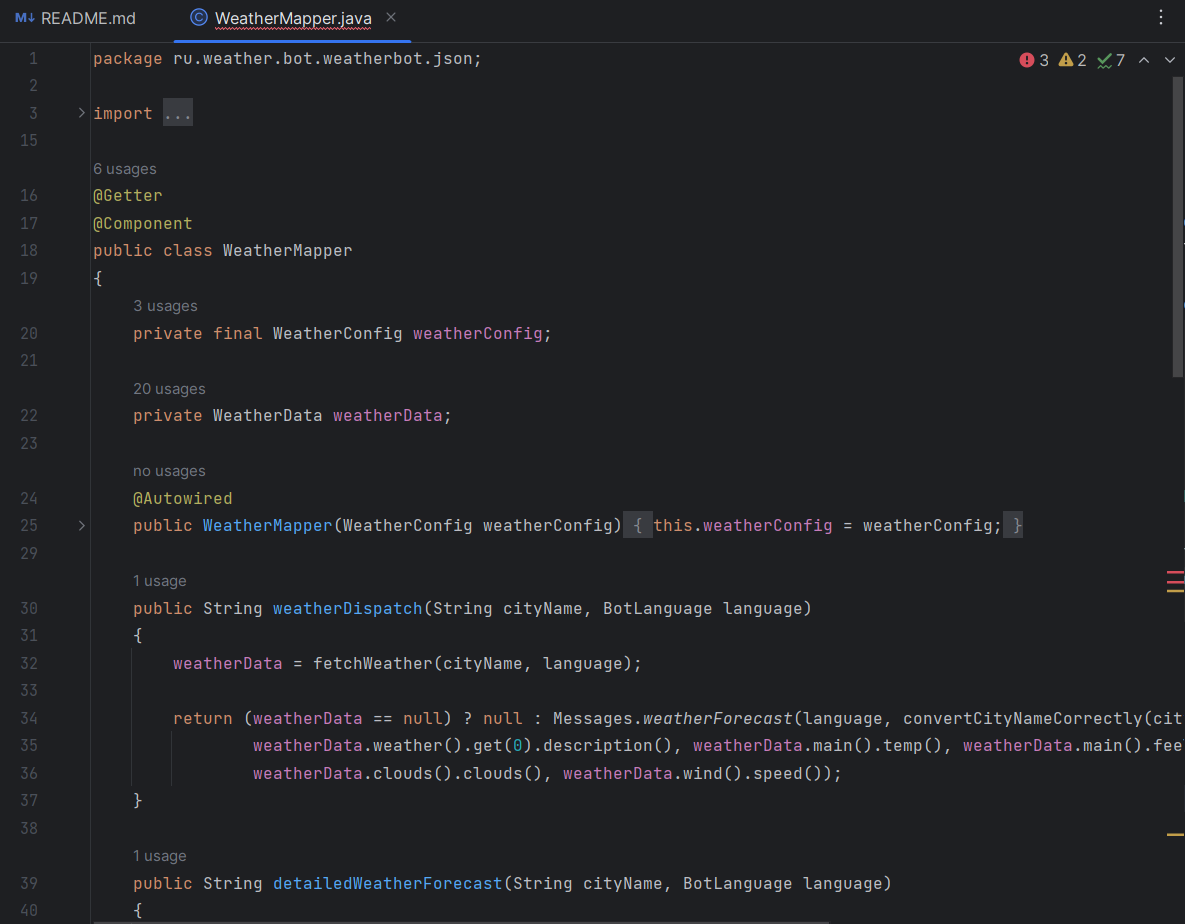


Рисунок 8 - Наполнение класса WeatherMapper

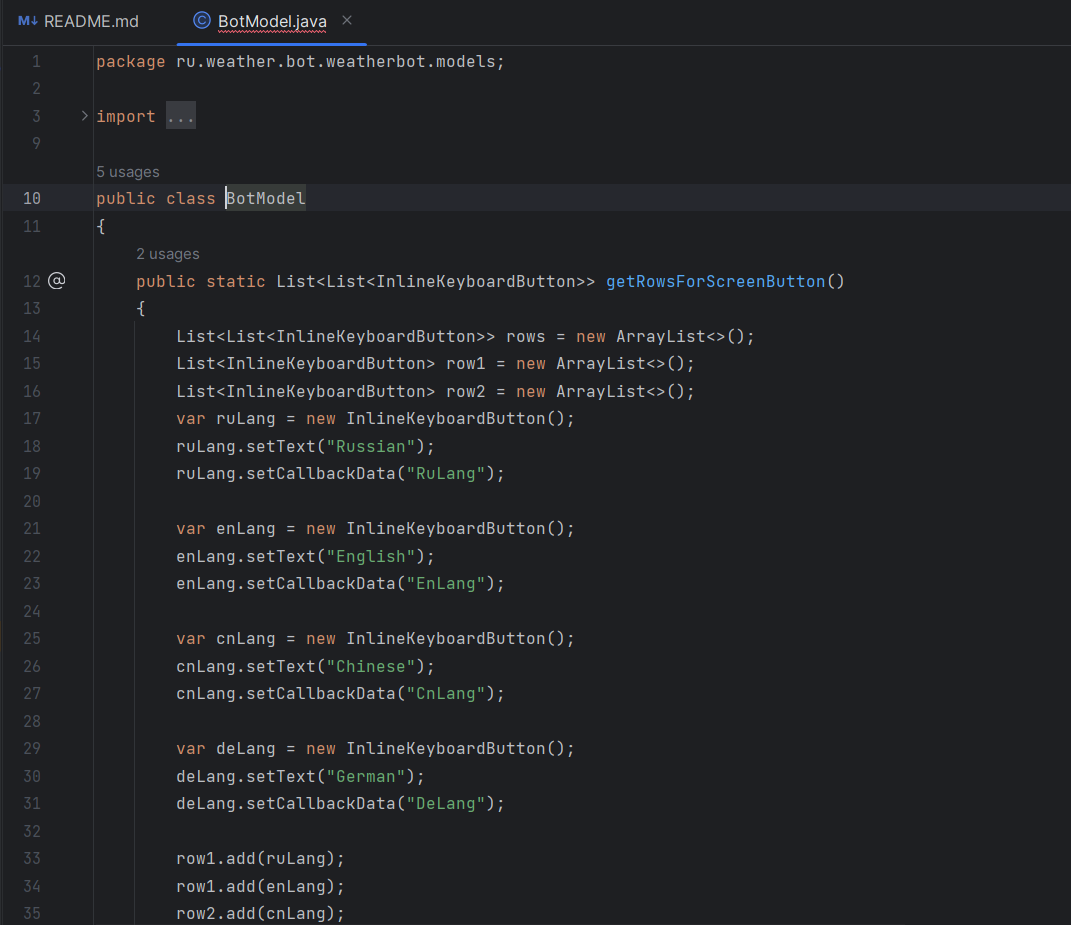


Рисунок 9 - Наполнение класса BotModel

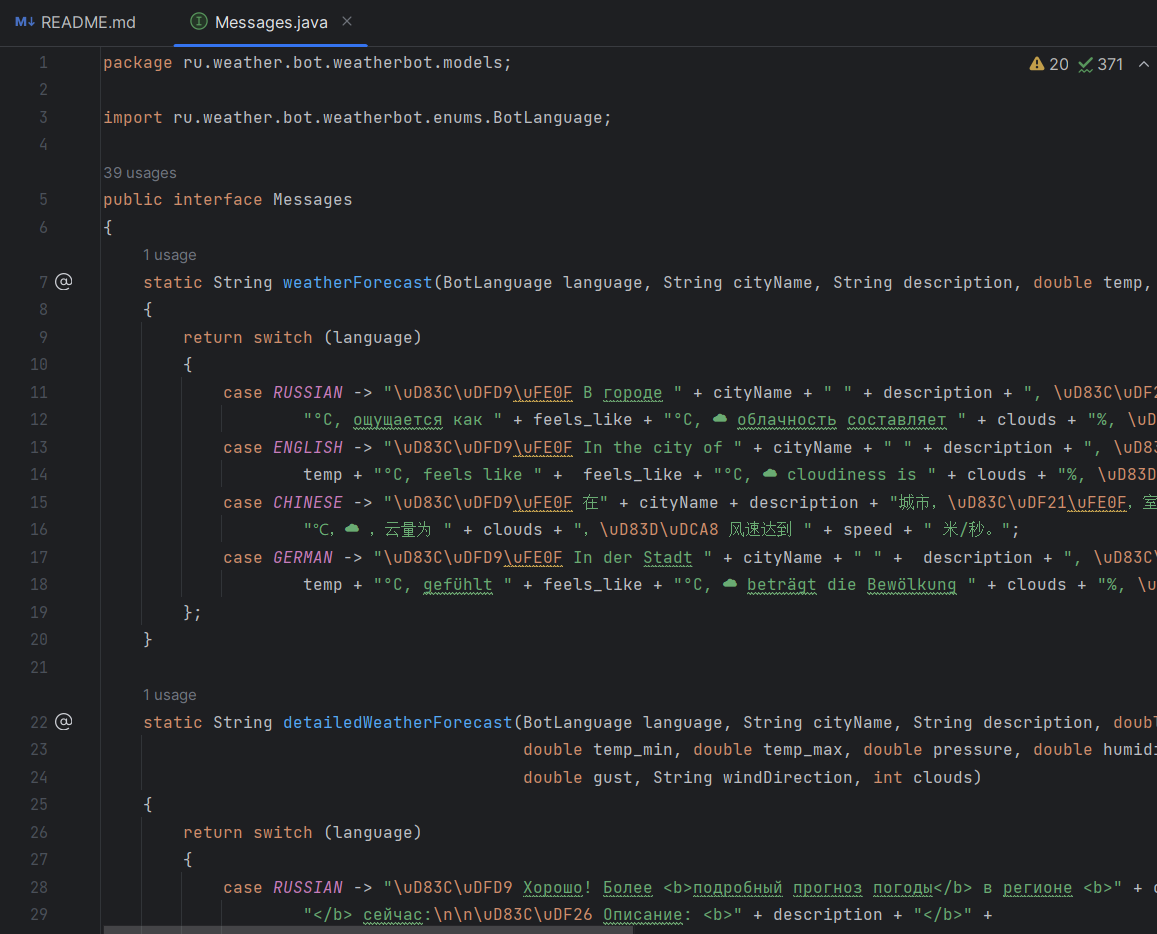


Рисунок 10 - Наполнение интерфейса Messages

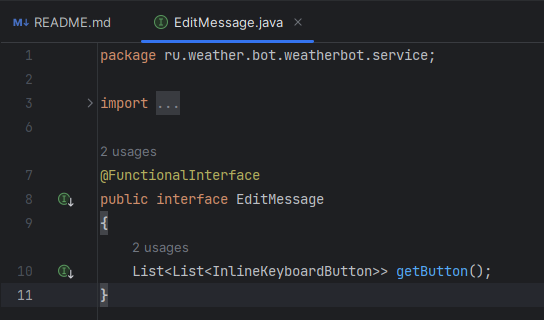


Рисунок 11 - Наполнение интерфейса EditMessage

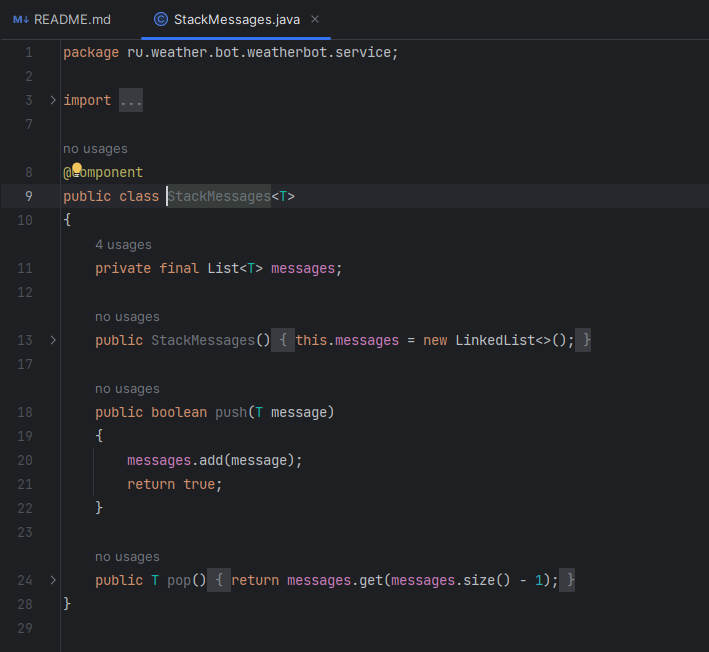


Рисунок 12 - Наполнение класса StackMessage

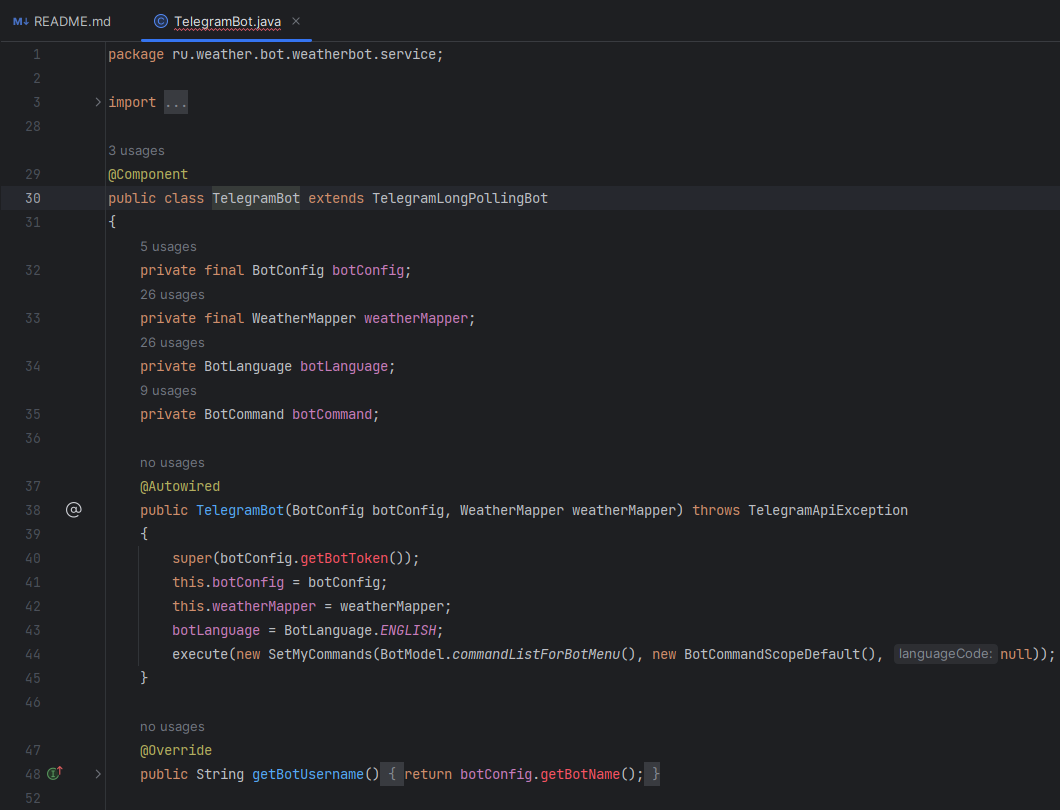


Рисунок 13 - Наполнение класса TelegramBot



Рисунок 14 - Точка запуска приложения

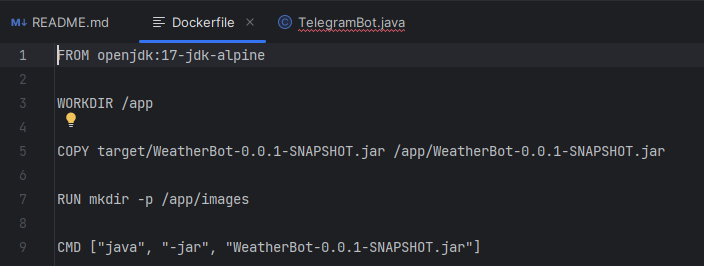


Рисунок 15 - Dockerfile для запуска приложения в контейнере

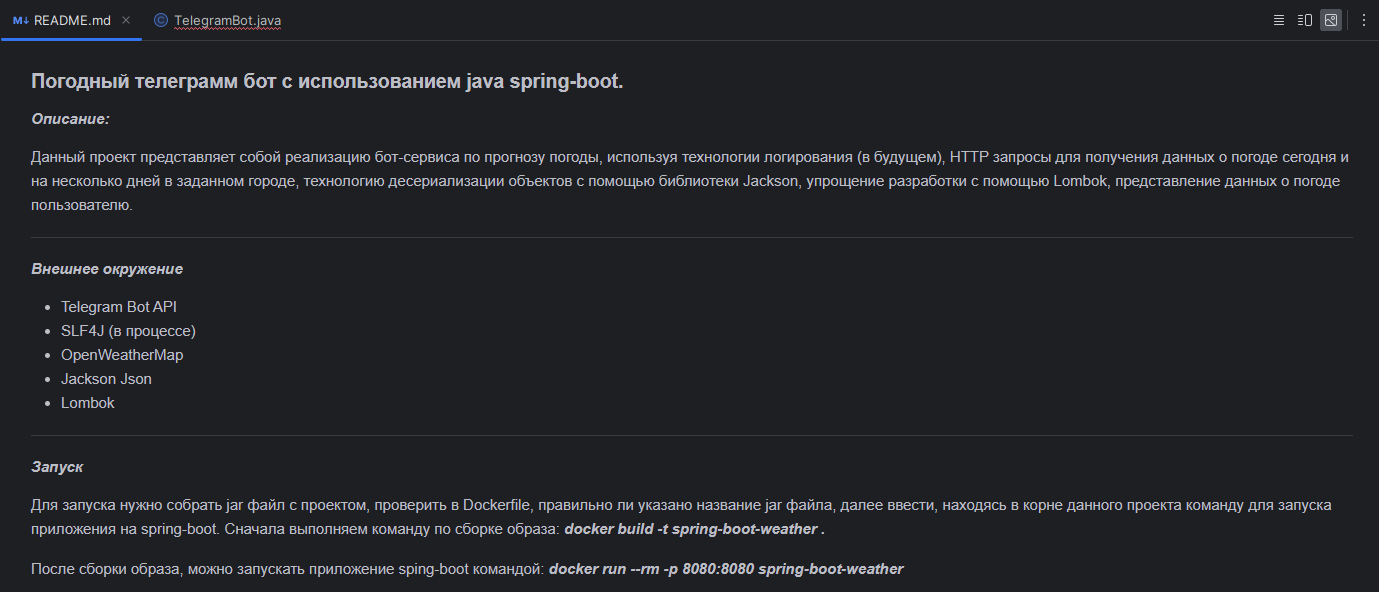


Рисунок 16 – Readme file

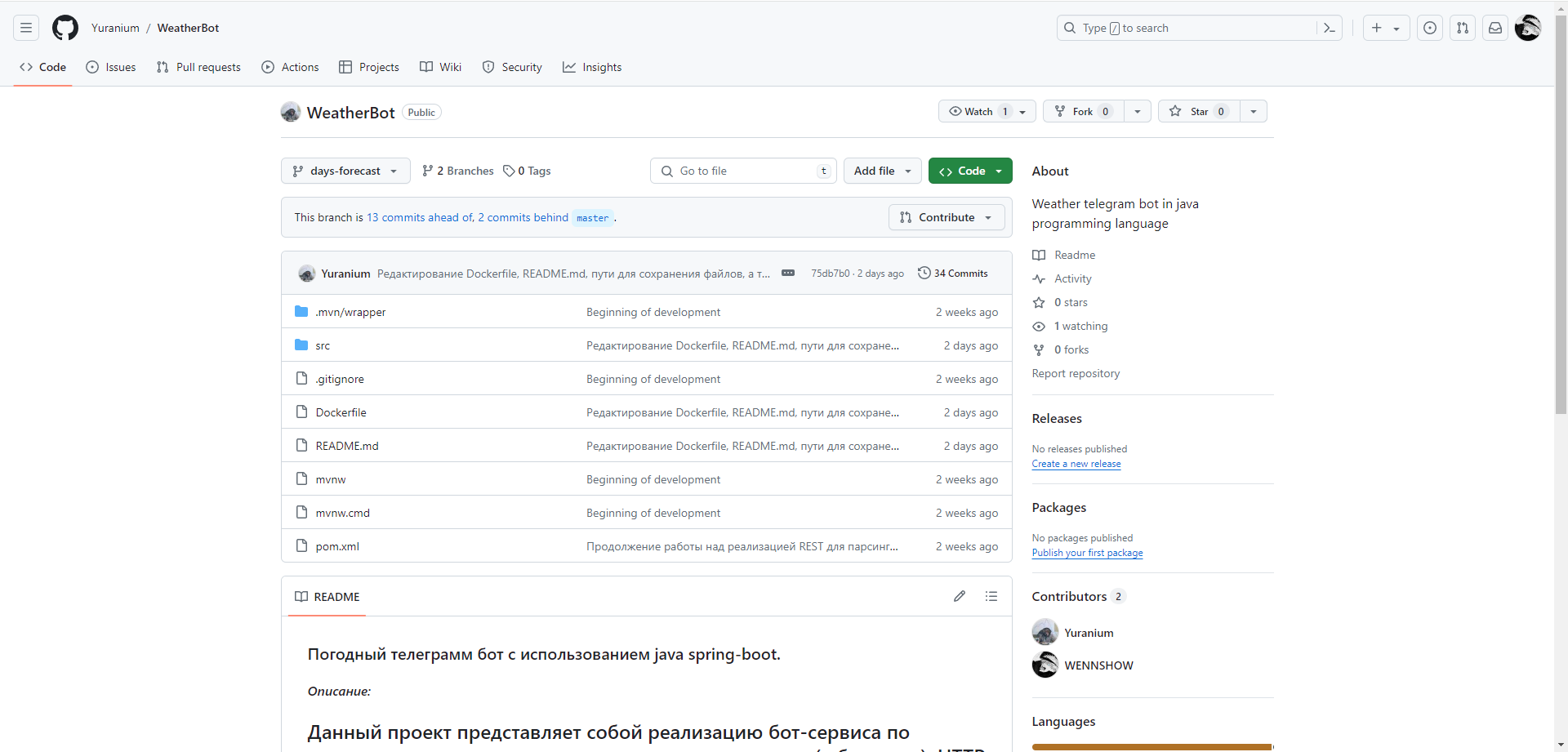


Рисунок 17 - Приложение в github

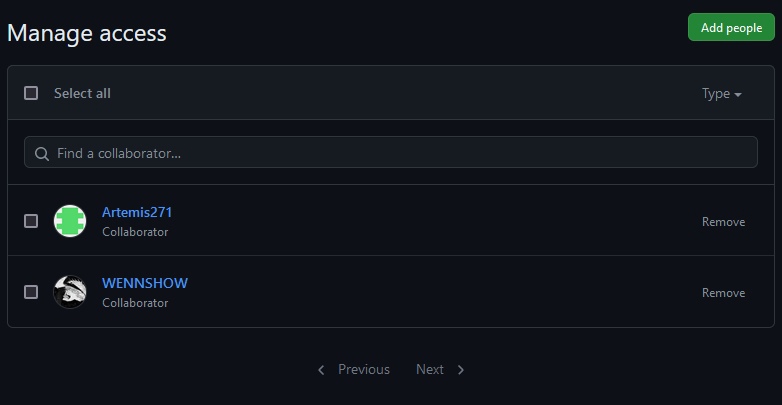


Рисунок 18 - Contributors в github

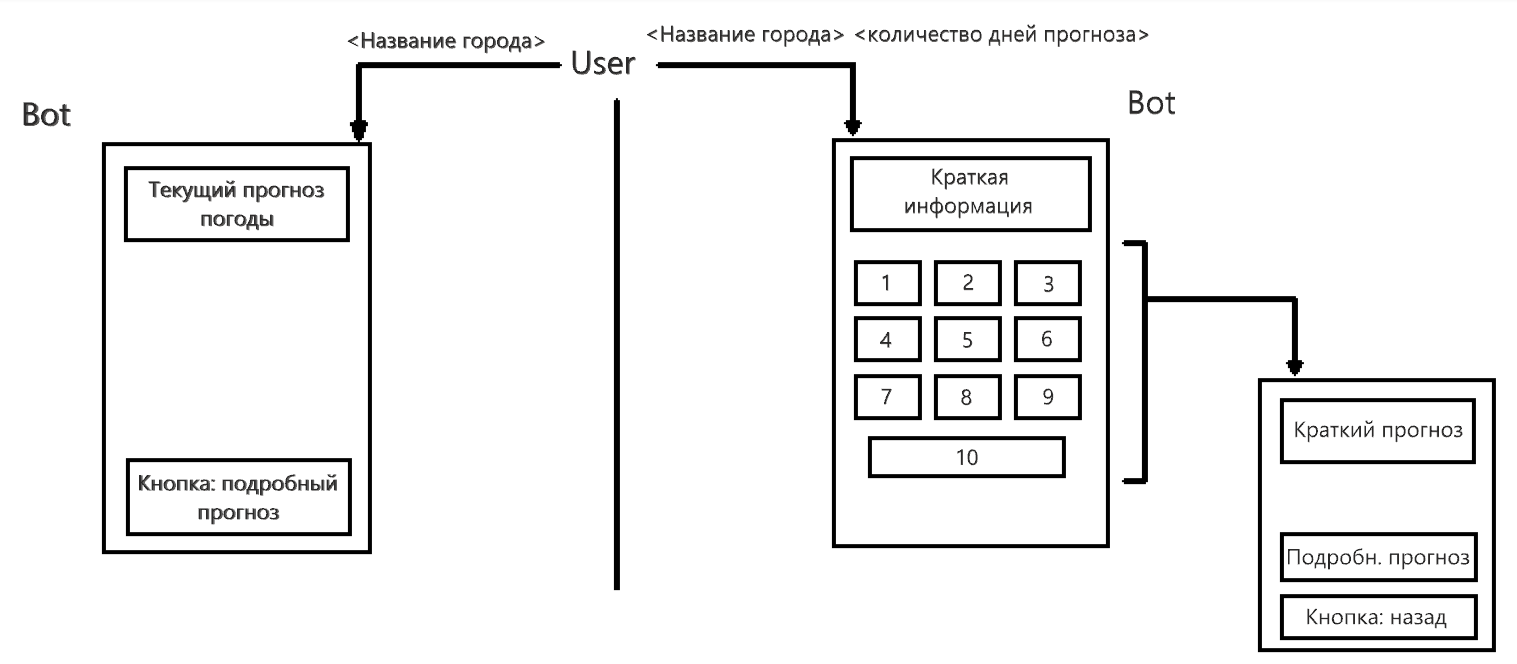


Рисунок 19 – Примерная навигация по сообщениям бота

**Тестирование и взаимодействие**

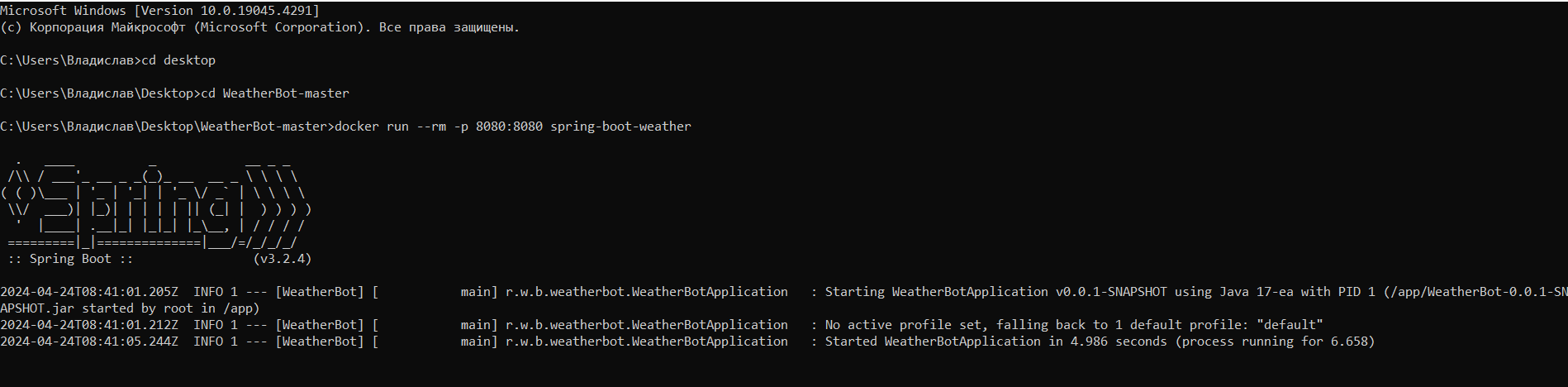
****

Рисунок 20 - Развертывание при помощи docker

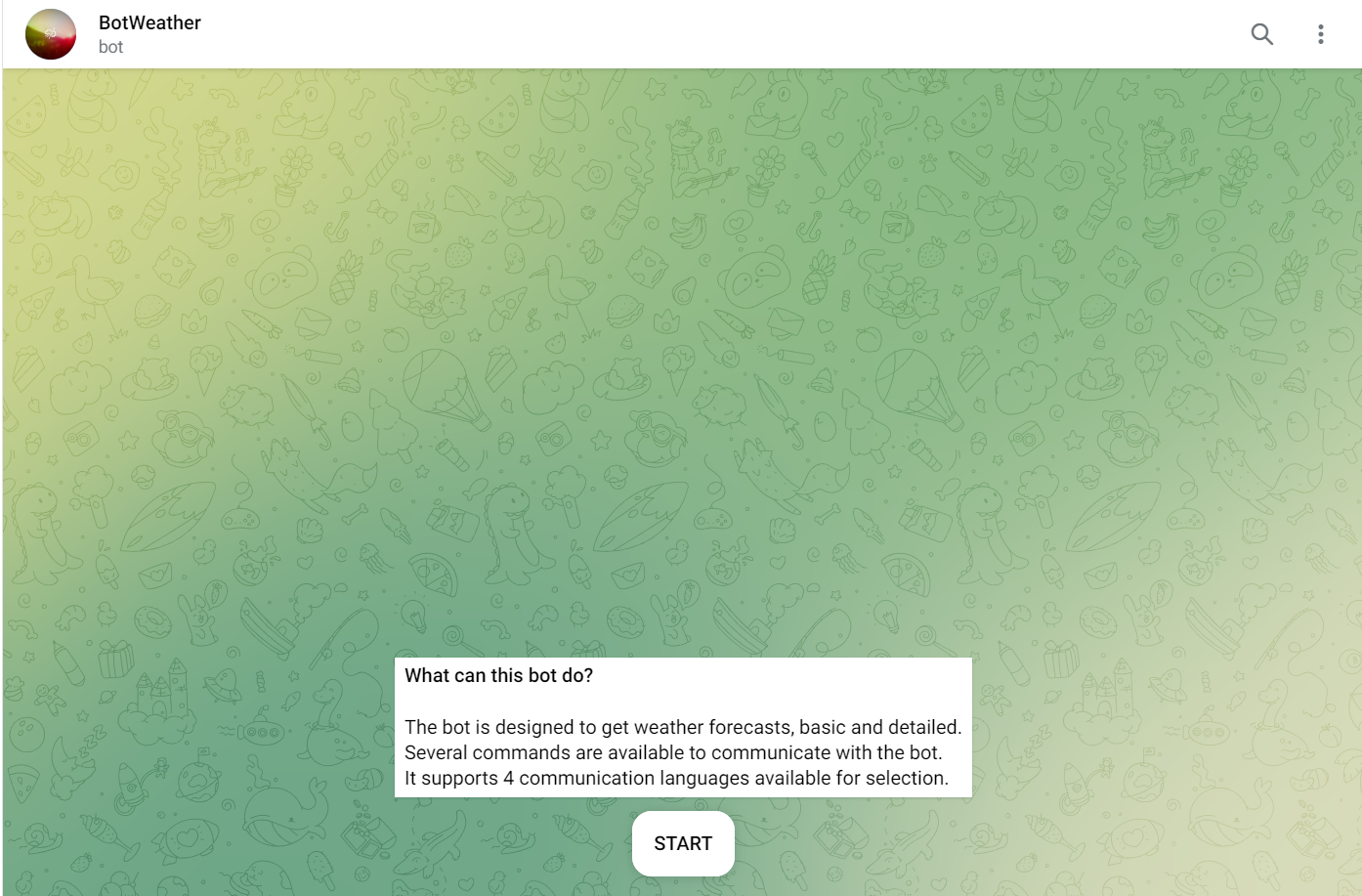


Рисунок 21 - Работа с ботом, часть 1

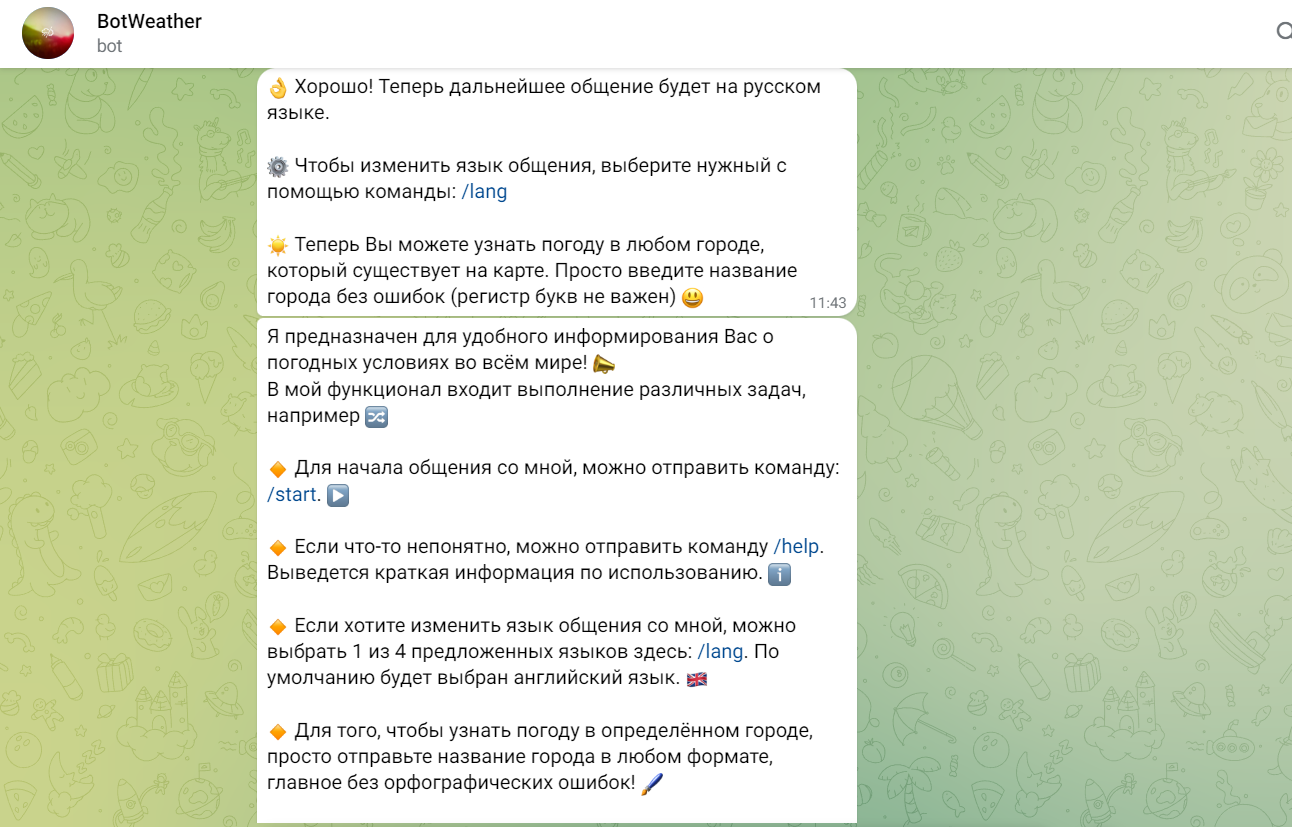


Рисунок 22 - Работа с ботом, часть 2

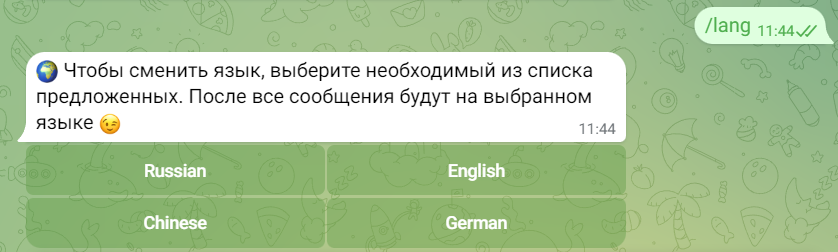


Рисунок 23 - Работа с ботом, часть 3

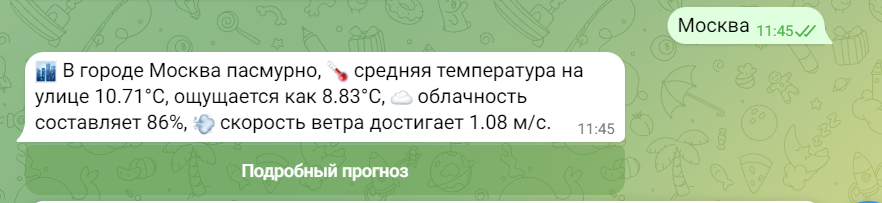


Рисунок 24 - Работа с ботом, часть 4

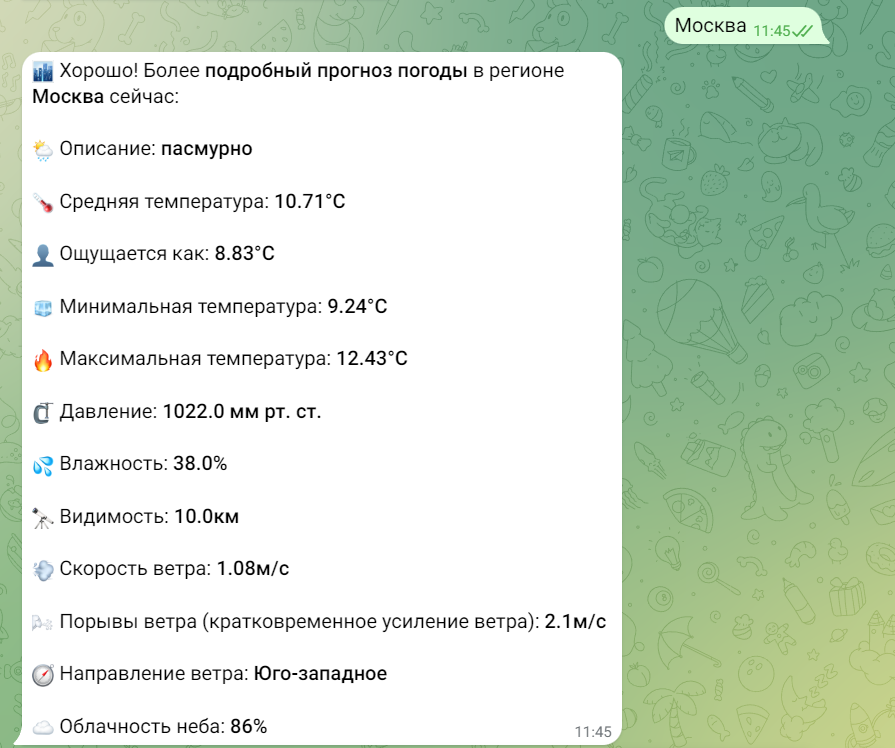


Рисунок 25 - Работа с ботом, часть 5

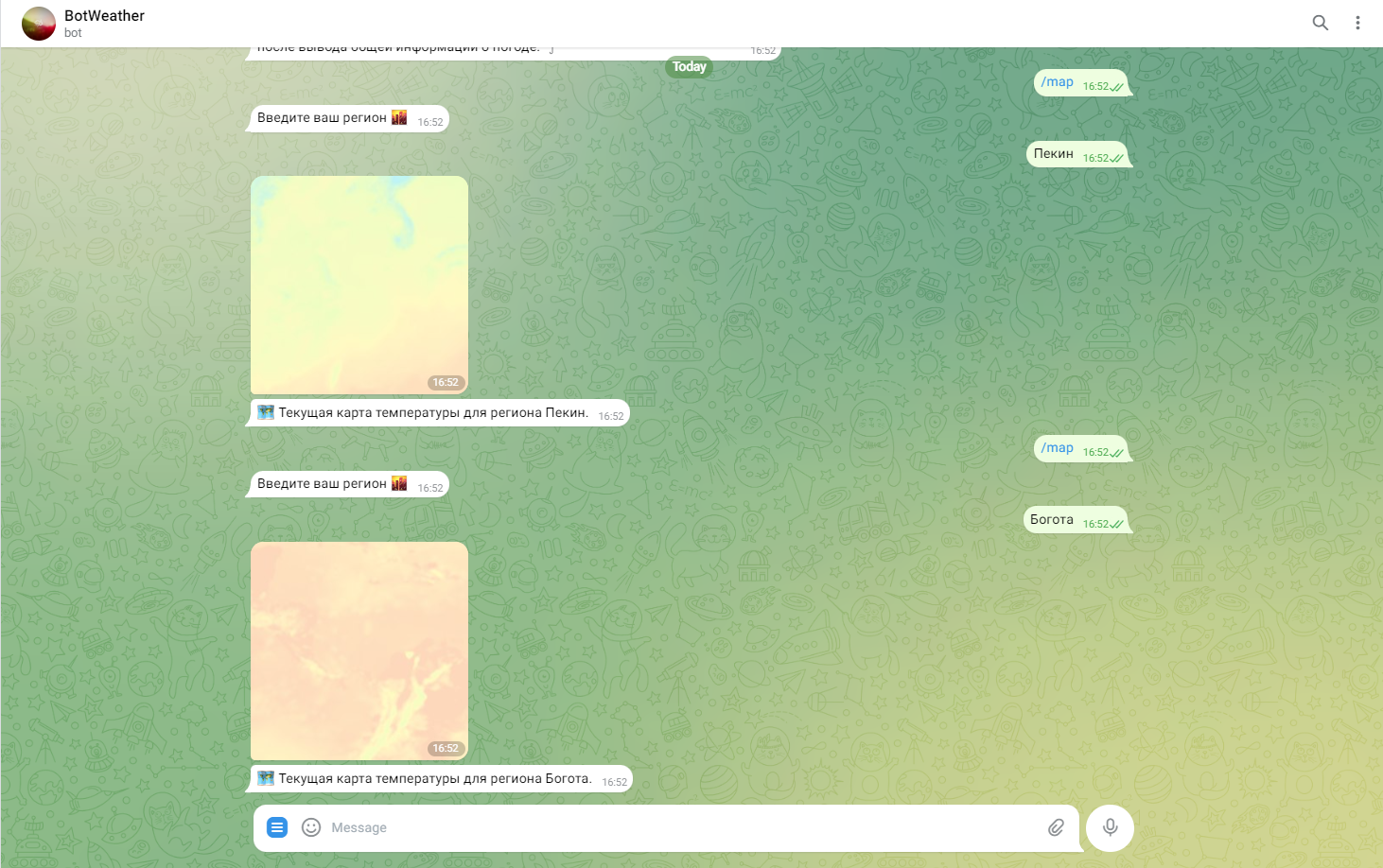


Рисунок 26 - Работа с ботом, часть 6

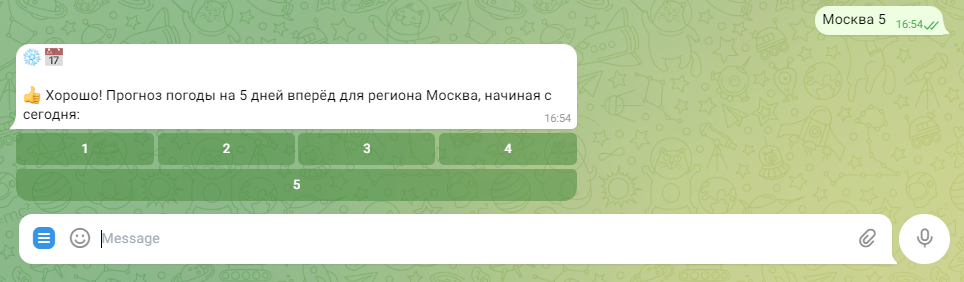


Рисунок 27 - Работа с ботом, часть 7

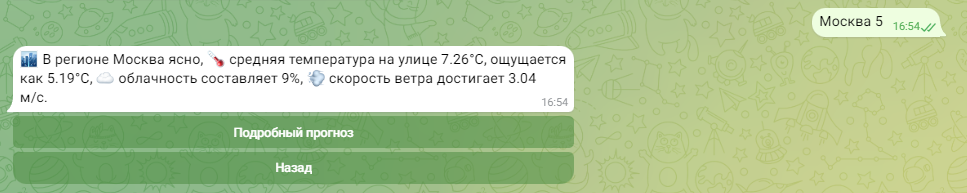


Рисунок 28 - Работа с ботом, часть 8

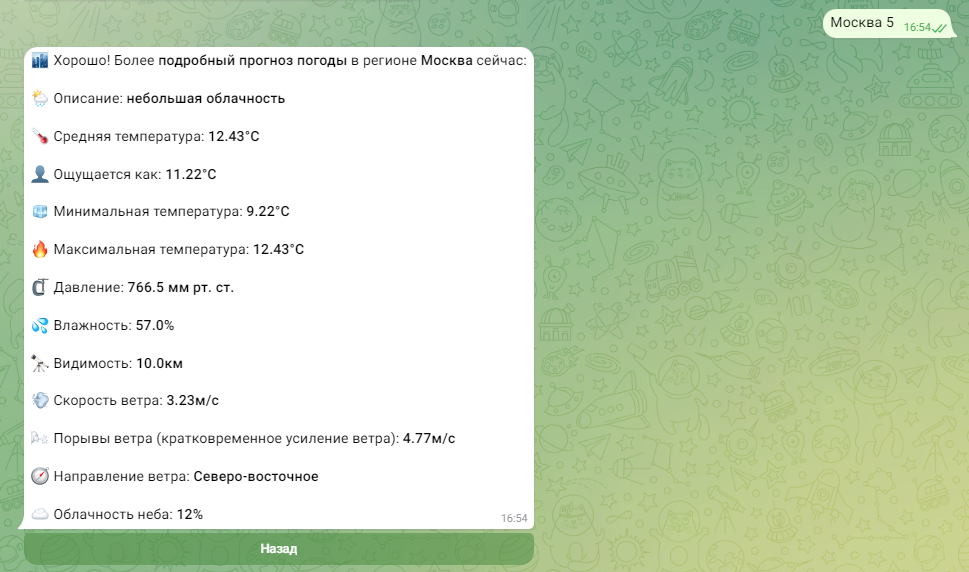


Рисунок 29 - Работа с ботом, часть 9

**Выводы**

В ходе выполнения проекта по разработке погодного телеграм-бота на Java с интеграцией OpenWeatherMap API были достигнуты значимые результаты. Бот был успешно реализован с использованием современных технологий, таких как Java и Spring Boot, что обеспечило надежную и эффективную работу приложения. Интеграция с Telegram Bot API позволила боту взаимодействовать с пользователями, принимать запросы и отправлять информацию о погоде.

Важным аспектом проекта была интеграция с внешним API OpenWeatherMap, которая позволила получать актуальные данные о погоде для указанных местоположений. Это дало боту функциональность предоставления пользователю оперативной и точной информации о текущей погоде.

Процесс тестирования позволил выявить и устранить ошибки, обеспечивая корректную работу бота. Практическое применение разработанного телеграм-бота может быть полезным для конечных пользователей, обеспечивая им легкий доступ к метеорологической информации.

Дальнейшее развитие проекта может включать расширение функциональности бота, улучшение интерфейса, оптимизацию производительности и добавление новых возможностей для более полезного и удобного пользовательского опыта. В целом, выполнение данного проекта дало ценный опыт в разработке ботов для мессенджеров и интеграции с внешними API, открывая дорогу к дальнейшим исследованиям и разработкам в этой области.