Министерство образования и науки РФ

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа «Киберфизических систем и управления»

ОТЧЕТ

по дисциплине «Цифровой практикум»

**Лабораторная работа №2**

**Выполнил:**

Студент гр. 3532703/10001 Кондрин А. С.

**Проверил**

Ассистент А.С. Свистунова

Санкт-Петербург

2021 г.

## **Задание**

Задачей этой лабораторной работы было создание Telegram-бота, который будет выполнять различные команды и предоставлять информацию пользователю. Моим персональным вариантом было создание бот-поисковик аниме, который находит аниме и дает ссылки на AniList и трейлер. Так же функция случайного аниме.

## **Код программы**

package com.ask0n;  
  
import org.telegram.telegrambots.meta.TelegramBotsApi;  
import org.telegram.telegrambots.meta.exceptions.TelegramApiException;  
import org.telegram.telegrambots.updatesreceivers.DefaultBotSession;  
  
public class Main {  
 public static void main(String[] args) {  
 try {  
 TelegramBotsApi telegramBotsApi = new TelegramBotsApi(DefaultBotSession.class);  
 telegramBotsApi.registerBot(new AnimeBot());  
 } catch (TelegramApiException e) {  
 e.printStackTrace();  
 }  
 }  
}

package com.ask0n;  
  
import org.telegram.telegrambots.meta.api.objects.InputFile;  
  
public interface Constants {  
 //settings  
 String *USERNAME* = "botname";  
 String *API\_TOKEN* = "token";  
 String *PREFIX* = "/";  
  
 //ani api  
 String *TOKEN* = "aniapi-token";  
 String *BASE\_URL* = "https://api.aniapi.com";  
 String *ANILIST\_URL* = "https://anilist.co/anime/";  
  
 //replies  
 String *START* = "<b>Привет, я AniFinder (@Ask0nBot).</b>\n" +  
 "Я помогу тебе найти любое аниме которое знаю.\n\n" +  
 "Для поиска введи название аниме (на английском/японском языке) и дождись результата.\n\n" +  
 "Мой создатель: @Ask0n";  
 String *NOT\_FOUND* = "<b>Я не смог найти твое аниме прости. =( </b>";  
 String *TRY\_AGAIN* = "<b>Слишком много результатов попробуй уточнить получше.</b>";  
 String *ERROR* = "<b>Ошибка! Ошибка! АААААА! =( </b>";  
 String *NOT\_FOUND\_COMMAND* = "Команда не найдена! Введите /help для справки.";  
 String *HELP* = "<b>Список доступных команд:</b>\n" +  
 "• /start - Краткое описание бота\n" +  
 "• /random - Случайное аниме\n" +  
 "• /help - Список команд\n" +  
 "Для поиска аниме просто введите его название (на английском/японском языке).";  
 String *CHOICE* = "Результатов так много, выбери нужный тебе";  
 String *GENRES* = "<b>Список всех доступных жанров (есть секретные жанры):</b> \n";  
  
 //images  
 InputFile *START\_IMAGE* = new InputFile("https://static.zerochan.net/Shinomiya.Kaguya.full.2527457.png");  
 InputFile *NOT\_FOUND\_IMAGE* = new InputFile("https://static.wikia.nocookie.net/v\_\_/images/5/5f/404\_not\_found.png/revision/latest?cb=20171104190424&path-prefix=vocaloidlyrics");  
 InputFile *ERROR\_IMAGE* = new InputFile("https://cdn141.picsart.com/338169980054201.jpg");  
 InputFile *NO\_IMAGE* = new InputFile("https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQd5eWeOjIjpXYmobzaxnI7NuP\_yBh8P09HD2GFw7EtVnM6NegWjc4EyjUhKyo8S04OEMw&usqp=CAU");  
 InputFile *TRY\_AGAIN\_IMAGE* = new InputFile("https://i.pinimg.com/originals/49/8c/96/498c96c102b9cfca04cb8bc5c63643a2.jpg");  
 InputFile *NOT\_FOUND\_COMMAND\_IMAGE* = new InputFile("https://cdn.discordapp.com/attachments/569495372204998668/937106823822057502/prostite.png");  
 InputFile *HELP\_IMAGE* = new InputFile("https://i.quotev.com/img/q/u/19/4/26/w6p7konwyz.jpg");  
 InputFile *CHOICE\_IMAGE* = new InputFile("https://pa1.narvii.com/6019/5f978676a2f3157525dd3695de47fc695350f0f8\_hq.gif");  
 InputFile *GENRES\_IMAGE* = new InputFile("https://toppng.com/uploads/preview/confused-anime-png-anime-question-png-gif-11563638774vp3mpnwbez.png");  
}

package com.ask0n;  
  
import java.util.Arrays;  
  
import static com.ask0n.Constants.*PREFIX*;  
  
public enum Commands {  
 *START*("start"), *HELP*("help"), *RANDOM*("random"), *GENRES*("genres"), *FIND\_BY\_GENRES*("findbygenres");  
  
 private final String command;  
  
 Commands(String command) {  
 this.command = *PREFIX* + command;  
 }  
  
 public static Commands fromString(String s) throws IllegalArgumentException {  
 return Arrays.*stream*(Commands.*values*())  
 .filter(v -> v.command.equals(s))  
 .findFirst()  
 .orElseThrow(() -> new IllegalArgumentException("Такой команды нет: " + s));  
 }  
}

package com.ask0n;  
  
import com.ask0n.domains.Anime;  
import org.telegram.telegrambots.meta.api.methods.send.SendMessage;  
import org.telegram.telegrambots.meta.api.methods.send.SendPhoto;  
import org.telegram.telegrambots.meta.api.objects.InputFile;  
import org.telegram.telegrambots.meta.api.objects.replykeyboard.InlineKeyboardMarkup;  
import org.telegram.telegrambots.meta.api.objects.replykeyboard.buttons.InlineKeyboardButton;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import static com.ask0n.Constants.\*;  
  
public class MessageService {  
 public static SendPhoto message(String chatId, InputFile image, String text) {  
 return SendPhoto.*builder*()  
 .chatId(chatId)  
 .parseMode("HTML")  
 .photo(image)  
 .caption(text)  
 .build();  
 }  
  
 public static SendMessage message(String chatId, String text) {  
 return SendMessage.*builder*()  
 .chatId(chatId)  
 .parseMode("HTML")  
 .text(text)  
 .build();  
 }  
  
 public static SendPhoto keyboardMessage(String chatId, List<Anime> animeList) {  
 final SendPhoto message = new SendPhoto();  
  
 message.setChatId(chatId);  
 message.setCaption(*CHOICE*);  
 message.setPhoto(*CHOICE\_IMAGE*);  
  
 InlineKeyboardMarkup inlineKeyboardMarkup = new InlineKeyboardMarkup();  
 List<List<InlineKeyboardButton>> keyboard = new ArrayList<>();  
  
 for (Anime anime : animeList) {  
 List<InlineKeyboardButton> buttons = new ArrayList<>();  
 InlineKeyboardButton button = new InlineKeyboardButton(anime.getTitles().get("en"));  
 button.setCallbackData(String.*valueOf*(anime.getId()));  
 buttons.add(button);  
 keyboard.add(buttons);  
 }  
  
 inlineKeyboardMarkup.setKeyboard(keyboard);  
 message.setReplyMarkup(inlineKeyboardMarkup);  
  
 return message;  
 }  
  
 public static SendPhoto animeMessage(String chatId, Anime anime) {  
 return SendPhoto.*builder*()  
 .chatId(chatId)  
 .parseMode("HTML")  
 .photo(anime.getCoverImage())  
 .caption(anime.getCaption())  
 .build();  
 }  
  
 public static SendPhoto notFoundMessage(String chatId) {  
 return SendPhoto.*builder*()  
 .chatId(chatId)  
 .parseMode("HTML")  
 .photo(*NOT\_FOUND\_IMAGE*)  
 .caption(*NOT\_FOUND*)  
 .build();  
 }  
  
 public static SendPhoto tryAgainMessage(String chatId) {  
 return SendPhoto.*builder*()  
 .chatId(chatId)  
 .parseMode("HTML")  
 .photo(*TRY\_AGAIN\_IMAGE*)  
 .caption(*TRY\_AGAIN*)  
 .build();  
 }  
  
 public static SendPhoto errorMessage(String chatId) {  
 return SendPhoto.*builder*()  
 .chatId(chatId)  
 .parseMode("HTML")  
 .photo(*ERROR\_IMAGE*)  
 .caption(*ERROR*)  
 .build();  
 }  
  
 public static SendPhoto notFoundCommandMessage(String chatId) {  
 return SendPhoto.*builder*()  
 .chatId(chatId)  
 .parseMode("HTML")  
 .photo(*NOT\_FOUND\_COMMAND\_IMAGE*)  
 .caption(*NOT\_FOUND\_COMMAND*)  
 .build();  
 }  
}

package com.ask0n;  
  
import com.ask0n.domains.AniResponse;  
import com.ask0n.domains.Anime;  
import com.ask0n.domains.Pagination;  
import com.ask0n.domains.filters.AnimeFilter;  
import com.ask0n.enums.Commands;  
import lombok.SneakyThrows;  
import org.apache.logging.log4j.LogManager;  
import org.apache.logging.log4j.Logger;  
import org.telegram.telegrambots.bots.TelegramLongPollingBot;  
import org.telegram.telegrambots.meta.api.methods.send.SendPhoto;  
import org.telegram.telegrambots.meta.api.methods.updatingmessages.DeleteMessage;  
import org.telegram.telegrambots.meta.api.objects.Update;  
  
import java.util.Arrays;  
import java.util.List;  
  
import static com.ask0n.Constants.\*;  
  
public class AnimeBot extends TelegramLongPollingBot {  
 private static final Logger *log* = LogManager.*getLogger*(AniClient.class);  
 private final AniClient aniClient = AniClient.*builder*().token(*TOKEN*).build();  
  
 @Override  
 public String getBotUsername() {  
 return Constants.*USERNAME*;  
 }  
  
 @Override  
 public String getBotToken() {  
 return Constants.*API\_TOKEN*;  
 }  
  
 @SneakyThrows  
 @Override  
 public void onUpdateReceived(Update update) {  
 try {  
 if (update.hasMessage() && update.getMessage().hasText()) {  
 *log*.info(String.*format*("Received new message: [from: %s, message: %s]",  
 update.getMessage().getFrom().getUserName(),  
 update.getMessage().getText()));  
 if (update.hasMessage() && update.getMessage().hasText() && update.getMessage().getText().startsWith(*PREFIX*)) {  
 handleCommand(update);  
 } else {  
 execute(handleText(update));  
 }  
 return;  
 }  
  
 if (update.hasCallbackQuery()) {  
 *log*.info(String.*format*("Received callback query: [from: %s, data: %s]",  
 update.getCallbackQuery().getFrom().getUserName(),  
 update.getCallbackQuery().getData()));  
 DeleteMessage deleteMessage = new DeleteMessage();  
 deleteMessage.setChatId(update.getCallbackQuery().getFrom().getId().toString());  
 deleteMessage.setMessageId(update.getCallbackQuery().getMessage().getMessageId());  
 execute(deleteMessage);  
 execute(handleCallback(update));  
 }  
 } catch (IllegalArgumentException e) {  
 *log*.warn(e.getMessage());  
 execute(MessageService.*notFoundCommandMessage*(update.getMessage().getChatId().toString()));  
 } catch (Exception e) {  
 *log*.error(e.getMessage());  
 *log*.error(Arrays.*toString*(e.getStackTrace()));  
 }  
 }  
  
 public SendPhoto handleCallback(Update update) throws Exception {  
 final String chatId = update.getCallbackQuery().getFrom().getId().toString();  
 final String id = update.getCallbackQuery().getData();  
 final AniResponse<Anime> response = aniClient.getAnime(Long.*valueOf*(id)).get();  
 if (response.getStatusCode() != 200) {  
 *log*.info("Not found - " + update.getMessage().getFrom().getUserName());  
 return MessageService.*notFoundMessage*(chatId);  
 }  
 *log*.info(String.*format*("Found: [title: %s for: %s]",  
 response.getData().getTitles().get("en"),  
 update.getCallbackQuery().getFrom().getUserName()));  
 return MessageService.*animeMessage*(chatId, response.getData());  
 }  
  
 public SendPhoto handleText(Update update) throws Exception {  
 final String chatId = update.getMessage().getChatId().toString();  
 final String name = update.getMessage().getText();  
 final AniResponse<Pagination<Anime>> response = aniClient  
 .getAnimeList(AnimeFilter.*builder*().title(name).build())  
 .get();  
 if (response.getStatusCode() != 200) {  
 *log*.info("Not found - " + update.getMessage().getFrom().getUserName());  
 return MessageService.*notFoundMessage*(chatId);  
 }  
 if (response.getData().getCount() > 100) {  
 *log*.info("Found over 100 results - " + update.getMessage().getFrom().getUserName());  
 return MessageService.*tryAgainMessage*(chatId);  
 }  
 if (response.getData().getCount() == 1) {  
 *log*.info(String.*format*("Found: [title: %s for: %s]",  
 response.getData().getDocuments().get(0).getTitles().get("en"),  
 update.getMessage().getFrom().getUserName()));  
 return MessageService.*animeMessage*(chatId, response.getData().getDocuments().get(0));  
 }  
 return MessageService.*keyboardMessage*(chatId, response.getData().getDocuments());  
 }  
  
 public void handleCommand(Update update) throws Exception {  
 final String chatId = update.getMessage().getChatId().toString();  
 final String[] splitMessage = update.getMessage().getText().split("\\s+", 2);  
 final String command = splitMessage[0];  
 switch (Commands.*fromString*(command)) {  
 case *START*:  
 execute(MessageService.*message*(chatId, *START\_IMAGE*, *START*));  
 break;  
 case *HELP*:  
 execute(MessageService.*message*(chatId, *HELP\_IMAGE*, *HELP*));  
 break;  
 case *RANDOM*:  
 final AniResponse<List<Anime>> response = aniClient.getRandomAnime().get();  
 final Anime anime = response.getData().get(0);  
 *log*.info(String.*format*("Found: [title: %s for: %s]",  
 anime.getTitles().get("en"),  
 update.getMessage().getFrom().getUserName()));  
 execute(MessageService.*animeMessage*(chatId, anime));  
 break;  
 }  
 }  
}

package com.ask0n;  
  
import com.ask0n.domains.AniResponse;  
import com.ask0n.domains.Anime;  
import com.ask0n.domains.Pagination;  
import com.ask0n.domains.filters.AnimeFilter;  
import com.fasterxml.jackson.annotation.JsonInclude;  
import com.fasterxml.jackson.core.JsonProcessingException;  
import com.fasterxml.jackson.core.type.TypeReference;  
import com.fasterxml.jackson.databind.DeserializationFeature;  
import com.fasterxml.jackson.databind.PropertyNamingStrategy;  
import com.fasterxml.jackson.databind.SerializationFeature;  
import com.fasterxml.jackson.databind.json.JsonMapper;  
import lombok.AllArgsConstructor;  
import lombok.Builder;  
import lombok.EqualsAndHashCode;  
import lombok.SneakyThrows;  
import org.apache.logging.log4j.LogManager;  
import org.apache.logging.log4j.Logger;  
  
import java.net.URI;  
import java.net.http.HttpClient;  
import java.net.http.HttpRequest;  
import java.net.http.HttpResponse;  
import java.util.List;  
import java.util.concurrent.CompletableFuture;  
  
@Builder  
@AllArgsConstructor  
@EqualsAndHashCode  
public class AniClient {  
 private static final Logger *log* = LogManager.*getLogger*(AniClient.class);  
 private static final String *ENDPOINT* = "https://api.aniapi.com/v1/";  
 private static final HttpClient *HTTP\_CLIENT* = HttpClient.*newBuilder*()  
 .version(HttpClient.Version.*HTTP\_2*)  
 .build();  
 private static final JsonMapper *mapper* = JsonMapper.*builder*()  
 .findAndAddModules()  
 .enable(SerializationFeature.*WRITE\_ENUM\_KEYS\_USING\_INDEX*)  
 .enable(DeserializationFeature.*ACCEPT\_EMPTY\_STRING\_AS\_NULL\_OBJECT*)  
 .enable(DeserializationFeature.*ACCEPT\_EMPTY\_ARRAY\_AS\_NULL\_OBJECT*)  
 .disable(SerializationFeature.*WRITE\_DATES\_AS\_TIMESTAMPS*)  
 .serializationInclusion(JsonInclude.Include.*NON\_NULL*)  
 .configure(DeserializationFeature.*FAIL\_ON\_UNKNOWN\_PROPERTIES*, false)  
 .propertyNamingStrategy(PropertyNamingStrategy.*SNAKE\_CASE*)  
 .build();  
  
 private final String token;  
  
 @SneakyThrows  
 private <T> CompletableFuture<T> fetch(String uri, TypeReference<T> clazz) {  
 *log*.info(String.*format*("GET | %s%s", *ENDPOINT*, uri));  
 HttpRequest request = HttpRequest.*newBuilder*()  
 .uri(URI.*create*(*ENDPOINT* + uri))  
 .header("Content-Type", "application/json")  
 .header("Authorization", "Bearer " + token)  
 .GET()  
 .build();  
  
 return *HTTP\_CLIENT* .sendAsync(request, HttpResponse.BodyHandlers.*ofString*())  
 .thenApplyAsync(res -> {  
 try {  
 return *mapper*.readValue(res.body(), clazz);  
 } catch (JsonProcessingException e) {  
 e.printStackTrace();  
 return null;  
 }  
 });  
 }  
  
 public CompletableFuture<AniResponse<Anime>> getAnime(Long id) {  
 return fetch("anime/" + id, new TypeReference<>() {  
 });  
 }  
  
 public CompletableFuture<AniResponse<Pagination<Anime>>> getAnimeList(AnimeFilter filter) {  
 return fetch("anime" + filter.toQueryString(), new TypeReference<>() {  
 });  
 }  
  
 public CompletableFuture<AniResponse<List<Anime>>> getRandomAnime() {  
 return fetch("random/anime/1/true", new TypeReference<>() {  
 });  
 }  
}

package com.ask0n.enums;  
  
import java.util.Arrays;  
  
import static com.ask0n.Constants.*PREFIX*;  
  
public enum Commands {  
 *START*("start"), *HELP*("help"), *RANDOM*("random"), *GENRES*("genres"), *FIND\_BY\_GENRES*("findbygenres");  
  
 private final String command;  
  
 Commands(String command) {  
 this.command = *PREFIX* + command;  
 }  
  
 public static Commands fromString(String s) throws IllegalArgumentException {  
 return Arrays.*stream*(Commands.*values*())  
 .filter(v -> v.command.equals(s))  
 .findFirst()  
 .orElseThrow(() -> new IllegalArgumentException("Not found: " + s));  
 }  
}

package com.ask0n.enums;  
  
public enum AnimeStatusEnum {  
 *FINISHED*,  
 *RELEASING*,  
 *NOT\_YET\_RELEASED*,  
 *CANCELLED*}

package com.ask0n.enums;  
  
public enum AnimeSeasonEnum {  
 *WINTER*,  
 *SPRING*,  
 *SUMMER*,  
 *FALL*,  
 *UNKNOWN*}

package com.ask0n.enums;  
  
public enum AnimeFormatEnum {  
 *TV*,  
 *TV\_SHORT*,  
 *MOVIE*,  
 *SPECIAL*,  
 *OVA*,  
 *ONA*,  
 *MUSIC*}

package com.ask0n.domains;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import lombok.ToString;  
  
import java.util.List;  
  
@Data  
@ToString  
@NoArgsConstructor  
@AllArgsConstructor  
public class Pagination<T> {  
 private Integer currentPage;  
 private Integer count;  
 private List<T> documents;  
 private Integer lastPage;  
}

package com.ask0n.domains;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import lombok.ToString;  
  
@Data  
@ToString  
@NoArgsConstructor  
@AllArgsConstructor  
public class AniResponse<T> {  
 private Integer statusCode;  
 private String message;  
 private T data;  
 private String version = "1";  
}

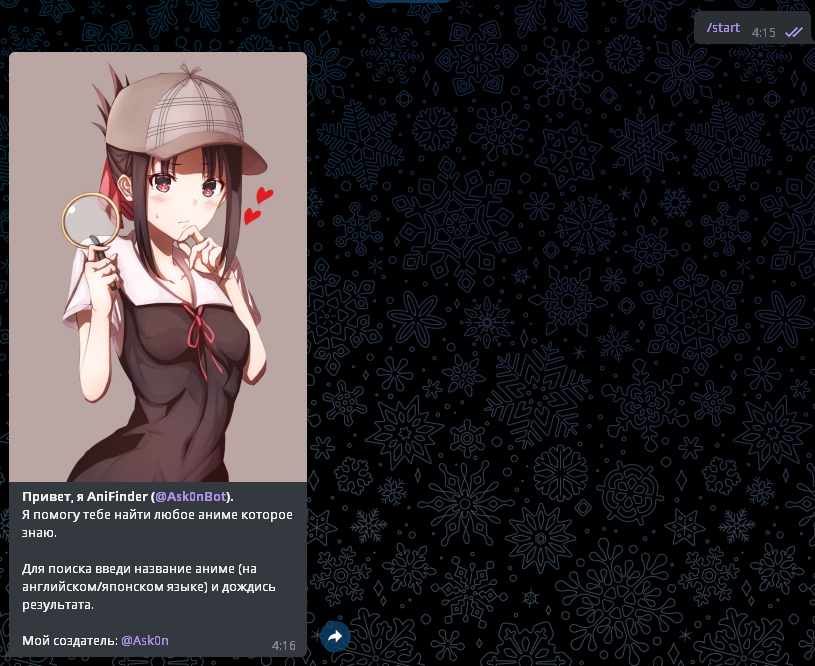
package com.ask0n.domains;  
  
import com.ask0n.enums.AnimeFormatEnum;  
import com.ask0n.enums.AnimeSeasonEnum;  
import com.ask0n.enums.AnimeStatusEnum;  
import lombok.Data;  
import lombok.\*;  
import org.telegram.telegrambots.meta.api.objects.InputFile;  
  
import java.util.Date;  
import java.util.List;  
import java.util.Map;  
  
import static com.ask0n.Constants.*ANILIST\_URL*;  
import static com.ask0n.Constants.*NO\_IMAGE*;  
  
@Data  
@ToString  
@NoArgsConstructor  
@AllArgsConstructor  
@EqualsAndHashCode  
public class Anime {  
 private Long id;  
 private Long anilistId;  
 private Long malId;  
 private AnimeFormatEnum format;  
 private AnimeStatusEnum status;  
 private Map<String, String> titles;  
 private Map<String, String> descriptions;  
 private Date startDate;  
 private Date endDate;  
 private AnimeSeasonEnum seasonPeriod;  
 private Integer seasonYear;  
 private Integer episodesCount;  
 private Integer episodeDuration;  
 private String trailerUrl;  
 private String coverImage;  
 private String coverColor;  
 private String bannerImage;  
 private List<String> genres;  
 private Long sequel;  
 private Long prequel;  
 private Integer score;  
  
 public InputFile getCoverImage() {  
 return coverImage != null ? new InputFile(coverImage) : *NO\_IMAGE*;  
 }  
  
 public String getCaption() {  
 final StringBuilder builder = new StringBuilder();  
 builder.append("Название:\n");  
 titles.forEach((k, v) -> builder.append("<b>").append(v).append("</b> (").append(k).append(")\n"));  
 if (seasonYear != null && seasonYear > 0) builder.append("\nГод: ").append(seasonYear).append("\n\n");  
 builder.append("Ссылки:\n");  
 builder.append("<a href=\"").append(*ANILIST\_URL*).append(anilistId).append("\">AniList</a>\n");  
 builder.append("<a href=\"").append(trailerUrl).append("\">Трейлер</a>\n\n");  
 builder.append("Оценка: ").append(score).append(" | 100\n\n");  
 genres.forEach(g -> builder.append("#").append(g.replaceAll("([\\s]|-)", "\_")).append(" "));  
 return builder.toString();  
 }  
}

package com.ask0n.domains.filters;  
  
import lombok.NoArgsConstructor;  
import lombok.experimental.SuperBuilder;  
  
import java.net.URLEncoder;  
import java.nio.charset.StandardCharsets;  
import java.util.Map;  
import java.util.stream.Collectors;  
  
@SuperBuilder  
@NoArgsConstructor  
public abstract class Filter {  
 protected abstract Map<String, String> getParams();  
  
 public String toQueryString() {  
 Map<String, String> params = getParams();  
  
 return params.isEmpty() ? ""  
 : "?" + params.entrySet().stream()  
 .filter(entry -> entry.getValue() != null)  
 .filter(entry -> !entry.getValue().isBlank())  
 .map(entry -> entry.getKey() + "=" + URLEncoder.*encode*(entry.getValue(), StandardCharsets.*UTF\_8*))  
 .collect(Collectors.*joining*("&"));  
 }  
}

package com.ask0n.domains.filters;  
  
import lombok.\*;  
import lombok.experimental.SuperBuilder;  
  
import java.util.HashMap;  
import java.util.Map;  
  
@Data  
@SuperBuilder  
@NoArgsConstructor  
@AllArgsConstructor  
@EqualsAndHashCode(callSuper = true)  
public class AnimeFilter extends Filter {  
 private String title;  
 @Builder.Default  
 private Boolean nsfw = true;  
  
 @Override  
 protected Map<String, String> getParams() {  
 Map<String, String> params = new HashMap<>();  
 params.put("title", title);  
 params.put("nsfw", nsfw.toString());  
 return params;  
 }  
}

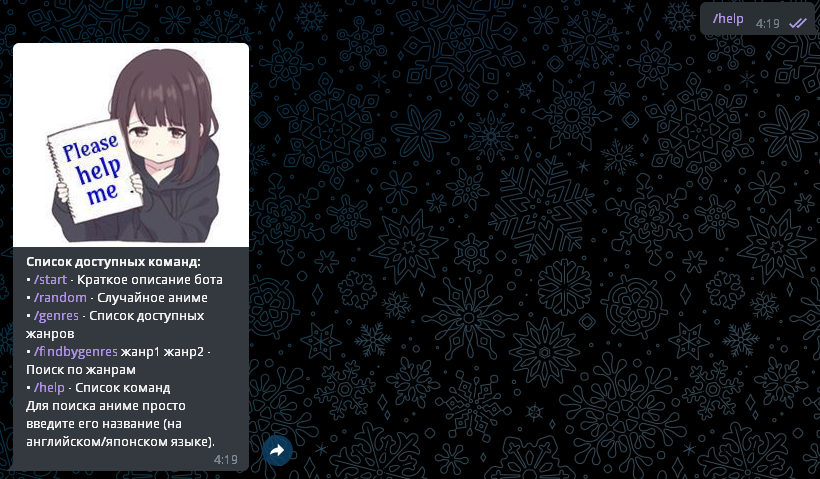
**Пример работы программы**

Входные данные: /start

Выходные данные:  


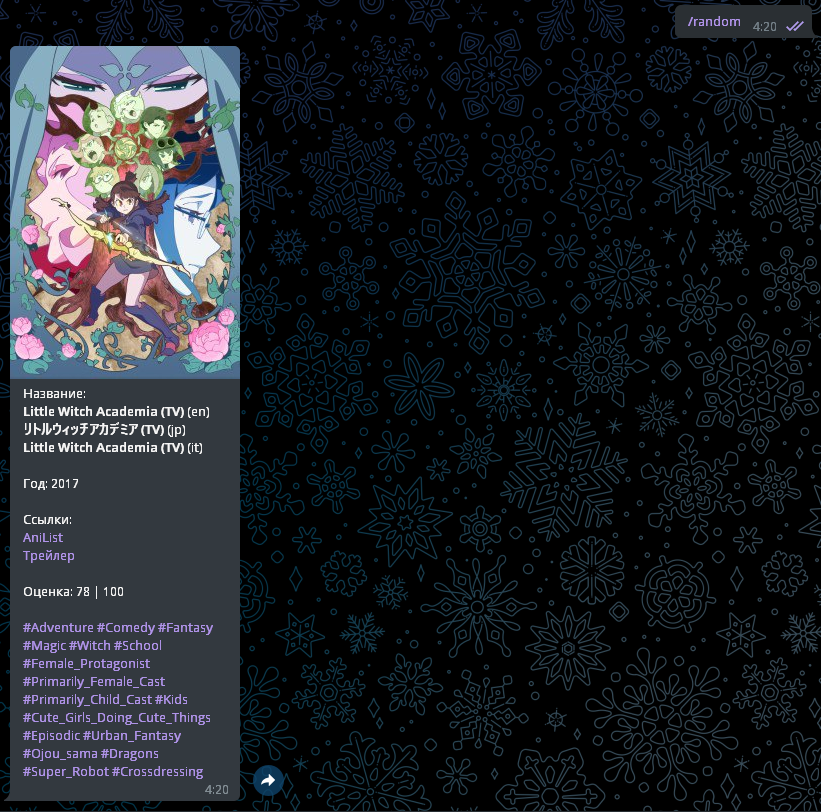
Входные данные: /help

Выходные данные:



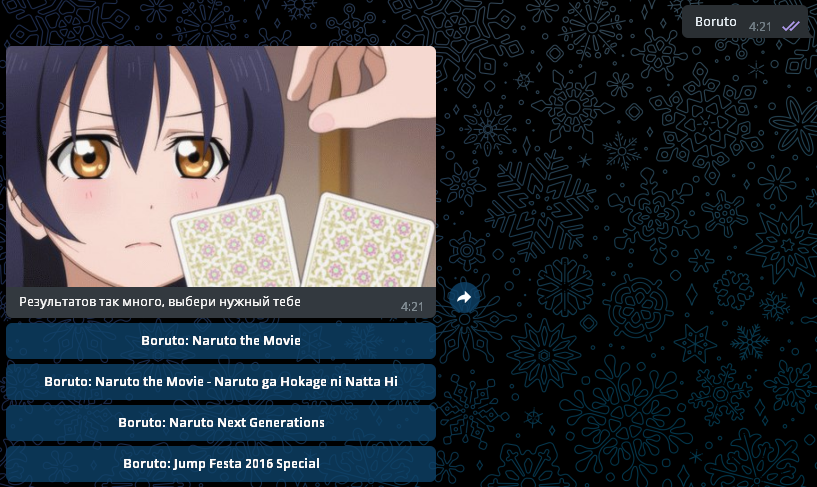
Входные данные: /random

Выходные данные:

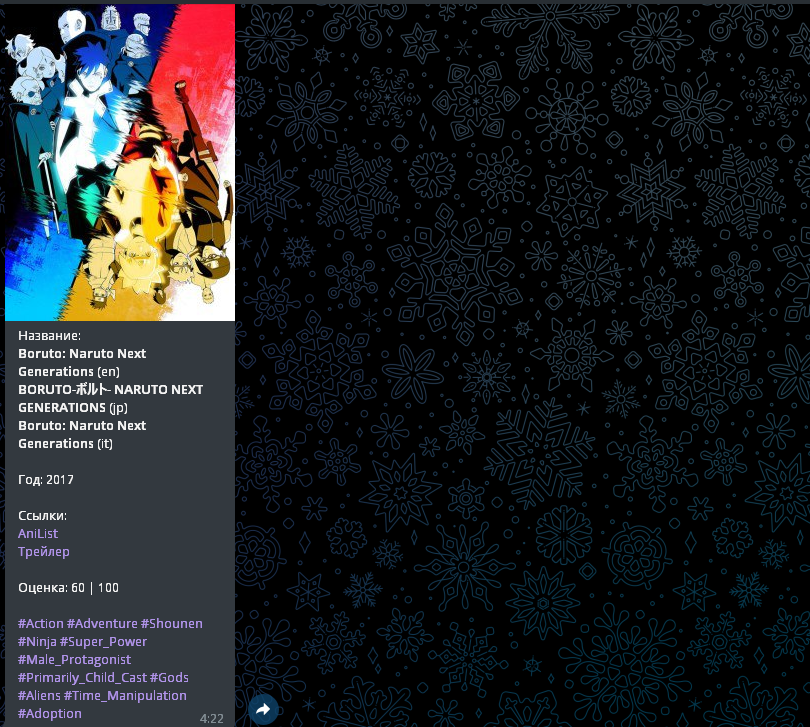


Входные данные: Boruto

Выходные данные:



Нажимаем 3 кнопку.



## **Вывод**

Создание данного Telegram – бота помогло развить навыки в написании кода для корректной работы бота. Также я нашел много новых и интересных библиотек для работы как с ботом, так и с открытыми api.