Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ

УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

**Реализация чат-бота на Telegram.Bot**

КУРСОВАЯ РАБОТА

студента 2 курса 241 группы

направления 02.03.03 Математическое обеспечение и администрирование информационных систем

факультета КНиИТ

**Емельянова Дениса Олеговича**

Научный руководитель

Старший преподаватель \_\_\_\_\_\_\_\_\_\_ Портенко М.С.

подпись, дата

Зав. кафедрой информатики и программирования,

кандидат физико-математических наук \_\_\_\_\_\_\_\_\_\_ Огнева М.В. подпись, дата

Саратов 2018

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_gjdgxs)

[**Возможности и примеры применение чат-ботов** 4](#_30j0zll)

[**Платформа для разработки чат-бота Telegram** 7](#_1fob9te)

[**Реализация проекта** 11](#_tyjcwt)

[**Список литературы** 22](#_3dy6vkm)

[**Приложения А.** 23](#_44sinio)

# **ВВЕДЕНИЕ**

Социальные сети, с момента их появления все больше проникают в нашу жизнь. Это происходит потому, что они позволяют быстро получать и давать информацию. В связи с этим стали развиваться чат-боты, которые позволяют сэкономить время на поиске нужного запроса.

Одним из видов чат-бота является TelegramBot. Он помогает выполнять разные действия: переводить и комментировать(@ytranslatebot), обучать и тестировать (@InMindTechBot), искать и находить (@fmusbot), спрашивать и отвечать(@AccentsBot), играть и развлекать(@daenerys\_cash\_game\_bot), транслировать и агрегировать(@videollamada\_bot), встраиваться в другие сервисы и платформы(@airtrack\_bot), взаимодействовать с датчиками и вещами, подключенными к интернету. И это далеко не все возможности, которые представляет платформа для их создания. Чат-боты на платформе Telegram активно развиваются и набирают все больше популярность.

Для бизнеса Telegram применяется достаточно широко. Это обуславливается тем, что возможности мессенджера позволяют оперативно делиться новостями, акциями и рассылками с большим количеством пользователей. Боты уменьшают нагрузку на менеджеров, беря на себя выполнение монотонных заданий.

Цель работы: создание чат-бота на платформе Telegram с помощью библиотеки Telegram.Bot для реализации поиска ближайшего кинотеатра относительно пользователя.

Задачи:

1. Анализ платформы Telegram;
2. Разработка структуры проекта;
3. Реализация проекта;
4. Тестирование и анализ недочетов.

# **Возможности и примеры применение чат-ботов**

Чат-бот – это робот-помощник, который позволяет выполнять определенные функции, не привлекая людей, а работа будет выполнена моментально.

Они делают жизнь людей намного проще, – заказать пиццу или такси, забронировать билеты или купить платье можно теперь не выходя из мессенджера.

Их использование, с одной стороны, полезно для клиентов, т.к. они могут получить нужную информацию или совершать какие-либо действия более простым и удобным способом, и, с другой стороны, это выгодно для компаний, которые с помощью чатботов могут продвинуть свой бренд, собрать от ЦА необходимые сведения и повысить ее лояльность.

Об эффективности использования чатботов рассказывал Марк Цукерберг на конференции F8 в 2016 году, – на его взгляд, они существенно сокращают время обслуживания клиентов и являются одним из самых актуальных трендов на сегодняшний день [1].

Вся переписка с ним ведётся через обычный чат Telegram. Бот получает в виде сообщения команды, которые он готов выполнять круглосуточно. Его основная задача ответить на вопрос пользователя, согласно своей программе. Боты помогают, экономят время и управлять ими очень просто.

На сегодня роботы Telegram могут:

* проводить обучение;
* развлекать и играть с пользователем;
* работать поисковиками в интернете;
* скачивать текстовую информацию, видео или аудио;
* быть обычной напоминалкой;
* участвовать в групповых чатах, допустим, для согласования времени встречи, оптимальной для всех участников;
* комментировать нужные статьи;
* использоваться для управления умным домом и др.;
* делать покупки.

Существует множество ботов, которыми пользуются ежедневно. Одними из популярных являются:

1. @TGhost\_bot – бот-файлообменник;
2. @InMindTechBot – позволяет изучать английский язык с возможностью практиковаться в общении со случайным собеседником;
3. @maksobot – озвучивает сообщения пользователя;
4. @flibustafreebookbot - Бот для поиска и скачивания книг;
5. @ytranslatebot - Яндекс Переводчик - с помощью этого бота можно перевести свои сообщения на любой из 12 языков.

Боты являются посредниками между человеком и многочисленными веб-службами. Одно из главных преимуществ – это общая оболочка, так как Telegram позволяет на своей платформе реализовать вышеперечисленное.

Несомненный плюс в экономии времени за счёт уменьшения количества приложений в гаджете. Люди всегда хотят получать всё, что им нужно, в одном окне. Теперь нет необходимости в установке лишних приложений для погоды, новостей и т. п., что было особенно критично для маломощного смартфона. Теперь всё это заменяет Telegram с возможностью установки необходимых роботов.

Принцип работы: выбирается бот, отправляется ему текстовое сообщение (команду) и через доли секунды приходит ответ.

Достоинства:

* Круглосуточная помощь — по сути, их работу остановит только авария на сервере, что случается крайне редко.
* Удобство использования — большинство команд находится в меню бота.
* Ответы приходят за несколько секунд.
* Для работы используются ресурсы сторонних серверов, так что мощности вашего устройства не задействованы.
* Безопасность. Ботам не доступны персональные данные пользователя. Они обрабатывают только команды из чата.
* Установка дополнительных программ не требуется [2].

# **Платформа для разработки чат-бота Telegram**

Telegram в своей сущности – из популярнейших текстовых мессенджеров, имеющий клиенты для всех современных платформ : Android, iOS (iPhone/iPad), Windows Phone, Windows, macOS, Linux, а также веб-версия, доступная на всех веб-браузерах, выпущенных после 2012 года. Рисунок 1 демонстрирует график роста общей аудитории.

С точки построения чат-ботов Telegram обладает инструментарием под названием Botfather. Он позволяет быстро настроить внешнее представление бота для пользователей (его никнейм, имя пользователя, аватар), а также сгенерировать ключ для Telegram API, посредством которого будет взаимодействие пользователя с сущностью бота, реализующей исполняемый код. Таким образом, разработка программы, выполняющей алгоритмы работы бота, не имеет жесткой привязки к платформе.

Статистика аудитории Telegram на январь 2018:

* 180 миллионов активных пользователей в месяц
* каждый день в Telegram регистрируется минимум 500 тысяч пользователей. Рисунок 2 демонстрирует динамику роста аудитории Telegram 2016-2017 г.
* сообщения: ежедневно отправляется 70 миллиардов сообщений
* количество ботов: в Telegram активны 800 тысяч ботов, регулярно используемые 52 миллионами человек.
* каналы: каналами Telegram пользуются 80 миллионов человек, каждый месяц генерируя 30 миллиардов просмотров [3].

Изображение выглядит как снимок экрана

Описание создано с очень высокой степенью достоверности

Рисунок 1 - Общая аудитория

Изображение выглядит как карта, текст

Описание создано с очень высокой степенью достоверности

Рисунок 2- Динамика роста аудитории Telegram

**Средства и методы разработки**

**Язык программирования**

Для реализации проекта был выбран язык программирования C#.

Язык C#, оптимизированный для программирования в Windows, вместе с Visual Studio позволяет сфокусироваться на непосредственных задачах.

C# и Visual Studio позволят без дополнительных усилий выполнять следующие задачи:

* Быстро создавать приложения. Программировать на C# очень просто. Это мощный, легко осваиваемый язык, а Visual Studio позволяет автоматизировать большинство процессов.
* Фокусироваться на решении реальных задач**.**За конечный результат работы, разумеется, отвечает разработчик. Но IDE позволяет концентрироваться на глобальных вещах, взяв на себя:
* слежение за всеми проектами;
* упрощенное редактирование кода;
* отслеживание графики, аудиофайлов, значков и прочих ресурсов;
* управление данными [4].

**СУБД**

Для хранения данных о пользователях, группах, новостях и другой информации необходимо использовать базу данных. В качестве системы управления базой данных была выбрана встраиваемая реляционная СУБД MC SQL.

MySQL – это система управления базами данных (СУБД) с открытым кодом. Это высокопроизводительная и масштабируемая СУБД с множеством программных интерфейсов. Она обладает огромными функциональными возможностями и подходит для решения самых разных задач [8].

Преимущества MС SQL:

* Обладает высокой степенью производительности и отказоустойчивости;
* Является многопользовательской СУБД и работает по принципу «клиент-сервер»;
* Клиентская часть системы поддерживает создание пользовательских запросов и их отправку для обработки на сервер.
* Тесная интеграция с операционной системой Windows;
* Поддержка удаленных подключений;
* Поддержка популярных типов данных, а также возможность создания триггеров и хранимых процедур;
* Встроенная поддержка ролей пользователей;
* Расширенная функция резервного копирования баз данных;
* Высокая степень защищенности [5].

**Telegram Bot API**

Telegram Bot API – это интерфейс, основанный на HTTP, созданный для разработчиков, заинтересованных в создании ботов для Telegram. Чтобы воспользоваться функциями Telegram Bot API необходимо получить специальный ключ аутентификации. Каждому боту при создании присваивается свой уникальный ключ.

Запросы к Telegram Bot API выглядят следующим образом: https://api.telegram.org/bot<token>/команда, где <token> – это ключ аутентификации, полученный при регистрации бота, команда – обращения к Telegram Bot API [6].

# **Реализация проекта**

Прежде чем начинать разработку, бота необходимо зарегистрировать и получить его уникальный id, являющийся одновременно и токеном. Для этого в Telegram существует специальный бот —@BotFather.

Команды @BotFather:

1. */start* – выдает список доступных команд.
2. */newbot* – команда, которая позволит создать нового бота, выбрать ему ник (name) и название аккаунта (username) — тут есть ограничение, название должно обязательно оканчиваться на слово Bot.

После успешного создания бота, получим <token> для дальнейшей реализации проекта.

В Visual Studio создается консольное приложение.

Устанавливаем библиотеку Telegram.Bot и подключаем:

using Telegram.Bot;

using Telegram.Bot.Args;

using Telegram.Bot.Types.Enums;

using Telegram.Bot.Types.InlineKeyboardButtons;

using Telegram.Bot.Types.ReplyMarkups;

using Telegram.Bot.Types;

Далее создается объект для работы с ботом:

static TelegramBotClient Bot;

Подключается ранее созданный бот:

Bot = new TelegramBotClient("токен");

Вызываются функции, которые заставляют бота слушать сообщения, а после выключиться:

Bot.StartReceiving();

Console.ReadLine();

Bot.StopReceiving();

Создается событие, чтобы получать сообщения от пользователя:

Bot.OnMessage += BotOnMessageReceived;

Далее реализуется функция, которую будет вызывать это событие:

private static async void BotOnMessageReceived(object sender, Telegram.Bot.Args.MessageEventArgs e)

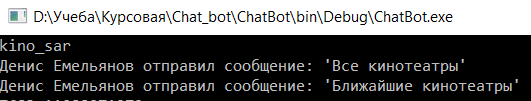
В этой функции создается:

* Переменная для записи сообщения пользователя:

var message = e.Message;

* Для удобства выводим в консоль сообщение от пользователя string name = $"{message.From.FirstName} {message.From.LastName}";: Console.WriteLine($"{name} отправил: '{message.Text}'");

В результате в консоли будет выводится так:



* Далее с помощью оператора Switch() будем обрабатывать сообщения пользователя. Создадим 5 меток:

case "/start":

var menu = new ReplyKeyboardMarkup(new[] {

new KeyboardButton("Геопозиция"){RequestLocation = true},

});

await Bot.SendTextMessageAsync(message.Chat.Id, "Отправьте геолокацию и перейдите в меню!", replyMarkup: menu);

break;

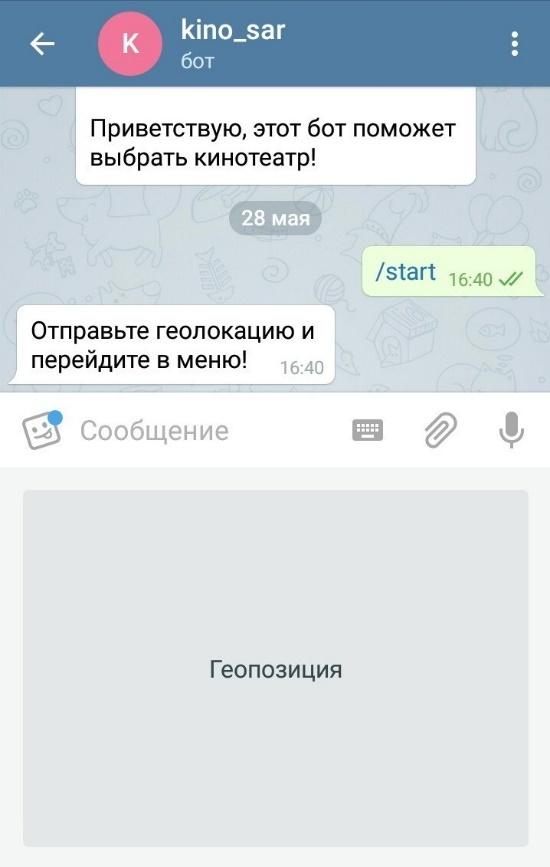
- главная метка, которая запрашивает геопозицию пользователя.

ReplyKeyboardMarkup – объект, который предоставляет клавиатуру.

KeyboardButton – кнопка, которая будет выводится у пользователя.

Bot.SendTextMessageAsync – иcпользуется для отправки ботом сообщение.

Как это будет выглядеть:



case "Назад":

case "Меню":

var replySpisok = new ReplyKeyboardMarkup(new[] {

new[]

{

new KeyboardButton("Все кинотеатры")

},

new[]

{

new KeyboardButton("Ближайшие кинотеатры")

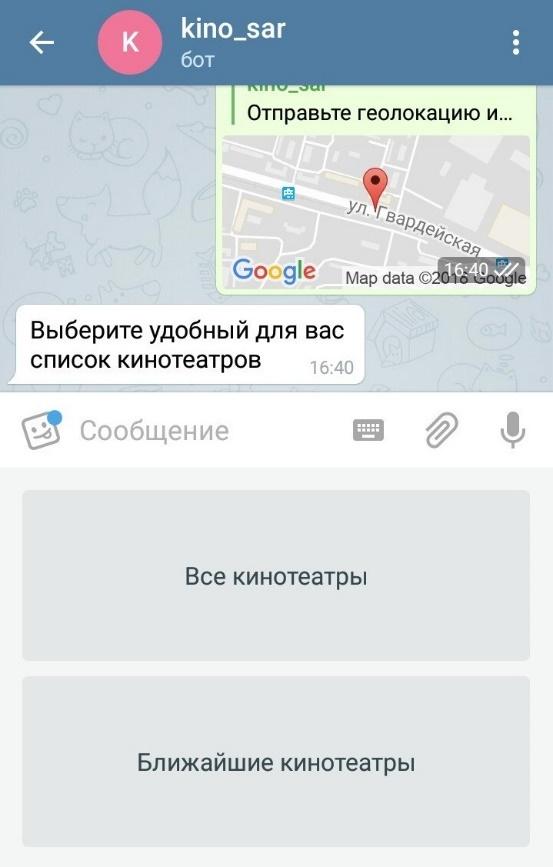
},

});

await Bot.SendTextMessageAsync(message.Chat.Id, "Выберите, то что вам нужно: ", replyMarkup: replySpisok);

break;

- метки, которые выводят две кнопки «Все кинотеатры» и «Ближайшие кинотеатры». При тестировании получим:



case "Все кинотеатры":

…

break;

Здесь мы уже должны будем подключиться к базе данных, затем вывести список кинотеатров.

Подключаем базу данных и считываем оттуда список кинотеатров в двумерный массив spicokCinema:

string connStr = "БД";

SqlConnection conn = new SqlConnection(connStr);

conn.Open();

string sql = "SELECT id, Name, Adress, Latitude, Longitude FROM Cinema2";

SqlCommand command = new SqlCommand(sql, conn);

SqlDataReader reader = command.ExecuteReader();

string[,] spisokCinema = new string[5, 5];

int i = 0;

while (reader.Read() && i != 5)

{

spisokCinema[i, 0] = reader[0].ToString();

spisokCinema[i, 1] = reader[1].ToString();

spisokCinema[i, 2] = reader[2].ToString();

spisokCinema[i, 3] = reader[3].ToString();

spisokCinema[i, 4] = reader[4].ToString();

i++;

}

reader.Close();

conn.Close();

В connStr записываем путь к БД. SqlConnection - подключает БД. В sql записываем нужный нам запрос. SqlCommand выполняет запрос. SqlDataReader - cсоздаём объект для считывания данных и записываем их.

Далее выводим кинотеатры в виде кнопок для этого будем использовать InlineKeyboardMarkup и InlineKeyboardButton:

InlineKeyboardMarkup – встроенная клавиатура, которая появляется под соответствующим сообщением.

InlineKeyboardButton – кнопка встроенной клавиатуры. Так же будем использовать метод WithCallbackData для того, что бы определить, какой кинотеатр выбрал пользователь.

var cinema = new InlineKeyboardMarkup

{

InlineKeyboard = new[]

{

new[]

{

InlineKeyboardButton.WithCallbackData("'"+spisokCinema[0,1] + "' " + spisokCinema[0,2], "callback1"),

},

…

}

};

await Bot.SendTextMessageAsync(message.Chat.Id, "Выберите кинотеатр: ", replyMarkup: cinema);

Создаем кнопки в диалоговом окне и выводим их.

После создадим обротку кнопок, которая будет определять фильмы какого кинотеатра нужно выводить. Для этого подключаемся к БД фильмов и выводим нужный список:

Bot.OnCallbackQuery += async (object sc, Telegram.Bot.Args.CallbackQueryEventArgs ev) =>

{

conn.Open();

string sqlfilm = "SELECT id, name, genre, price, address FROM filmsTriumf";

SqlCommand command2 = new SqlCommand(sqlfilm, conn);

SqlDataReader reader2 = command2.ExecuteReader();

string[,] spisokFilms = new string[4, 5];

i = 0;

while (reader2.Read() && i != 4)

{

spisokFilms[i, 0] = reader2[0].ToString();

spisokFilms[i, 1] = reader2[1].ToString();

spisokFilms[i, 2] = reader2[2].ToString();

spisokFilms[i, 3] = reader2[3].ToString();

spisokFilms[i, 4] = reader2[4].ToString();

i++;

}

var cinema2 = new InlineKeyboardMarkup

{

InlineKeyboard = new[]

{

new[]

{

InlineKeyboardButton.WithUrl("'"+spisokFilms[0,1] + "' " + spisokFilms[0,2]+" " + spisokFilms[0,3] +"p",spisokFilms[0,4]),

},

…

}

При создании кнопки фильма, используем метод WithUrl для того, чтобы добавить ссылку на данный фильм.

В итоге получится:

case "Ближайшие кинотеатры":

string connStr2 = "БД";

SqlConnection conn2 = new SqlConnection(connStr2);

conn2.Open();

string sql2 = "SELECT id, Name, Adress, Latitude, Longitude FROM Cinema2";

SqlCommand command3 = new SqlCommand(sql2, conn2);

SqlDataReader reader3 = command3.ExecuteReader();

string[,] spisokCinema2 = new string[5, 5];

i = 0;

while (reader3.Read() || i != 5)

{

spisokCinema2[i, 0] = reader3[0].ToString();

spisokCinema2[i, 1] = reader3[1].ToString();

spisokCinema2[i, 2] = reader3[2].ToString();

spisokCinema2[i, 3] = reader3[3].ToString();

spisokCinema2[i, 4] = reader3[4].ToString();

i++;

}

reader3.Close();

conn2.Close();

Console.WriteLine(distance(lat,lon,double.Parse(spisokCinema2[0, 3]),double.Parse(spisokCinema2[0, 4])));

var kino1 = new InlineKeyboardMarkup

{

InlineKeyboard = new[]

{

new[]

{

InlineKeyboardButton.WithCallbackData("'"+spisokCinema2[0,1] + "' " + spisokCinema2[0,2], "callback1"),

},

…

if (distance(lat, lon, double.Parse(spisokCinema2[1, 3]), double.Parse(spisokCinema2[1, 4])) < 5000)

{

Await Bot.SendTextMessageAsync(message.Chat.Id,distance(lat, lon, double.Parse(spisokCinema2[1,3]),"Расстояние: " + double.Parse(spisokCinema2[1, 4])).ToString() + "м", replyMarkup: kino2);

}

…

}

};

При выводе ближайших кинотеатров, подключаемся к базе данных и выводим те кинотеатры, которые ближе 5000 метров. Для вычисления дистанции используем функцию distance().

static public double distance(double lat, double lon, double lat2, double lon2)

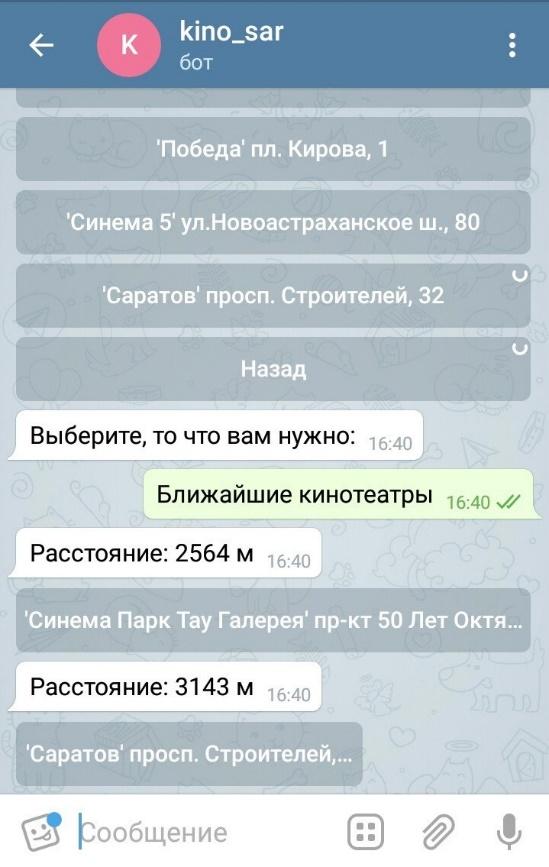
{

double k = 111.2 \* Math.Sqrt((lon - lon2) \* (lon - lon2) + (lat - lat2) \* Math.Cos(Math.PI \* lon / 180) \* (lat - lat2) \* Math.Cos(Math.PI \* lon / 180));

return k;

}

В итоге получим:



**Заключение**

В ходе данной работы было произведено ознакомление с библиотекой Telegram.Bot и раскрыты поставленные задачи. В теоретической части работы были изучены вопросы, связанные с чат-ботами, их виды и классификация, возможности и требования. Так же была проанализирована платформа Telegram.

В практической части мы рассмотрели реализацию чат-бота с помощью библиотеки Telegram.Bot, который позволяет пользователю просмотреть список фильмов в ближайших кинотеатрах, относительно их местоположения.

По ходу работы были сделаны следующие выводы:

* Чат-боты помогают оперативно получать нужную информацию, а благодаря простому интерфейсу легко указать то, что требуется пользователю.
* Благодаря популярности Telegram каждый может создавать чат-боты для своего бизнеса, которые будут увеличивать его функционал или распространять нужную информацию всем пользователям этой платформы, либо только коллегам по бизнесу.

В дальнейшем планируется автоматизация процесса добавления фильмов и кинотеатров, а также распространение по всем городам. Будет реализована бронирование билетов через бот и улучшена работа геопозиции.

# **Список литературы**

1. Почему чатботы захватывают мир +25 примеров для вдохновления [Электронный ресурс]. – URL: https://texterra.ru/blog/pochemu-chatboty-zakhvatyvayut-mir-25-primerov-dlya-vdokhnoveniya.html (Дата обращения 25.03.2018).
2. Боты в Telegram [Электронный ресурс]. – URL: https://iklife.ru/socialnye-seti/telegram/boty-v-telegram-chto-eto-takoe.html (Дата обращения 11.03.2018).
3. Статистика аудитории Telegram на январь 2018 [Электронный ресурс]. – URL: https://telegram-region.com/statistika-auditorii-telegram-na-yanvar-2018/ (Дата обращения 11.03.2018).
4. Andrew Stellman Head First C# 3-rd Edition / пер.с англ. Рузмайкина И. – СПб.: Питер, 2014. — 816 с.
5. Как настроить и запустить Microsoft SQL Server [Электронный ресурс]. – URL: http://www.internet-technologies.ru/articles/kak-nastroit-i-zapustit-microsoft-sql-server.html (Дата обращения 21.03.2018).
6. Telegram API Bot [Электронный ресурс]. – URL: http://telegramzy.ru/api-bot/ (Дата обращения 21.03.2018).
7. Разработка чат-бота для Telegram C# [Электронный ресурс]. – URL: https://geekbrains.ru/events/781 (Дата обращения 21.03.2018).
8. MySQL 5.0. Библиотека программиста Виктор Гольцман – СПб.: Питер, 2010. — 253 с.

**Приложения А.**

**using** **System**;

**using** **System.Collections.Generic**;

**using** **System.Linq**;

**using** **System.Text**;

**using** **System.Threading.Tasks**;

**using** **Telegram.Bot**;

**using** **Telegram.Bot.Args**;

**using** **Telegram.Bot.Types.Enums**;

**using** **Telegram.Bot.Types.InlineKeyboardButtons**;

**using** **Telegram.Bot.Types.ReplyMarkups**;

**using** **Telegram.Bot.Types**;

**using** **Telegram.Bot.Types.InlineQueryResults**;

**using** **System.Device.Location**;

**using** **System.Data.SqlClient**;

**using** **MySql.Data.MySqlClient**;

**namespace** **ChatBot**

{

**class** **Program**

{

**static** **public** **double** **distance**(**double** lat, **double** lon, **double** lat2, **double** lon2)

{

**double** k = **111.2** \* Math.Sqrt((lon - lon2) \* (lon - lon2) + (lat - lat2) \* Math.Cos(Math.PI \* lon / **180**) \* (lat - lat2) \* Math.Cos(Math.PI \* lon / **180**));

**return** k;

}

**static** TelegramBotClient Bot;

**static** **void** **Main**(**string**[] args)

{

Bot = **new** TelegramBotClient("558188651:AAGdNsT1xBhEthgaG8NMN\_2sPeqaqLhFUa4");

Bot.OnMessage += BotOnMessageReceived;

Bot.OnCallbackQuery += BotOnCallbackQueryReceived;

**var** me = Bot.GetMeAsync().Result;

Console.WriteLine(me.FirstName);

Bot.StartReceiving();

Console.ReadLine();

Bot.StopReceiving();

}

**private** **static** **void** **BotOnCallbackQueryReceived**(**object** sender, Telegram.Bot.Args.CallbackQueryEventArgs e)

{

**string** buttonText = e.CallbackQuery.Data;

**string** name = $"{e.CallbackQuery.From.FirstName} {e.CallbackQuery.From.LastName}";

Console.WriteLine($"{name} нажал кнопку {buttonText}");

}

**private** **static** **async** **void** **BotOnMessageReceived**(**object** sender, Telegram.Bot.Args.MessageEventArgs e)

{

**double** lat = **0**;

**double** lon = **0**;

**var** message = e.Message;

//if (message == null || message.Type != MessageType.TextMessage)

// return;

**string** name = $"{message.From.FirstName} {message.From.LastName}";

Console.WriteLine($"{name} отправил сообщение: '{message.Text}'");

Location geo = e.Message.Location;

**try**

{

Console.WriteLine(geo.Latitude);

Console.WriteLine(geo.Longitude);

}

**catch**

{

// Console.WriteLine("Нет данных!");

}

**switch** (message.Text)

{

**case** "/start":

**var** menu = **new** ReplyKeyboardMarkup(**new**[] {

**new** **KeyboardButton**("Геопозиция"){RequestLocation = **true**},

});

**await** Bot.SendTextMessageAsync(message.Chat.Id, "Отправьте геолокацию и перейдите в меню!", replyMarkup: menu);

**break**;

**case** "Назад":

**case** "Меню":

**var** replySpisok = **new** ReplyKeyboardMarkup(**new**[] {

**new**[]

{

**new** **KeyboardButton**("Все кинотеатры")

},

**new**[]

{

**new** **KeyboardButton**("Ближайшие кинотеатры")

},

});

**await** Bot.SendTextMessageAsync(message.Chat.Id, "Выберите, то что вам нужно: ", replyMarkup: replySpisok);

**break**;

**case** "Все кинотеатры":

**string** connStr = "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=ChatBotBD;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=True;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";

SqlConnection conn = **new** SqlConnection(connStr);

conn.Open();

**string** sql = "SELECT id, Name, Adress, Latitude, Longitude FROM Cinema2";

SqlCommand command = **new** SqlCommand(sql, conn);

SqlDataReader reader = command.ExecuteReader();

**string**[,] spisokCinema = **new** **string**[**5**, **5**];

**int** i = **0**;

**while** (reader.Read() && i != **5**)

{

spisokCinema[i, **0**] = reader[**0**].ToString();

spisokCinema[i, **1**] = reader[**1**].ToString();

spisokCinema[i, **2**] = reader[**2**].ToString();

spisokCinema[i, **3**] = reader[**3**].ToString();

spisokCinema[i, **4**] = reader[**4**].ToString();

i++;

}

reader.Close(); // закрываем reader

// закрываем соединение с БД

conn.Close();

**var** cinema = **new** InlineKeyboardMarkup

{

InlineKeyboard = **new**[]

{

**new**[]

{

InlineKeyboardButton.WithCallbackData("'"+spisokCinema[**0**,**1**] + "' " + spisokCinema[**0**,**2**], "callback1"),

},

**new**[]

{

InlineKeyboardButton.WithCallbackData("'"+spisokCinema[**1**,**1**] + "' " + spisokCinema[**1**,**2**], "callback2"),

},

**new**[]

{

InlineKeyboardButton.WithCallbackData("'"+spisokCinema[**2**,**1**] + "' " + spisokCinema[**2**,**2**], "callback3"),

},

**new**[]

{

InlineKeyboardButton.WithCallbackData("'"+spisokCinema[**3**,**1**] + "' " + spisokCinema[**3**,**2**], "callback4"),

},

**new**[]

{

InlineKeyboardButton.WithCallbackData("'"+spisokCinema[**4**,**1**] + "' " + spisokCinema[**4**,**2**], "callback5"),

},

**new**[]

{

InlineKeyboardButton.WithCallbackData("Назад", "callback5"),

},

}

};

**await** Bot.SendTextMessageAsync(message.Chat.Id, "Выберите кинотеатр: ", replyMarkup: cinema);

Bot.OnCallbackQuery += **async** (**object** sc, Telegram.Bot.Args.CallbackQueryEventArgs ev) =>

{

**var** message1 = ev.CallbackQuery.Message;

**if** (ev.CallbackQuery.Data == "callback5")

{

**var** replySpisok2 = **new** ReplyKeyboardMarkup(**new**[] {

**new**[]

{

**new** **KeyboardButton**("Все кинотеатры")

},

**new**[]

{

**new** **KeyboardButton**("Ближайшие кинотеатры")

},

});

**await** Bot.SendTextMessageAsync(message.Chat.Id, "Выберите, то что вам нужно: ", replyMarkup: replySpisok2);

}

**if** (ev.CallbackQuery.Data == "callback1")

{

conn.Open();

**string** sqlfilm = "SELECT id, name, genre, price, adress FROM filmsTriumf";

SqlCommand command2 = **new** SqlCommand(sqlfilm, conn);

SqlDataReader reader2 = command2.ExecuteReader();

**string**[,] spisokFilms = **new** **string**[**4**, **5**];

i = **0**;

**while** (reader2.Read() && i != **4**)

{

spisokFilms[i, **0**] = reader2[**0**].ToString();

spisokFilms[i, **1**] = reader2[**1**].ToString();

spisokFilms[i, **2**] = reader2[**2**].ToString();

spisokFilms[i, **3**] = reader2[**3**].ToString();

spisokFilms[i, **4**] = reader2[**4**].ToString();

i++;

}

**var** cinema2 = **new** InlineKeyboardMarkup

{

InlineKeyboard = **new**[]

{

**new**[]

{

InlineKeyboardButton.WithUrl("'"+spisokFilms[**0**,**1**] + "' " + spisokFilms[**0**,**2**]+" " + spisokFilms[**0**,**3**] +"p",spisokFilms[**0**,**4**]),

},

**new**[]

{

InlineKeyboardButton.WithUrl("'"+spisokFilms[**1**,**1**] + "' " + spisokFilms[**1**,**2**]+" " + spisokFilms[**1**,**3**] +"p",spisokFilms[**1**,**4**]),

},

**new**[]

{

InlineKeyboardButton.WithUrl("'"+spisokFilms[**2**,**1**] + "' " + spisokFilms[**2**,**2**]+" " + spisokFilms[**2**,**3**] +"p",spisokFilms[**2**,**4**]),

},

**new**[]

{

InlineKeyboardButton.WithUrl("'"+spisokFilms[**3**,**1**] + "' " + spisokFilms[**3**,**2**]+" " + spisokFilms[**3**,**3**] +"p",spisokFilms[**3**,**4**]),

},

}

};

**await** Bot.SendTextMessageAsync(message.Chat.Id, "Выберите фильм: ", replyMarkup: cinema2);

reader2.Close(); // закрываем reader

// закрываем соединение с БД

conn.Close();

}

};

**break**;

**case** "Ближайшие кинотеатры":

**string** connStr2 = "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=ChatBotBD;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=True;ApplicationIntent=ReadWrite;MultiSubnetFailover=False";

SqlConnection conn2 = **new** SqlConnection(connStr2);

conn2.Open();

**string** sql2 = "SELECT id, Name, Adress, Latitude, Longitude FROM Cinema2";

SqlCommand command3 = **new** SqlCommand(sql2, conn2);

SqlDataReader reader3 = command3.ExecuteReader();

**string**[,] spisokCinema2 = **new** **string**[**5**, **5**];

i = **0**;

**while** (reader3.Read() || i != **5**)

{

spisokCinema2[i, **0**] = reader3[**0**].ToString();

spisokCinema2[i, **1**] = reader3[**1**].ToString();

spisokCinema2[i, **2**] = reader3[**2**].ToString();

spisokCinema2[i, **3**] = reader3[**3**].ToString();

spisokCinema2[i, **4**] = reader3[**4**].ToString();

i++;

}

reader3.Close(); // закрываем reader

// закрываем соединение с БД

conn2.Close();

Console.WriteLine(distance(lat, lon, **double**.Parse(spisokCinema2[**0**, **3**]), **double**.Parse(spisokCinema2[**0**, **4**])));

**var** kino1 = **new** InlineKeyboardMarkup

{

InlineKeyboard = **new**[]

{

**new**[]

{

InlineKeyboardButton.WithCallbackData("'"+spisokCinema2[**0**,**1**] + "' " + spisokCinema2[**0**,**2**], "callback1"),

},

}

};

**var** kino2 = **new** InlineKeyboardMarkup

{

InlineKeyboard = **new**[]

{

**new**[]

{

InlineKeyboardButton.WithCallbackData("'"+spisokCinema2[**1**,**1**] + "' " + spisokCinema2[**1**,**2**], "callback1"),

},

}

};

**var** kino3 = **new** InlineKeyboardMarkup

{

InlineKeyboard = **new**[]

{

**new**[]

{

InlineKeyboardButton.WithCallbackData("'"+spisokCinema2[**2**,**1**] + "' " + spisokCinema2[**2**,**2**], "callback1"),

},

}

};

**var** kino4 = **new** InlineKeyboardMarkup

{

InlineKeyboard = **new**[]

{

**new**[]

{

InlineKeyboardButton.WithCallbackData("'"+spisokCinema2[**3**,**1**] + "' " + spisokCinema2[**3**,**2**], "callback1"),

},

}

};

**var** kino5 = **new** InlineKeyboardMarkup

{

InlineKeyboard = **new**[]

{

**new**[]

{

InlineKeyboardButton.WithCallbackData("'"+spisokCinema2[**4**,**1**] + "' " + spisokCinema2[**4**,**2**], "callback1"),

},

}

};

**if** (distance(lat, lon, **double**.Parse(spisokCinema2[**1**, **3**]), **double**.Parse(spisokCinema2[**1**, **4**])) < **5000**)

{

**await** Bot.SendTextMessageAsync(message.Chat.Id, "Расстояние: " + distance(lat, lon, **double**.Parse(spisokCinema2[**1**, **3**]), **double**.Parse(spisokCinema2[**1**, **4**])).ToString() + "м", replyMarkup: kino2);

}

**if** (distance(lat, lon, **double**.Parse(spisokCinema2[**4**, **3**]), **double**.Parse(spisokCinema2[**4**, **4**])) < **5000**)

{

**await** Bot.SendTextMessageAsync(message.Chat.Id, "Расстояние: " + distance(lat, lon, **double**.Parse(spisokCinema2[**4**, **3**]), **double**.Parse(spisokCinema2[**4**, **4**])).ToString() + "м", replyMarkup: kino5);

}

**break**;

**default**:

geo = e.Message.Location;

lat = e.Message.Location.Latitude;

lon = e.Message.Location.Longitude;

**var** CinemaMenu = **new** ReplyKeyboardMarkup(**new**[]

{

**new**[]

{

**new** **KeyboardButton**("Все кинотеатры")

},

**new**[]

{

**new** **KeyboardButton**("Ближайшие кинотеатры")

},

});

**await** Bot.SendTextMessageAsync(message.Chat.Id, "Выберите удобный для вас список кинотеатров", replyMarkup: CinemaMenu);

**break**;

}

}

}

}