

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**   
**«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ»**

Факультет Физико-технический

Кафедра Компьютерных технологий (КТ)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | КТ |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | Т. В. Ермоленко |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. | |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе бакалавра 3 курса

на тему:

|  |
| --- |
| Реализация вопросно-ответной системы «Помощник вязальщицы» на основе экспертных данных |
|  |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| Автор работы |  |  | Д.Д.Литвиненко |

подпись

|  |  |  |
| --- | --- | --- |
| Направление | 09.03.01 | Информатика и вычислительная техника |

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преподаватель Ю.В.Котенко

подпись

Консультанты по разделам:

Сетевые технологии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преподаватель А.Е. Гукай

подпись

Нормоконтроль \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. лаборант В.Г. Медведева

подпись

Курсовая работа защищена \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата итоговая оценка комиссия

Подписи членов комиссии: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Донецк  
2018

ГОУ ВПО «Донецкий национальный университет»

кафедра Компьютерных технологий

Утверждаю

Зав. кафедрой КТ

\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

**ЗАДАНИЕ**

на курсовую работу студента 3 курса Литвиненко Д.Д.

*Тема курсовой работы*: Реализация автоматической вопросно-ответной системы "Помощник вязальщицы» на основе экспертных данных

*Краткая постановка задачи*: 1. Изучить и проанализировать часто задаваемые вопросы и запросы поисковых систем на тему вязания. 2. Описать бизнес-процессы системы. 3. Ознакомиться с программными продуктами для создания telegram-бота, выполнить установку приложения для разработки Visual Studio и библиотеку Telegram.Bot. 4. Разработать техническое задание на создание telegram-бота для мессенджера telegram. 5. Разработать проект программного обеспечения системы: спроектировать концептуальную, логическую и физическую модели базы данных, создать базу данных для хранения информации об идеях для вязания, а также расчетных данных для калькулятора пряжи. 6. Разработать средствами языка программирования C# telegram-бота для мессенджера telegram, который позволяет пользователю узнать необходимое количество пряжи для конкретного изделия определенного размера, и подобрать идеи для вязания по запросу пользователя. 8. Протестировать программное обеспечение. 9. Оформить отчёт.

*Исходные данные*: 1. Документация популярных запросах на тему вязания. 2. Документация по Visual Studio 2015. 3. Документация по Telegram Api. 4. Документация по Dialogflow. 4. Документация по системе управления базами Access.

*Ожидаемые результаты*: вопросно-ответной системы "Помощник вязальщицы» на основе экспертных данных

*Календарный план работы:*

|  |  |  |
| --- | --- | --- |
| Даты консультаций | Этапы выполнения работы | Отметки о выполнении |
| 01.02.2018 | Постановка задачи и обсуждение литературы | выполнено |
| 08.02.2018 | Предварительное утверждение содержания отчёта | выполнено |
| 15.02.2018 | Утверждение проекта, алгоритмов, методов, технологий | выполнено |
| 04.03.2018 | Ход реализации проекта | выполнено |
| 18.03.2018 | Обсуждения организации тестирования программы | выполнено |
| 25.04.2018 | Демонстрация программного продукта руководителю | выполнено |
| 28.04.2018 | Оформление отчёта | выполнено |
| 03.05.2018 | Предоставление отчёта руководителю | выполнено |

Дата выдачи задания 01.02.2018 года

Студент Д.Д. Литвиненко

Руководитель Ю.В.Котенко

**АННОТАЦИЯ**

Отчёт о курсовой работе: 45 с., 21 рис., 2 табл., 2 приложения,   
11 источников.

Объект исследования – часто задаваемые вопросы на тему вязания.

Предмет исследования – система, построенная на основе экспертных данных и отвечающая на запросы пользователя.

Цель работы – разработать вопросно-ответную систему "Помощник вязальщицы" для Telegram на основе экспертных данных.

Метод исследования – анализ возможностей языка C#, пакета Telegram.Bot в Visual Studio, Telegram API, Dialogflow и мессенджер Telegram.

Разработанный telegram-бот предоставляет идеи для вязания, выполняет расчет пряжи для конкретного изделия определенного размера и вида нити, также может поддержать элементарный разговор с пользователем.

Для реализации проекта использовался язык программирования - C#. Использовались такие технологии, как Telegram API и искусственный интеллект Dialogflow. Разработка велась в Visual Studio.

Telegram-бот может использоваться обычным пользователем.

Дальнейшее развитие системы связано с улучшением существующего функционала и написанием дополнительного.

TELEGRAM-БОТ, С#, TELEGRAM API, СУБД, ACCESS, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, DIALOGFLOW, РАСЧЕТ ПРЯЖИ, ВЯЗАНЫЕ ИДЕИ, VISUAL STUDIO.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc513413321)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc513413322)

[1.1 Состояние вопроса 7](#_Toc513413323)

[1.2 Моделирование существующих бизнес-процессов 8](#_Toc513413324)

[1.3 Актуальность и цель работы 9](#_Toc513413325)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 12](#_Toc513413326)

[2.1 Описание области применения и исходных данных приложения 12](#_Toc513413327)

[2.2 Требования к пользовательским интерфейсам 12](#_Toc513413328)

[1.3 Требования к аппаратным и программным интерфейсам 12](#_Toc513413329)

[2.4 Требования к пользователям продукта 13](#_Toc513413330)

[2.5 Функции продукта 13](#_Toc513413331)

[2.6 Ограничения 13](#_Toc513413332)

[3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ 15](#_Toc513413333)

[3.1 Язык программирования C# 15](#_Toc513413334)

[3.2 СУБД Access 16](#_Toc513413335)

[3.3 Telegram API 17](#_Toc513413336)

[3.4 DialogFlow 17](#_Toc513413337)

[4 РАЗРАБОТКА TELEGRAM-БОТА 19](#_Toc513413338)

[4.1 Концептуальное проектирование 19](#_Toc513413339)

[4.2 Логическое проектирование 19](#_Toc513413340)

[4.3 Физическое проектирование 20](#_Toc513413341)

[4.3 Проектирование структуры приложения 20](#_Toc513413342)

[4.4. Описание классов и методов приложения 21](#_Toc513413343)

[5 РАЗРАБОТКА TELEGRAM-БОТА 24](#_Toc513413344)

[5.1 Структура telegram-бота 24](#_Toc513413345)

[5.2 Описание установки пакета для Visual Studio 24](#_Toc513413346)

[5.3 Описание алгоритма работы telegram-бота 25](#_Toc513413347)

[6 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 27](#_Toc513413348)

[6.1 Аппаратные, системные и программные требования 27](#_Toc513413349)

[6.2 Описание установки мессенджера telegram 27](#_Toc513413350)

[6.2.1 Запуск telegram-бота 29](#_Toc513413351)

[6.3 Описание контрольных примеров 29](#_Toc513413352)

[ЗАКЛЮЧЕНИЕ 31](#_Toc513413353)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 32](#_Toc513413354)

[ПРИЛОЖЕНИЕ А 33](#_Toc513413355)

[Экранные формы 33](#_Toc513413356)

[ПРИЛОЖЕНИЕ Б 38](#_Toc513413357)

[Фрагменты листинга 38](#_Toc513413358)

# ВВЕДЕНИЕ

Данная работа направлена на исследование предметной области вязания. Курсовой проект был разработан в среде Visual Studio 2015. Среди множества языков C# является универсальным и легко изучаемым языком. При этом его удобно использовать для создания сложных приложений, баз данных, приложений крупного масштаба. Visual Studio позволяет использовать все современные возможности объектно-ориентированного программирования.

В курсовой работе будет разработан telegram-бот для мессенджера telegram, который позволит рассчитать количество пряжи для конкретного изделия определенного размера и вида пряжи (и фурнитуры), подобрать для пользователя различные идеи для вязания с подробным описанием и наглядным материалом. Также в функции бота входит поддержание легкой беседы, ответы на тривиальные вопросы пользователя.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Состояние вопроса

В последние годы [вязание](https://www.rukodelie.ru/) во всем мире нашло широкое распространение, стало модным. Ручное вязание смело можно причислить к самым старинным видам рукоделия. Вязание, трикотажные изделия всё больше входят в нашу жизнь, причем не столько в виде привычных свитеров для тепла, сколько в виде интереснейших моделей из самой разнообразной пряжи, именно таких, которых не будет ни у кого, кроме вас. Только вязание сделает ваш гардероб по-настоящему оригинальным.

[Вязание крючком](http://www.rukodelie.ru/articles/vjazanije_krjuchkom_ot__a__do__ja_/) и [вязание спицами](http://www.rukodelie.ru/articles/vjazanije_spicami___ljubimoje_i_poleznoje_uvlechenije/) перестало быть чисто практическим ремеслом: чулки, шапки, кофты, а переросло в настоящее искусство. В наши дни это один из любимых видов рукоделия.

Вязание не нуждается в рекламе. Ручное вязание издавна присутствовало в одежде и в моде. Самые дорогие вещи — те, что связаны вручную. Вязание спицами — это «спицетерапия», для тех, кто любит мастерить, фантазировать. Выбрать пряжу — дело ответственное, ведь как поведет себя при первой стирке вещь, зависит от качества пряжи.

Ручное вязание всякого рода исполнялось всегда и везде. Оно по своей красоте и самобытности ничуть не уступает машинному трикотажу. Можно лишь догадываться о том, что вдохновило древнего человека взять в руки два тонких прутика и сделать первую в истории вязания петлю. К сожалению, древние вязаные образцы не дожили до наших дней. Нам в наследство остались лишь изображения, свидетельствующие о том, что вязание является одним из древнейших видов искусства и появилось оно примерно в III — IV н. э.

В наши дни вязание спицами или вязание крючком как вид рукоделия продолжает развиваться: активно внедряются новые материалы, разрабатываются узоры, из трикотажа изготавливаются всевозможные аксессуары; с недавних пор даже обувь. Вязаная одежда была и остается в моде, поражая воображение свободой форм, цвета и стиля. Но это еще не все. Удивительным образом за последние несколько лет вязание из традиционного рукоделия превратилось в модное увлечение.

Одновременно с бурным развитием вязания в геометрической прогрессии увеличивается количество молодых девушек, которые только начинают познавать искусство вязания, и у которых каждый день возникает уйма вопросов по этой теме. Именно поэтому такой актуальной стала идея компьютерной системы (или бота), которая может дать совет, помочь с идеями и расчетами пряжи.

Перед разработкой собственного telegram-бота, была выполнена работа по поиску ботов с подобной тематикой и функционалом, однако таких примеров выявлено не было.

## 1.2 Моделирование существующих бизнес-процессов

Одним из методов улучшения эффективности и качества работы фирмы является моделирование бизнес-процессов. С помощью бизнес-процессов можно понять работу и провести анализ организации.

Стандарт IDEF0 является наиболее широко используемой методологией описания бизнес-процессов. Модель состоит из фрагментов текстов и диаграмм. На такой диаграмме располагаются функциональный блок и интерфейсная дуга (входящая, выходящая, управления или механизма), в зависимости от стороны блока, к которой она подходит.

Ниже представлена контекстная диаграмма (рисунок 1.1). Ее задача – отображение в целом процессов, которые описывают структуру и функции проекта. На данной диаграмме реализованы следующие интерфейсы:

* управляющая информация – законы и нормативы, требования;
* входная информация – клиенты;
* результаты – идеи вязания, калькулятор пряжи;
* механизм –Telegram API.

Telegram-бот

Законы и нормативы, требования

Клиенты

Идеи вязания

Калькулятор пряжи

Telegram API

Рисунок 1.2 – Контекстная диаграмма

## 1.3 Актуальность и цель работы

Бот — это некий помощник, который общается с пользователем посредством текстовых сообщений. Чат-бот понимает, что вы пытаетесь сказать, и отвечает вам или даже сразу же выполняет требуемое от него действие.

Актуальными же их делает ряд вещей:

Огромное количество времени, которое люди проводят, набирая текст в различных мессенджерах (ВКонтакте, Facebook, Telegram и т.д) делает эти самые мессенджеры быстро растущим сегментом на рынке приложений. Люди используют их не только для дружеских переписок, но и в интересах развития бизнеса.

Быстрое развитие технологий искусственного интеллекта, машинного обучения позволяет современным чат-ботам вести себя максимально приближённо к тому, как ведут себя люди.

Современные боты не опираются исключительно на текст, а ещё и умеют показывать всевозможные полезные карточки, картинки, ссылки, формы, создавая атмосферу использования приложения.

Всё это позволяет использовать их для совершенно различных целей, таких как помощь в совершении покупок, сервис обратной связи, новости, игры и многое другое. Хороший бот не должен уметь делать всё: если он просто показывает вам последние новости, ему совершенно необязательно уметь заказывать еду из ресторана. Он выполняет что-то одно и выполняет на совесть.

Боты могут быть разработаны на любом языке программирования, с помощью которого можно создать web API. Для большинства это будут либо Node.js, либо PHP, однако существует множество библиотек для создания ботов, написанных на С# или Python.

Это может быть любой мессенджер: от популярных вроде Facebook Messenger, Telegram до простых Realtime Chat With Node.js. Также один и тот же бот может работать везде, не ограничиваясь какой-то опеделенной платформой.

В зависимости от того, как программировались конкретные боты, можно разделить их на две большие группы: работающие по заранее заготовленным командам и обучающиеся.

Работающие по командам боты работают, опираясь на заранее написанные ключевые слова, которые они понимают. Каждая из таких команд должна быть написана разработчиком отдельно с использованием регулярных выражений или других форм анализа строк. Если пользователь задал вопрос, не использую ни одного ключевого слова, робот не может понять его и, как правило, отвечает сообщениями вроде «простите, я не понял».

Хоть функционал подобных ботов ограничен, они могут быть очень эффективными в некоторых ситуациях. Например, когда пользователю нужно выбрать один-два пункта из нескольких.

Обучающиеся боты опираются на искусственный интеллект, чтобы общаться с пользователями. Вместо заранее подготовленных ответов, робот отвечает адекватными предложениями по теме. К тому же, все слова, сказанные боту, им же записываются для последующей обработки.

Работа с обработкой естественного языка и машинным обучением — непростое задание. Сегодня существует большой выбор отличных библиотек и сервисов (wit.ai, api.ai), которые могут помочь обучить ботов некоторым разговорным навыкам.

Целью курсовой работы является разработка telegram-бота для мессенджера telegram, который написан на языке программирования C# с помощью библиотеки Telegram Api, который позволит пользователю различные идеи для вязания и рассчитать количество пряжи для вязания определенного изделия.

# 2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

## 2.1 Описание области применения и исходных данных приложения

Данное приложение предназначено для обработки запросов пользователя на данную тему (вязание). Технологии приложения дают широкие возможности по поиску идей для будущих изделий или расчета материалов для настоящих проектов. Создание telegram-бота дает преимущества перед конкурентами, так как данный вид отображения информации широко не используется в нашем регионе.

Достоинства telegram-бота:

* простота в использовании;
* охват большой аудитории людей;
* высокая скорость обработки сообщений;
* информация о заведениях;
* отображение отзывов.

## 2.2 Требования к пользовательским интерфейсам

Пользовательский интерфейс должен быть удобен для неопытного пользователя. Все кнопки должны иметь интуитивный характер и корректное отображение. Необходимо, чтобы пользователь, запустив впервые telegram-бота понимал, куда необходимо перейти, чтобы получить желаемый результат. Telegram-бот должен быть максимально понятен, чтобы у пользователя не было сомнений по поводу его использования и сложностей в использовании. Необходимо наличие русского интерфейса. Также необходимым условием является наличие клавиатуры со списком команд, которые доступны боту.

## Требования к аппаратным и программным интерфейсам

Для работы telegram-бота необходимо наличие телефона, с поддержкой установки приложений, или десктопной версии приложения, а также систему следующей базовой конфигурации:

* оперативная память: не менее 2Гб;
* свободное место на телефоне: не менее 100мб;

Необходимо обеспечить программное взаимодействие системы с:

* подключением к сети интернет;
* мессенджером telegram.

## 2.4 Требования к пользователям продукта

Обычные пользователи должны иметь возможность пользоваться данной системой. Для комфортного использования требуются лишь базовые знания пользования телефоном или компьютером.

## 2.5 Функции продукта

Основные возможности telegram-бота:

* Простой и удобный интерфейс;
* Вывод идей для вязания;
* Хранение базы данных в отдельных файлах;
* Удобный калькулятор для расчета пряжи для определенного изделия конкретного размера и вида пряжи;

## 2.6 Ограничения

1. Telegram-бот не будет работать при выключенном интернете;
2. Для работы telegram-бота наличие мессенджера telegram обязательно;
3. Нельзя редактировать информацию о различных идеях для вязания;
4. Калькулятор может рассчитать количество пряжи только для имеющихся изделий и видов пряжи.
5. Telegram-бот воспринимает только заготовленные команды.

# 3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ

## 3.1 Язык программирования C#

Язык программирования C# (си-шарп) – язык объектного ориентирования, который разрабатывался с 1998 года специально для сети Интернет, а вернее, для платформы .NET. Он опирается на строгую компонентную архитектуру и обеспечивает надежные алгоритмы безопасности программного кода.   
Основной синтаксис и операторы перешли в C# из C. Объектная модель, реализованная в C++, в C# гораздо усовершенствована. Если, по отношению к C и C++, первая версия C# является прямым потомком.  
Отличием от предшественников является то, что в C# включено поддержание написания компонентов программ. Подобной поддержкой, например, выступают такие средства, как события, методы и свойства. Но главным ключом в компонентной ориентированности C# является, конечно же, обеспечение безопасной работы в среде многоязыкового программирования.  
В 2005 году вышел окончательный релиз версии 2.0 C#. Она укрепила позиции этого языка программирования и окончательно убедила в его эффективности и дальнейшем развитии. Добавление новых возможностей, таких как анонимные методы, обобщения, частичные и параметризованные типы значительно расширили возможности применения C#. В 2010 году была выпущена **текущая версия C# 4.0**. Ее главным дополнением к предыдущим версиям стали именованные и необязательные аргументы. Именованные дают возможность привязки аргумента и параметра по имени, а неименованные позволяют указать аргумент, который используется по умолчанию, для каждого параметра. Не менее важное новшество – **тип dynamic**. Он объявляет объекты, проверка на соответствие типов которых происходит не при компиляции, а непосредственно во время выполнения программы. [3]

## 3.2 СУБД Access

Под базой данных обычно понимают некоторое хранилище информации, включающее в себя множество однотипных элементов с различными свойствами, которые сгруппированы по определенным признакам. База данных может быть сформирована, например, из списка сотрудников, заказчиков, расчетных ведомостей, сведений о дорожных происшествиях в этом году и т.д. Применение базы данных этими целями не ограничивается – везде, где необходимо хранить и обрабатывать большие объемы информации, использование компьютерной базы данных поможет существенно облегчить эти задачи.

Программа Access является мощным средством для создания баз данных различного назначения и работы с ними; это средство просто и удобно в работе несмотря на то, что предоставляет огромное количество возможностей. С помощью данной программы можно хранить и систематизировать данные, что позволяет легко ориентироваться в них и быстро находить нужные сведения.

Также, используя Microsoft Access, несложно и обновлять данные, то есть приводить в соответствие с фактическим положением дел – достаточно изменить только один из параметров, и программа автоматически обновит все данные, которые его касаются. Если необходимо отобрать и просмотреть некоторые сведения о различных элементах базы данных, программа может создать сводную таблицу, которая будет содержать только необходимую информацию.

Программа Microsoft Access в своей работе использует так называемые “реляционные базы данных” - когда все разновидности обрабатываемой информации сгруппированы во взаимосвязанные таблицы. Работа с такой базой данных сводится к манипулированию таблицами – заполнению и изменению ячеек, объединению и внедрению элементов одних таблиц в другие.

Таблицы реляционной базы данных формируются на основе информации об объектах определенного типа. Каждая строка таблицы включает данные об одном объекте. Столбцы таблицы, в свою очередь, формируются на основе характеристик объектов, носящих название **атрибутов**. Строки таблицы носят название **записей**.

Записи должны иметь одинаковую структуру. Эта структура представляет собой поля, в которых и хранятся атрибуты. Каждое поле должно содержать одну характеристику объекта. Для поля определяется тип данных. При этом набор записей содержит одинаковые поля, но значения атрибутов в них различаются. [4]

## 3.3 Telegram API

Данный вид API позволяет подключать ботов к системе Telegram. Для работы не требуется установка дополнительного номера телефона. Учетные записи служат интерфейсом для кода, работающего на собственном сервере. Чтобы использовать Telegram API не нужно ничего знать о работе протокола шифрования MTProto, промежуточный сервер будет обрабатывать все шифрование и связь с API Telegram для работы. Общение с этим сервером происходит с помощью простого HTTPS-интерфейса, который предлагает упрощенную версию.

## 3.4 DialogFlow

Dialogflow (ранее Api.ai) — это продукты для коммуникаций, среди которых есть средства создания ботов для разных платформ. Например, для Google Assistant, Amazon Alexa, Facebook Messenger и других. Конечный продукт можно создать без навыков программирования. Доступна пошаговая форма с выбором функций бота, привязки контекста и обработки логики поведения.

Создание бота происходит в три этапа. Пользователь проектирует основные моменты, затем соединяет бота с веб-сервисом, и запускает бота в строй. Краткий список того, что предлагает Dialogflow:

**Взаимодействие с пользователем на естественном языке**. Предоставьте несколько примеров того, что пользователь может сказать, и Dialogflow создаст уникальную модель, которая сможет распознавать, какие действия запускать и какие данные извлекать из сказанного или написанного пользователем. Именно это и создаёт эффект разговора с обычным человеком.

**Создайте одно приложение и развёртывайте его везде**. Используйте сервис для создания диалогового приложения и развёртывайте его на своём веб-сайте, в мобильном приложении или на 32 различных платформах, включая Google Assistant и другие популярные службы обмена сообщениями. Он также поддерживает [множество языков](https://dialogflow.com/docs/reference/language), поэтому вы сможете охватить пользователей по всему миру.

**Расширенные варианты фулфилмента**. [Фулфилмент](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BB%D1%84%D0%B8%D0%BB%D0%BC%D0%B5%D0%BD%D1%82" \t "_blank) определяет соответствующие реакции на всё сказанное или написанное пользователем. Встроенный редактор кода позволяет вам координировать, тестировать и реализовывать эти действия непосредственно в консоли Dialogflow.

**Голосовое управление с распознаванием речи**. Dialogflow позволяет вашему разговорному приложению отвечать на голосовые команды пользователя. Данная функция доступна в рамках одного вызова API, сочетающего распознавание речи с пониманием естественного языка. [8]

# 4 РАЗРАБОТКА TELEGRAM-БОТА

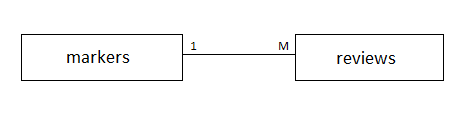
## 4.1 Концептуальное проектирование

Назначение концептуальной модели состоит в отражении информацию о предметной области таким образом, чтобы она было достаточно емкой для проработки проекта базы данных.

В процессе ознакомления с предметной областью выделяются следующие существенные объекты:

* Модель;
* Идеи.

Концептуальное представление базы данных приведено на рисунке 4.1.



Идеи

Модель

Рисунок 4.1 – Концептуальная модель базы данных

## 4.2 Логическое проектирование

Логическая модель предметной области иллюстрирует сущности, а также их взаимоотношения между собой.

На основе концептуальной модели была спроектирована логическая модель, показанная на рисунке 4.2

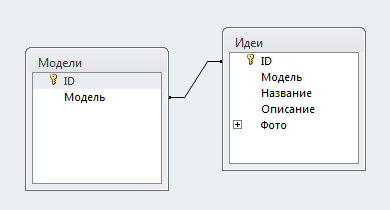


Рисунок 4.2 – Логическая модель базы данных

## 4.3 Физическое проектирование

Физические модели баз данных определяют способы размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне. СУБД создавала над этими файловыми моделями свою надстройку, которая позволяла организовать всю совокупность файлов таким образом, чтобы она работала как единое целое и получала централизованное управление от СУБД. Однако непосредственный доступ осуществлялся на уровне файловых команд, которые СУБД использовала при манипулировании всеми файлами, составляющими хранимые данные одной или нескольких баз данных. [5]

Физическое представление базы данных приведено в таблицах 4.1-4.2.

Таблица 4.1 — Таблица «Модели»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | | Тип | Комментарий |
| Id | PK | int(11) | Идентификатор модели |
| Модель |  | varchar(20) | Название |

Таблица 4.2 — Таблица «reviews»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | | Тип | Комментарий |
| Id | FK | int(11) | Идентификатор идеи |
| Модель |  | varchar(20) | Название модели |
| Название |  | varchar(20) | Название идеи |
| Описание |  | varchar(255) | Описание вязания |
| Фото |  | Int(1) | Фото изделия |

## 4.3 Проектирование структуры приложения

Проект KnitSencei\_bot состоит из файлов исходного кода:

1. Program.cs – главный класс программы. В нем подключаются все необходимые библиотеки, описываются основные методы получения сообщений от пользователя и отправки ответа;
2. CalcData.cs –хранение расчетных данных из csv-файла;
3. Dataofdata.cs –хранение структуры данных из текстового файла с расчетным материалом;
4. DataOfProduct.cs – хранение структуры данных из текстового файла с моделями идей;
5. DataOfIdea.cs – хранение идейных данных из csv-файла;
6. FillData–считывание данных из текстового файла расчетного материала;
7. FillProduct–считывание данных из текстового файла моделей идей;
8. Keyboard–описание основных команд, доступных боту с клавиатуры.

## 4.4. Описание классов и методов приложения

Главный класс приложения Program.cs содержит экземпляры классов и статические переменные:

1. static TelegramBotClient Bot – бот-клиент для работы с библиотекой Telegram;
2. static ApiAi apiAi – клиент для подключения искусственного интеллекта;
3. static FillIProduct fill\_product = new FillIProduct() – экземпляр класса заполнения листа данными из тектового файла;
4. static List<DataOfProduct> product = new List<DataOfProduct>() – лист с данными из файла;
5. static DataOfIdea idea = new DataOfIdea() – экземпляр класса генератора идей для вязания;
6. static CalcData cd = new CalcData() – экземпляр класса с расчетными данными;
7. static Keyboard keyboard = new Keyboard() – экземпляр класса с основными командами бота;
8. static List<DataOfIdea> bla = idea.ReadFile("ideas.csv") – лист, заполняющийся данными из файла;
9. static List<DataOfIdea> data = new List<DataOfIdea>() – лист, организующий запись идей;
10. static List<CalcData> bla2 = cd.ReadFile("data.csv") лист, заполняющийся данными с идеями из файла;
11. static List<CalcData> datas = new List<CalcData>() – лист, организующий запись расчетных данных;
12. Различные переменные для записи промежуточных и конечных результатов.
13. private static async void Bot\_OnCallbackQueryReceived(object sender, CallbackQueryEventArgs e) – обработчик события нажатия на кнопку;
14. private async static void Bot\_OnMessageReceived(object sender, MessageEventArgs e) – обработчик события получения сообщения.

Методы класса CalcData :

public void PieceData(string lin) – метод, разделяющий данные из csv-файла (разделитель – «;»);

public List<CalcData> ReadFile(string filename) – структура, заполняющаяся данными из из csv-файла;

Классы DataOfProduct и DataOfdata – классы, хранящие структуры текстовых файлов;

Методы класса FillData:

public void Fill\_Model(List<DataOfdata> dd) – метод, считывающий текстовый файл с моделями и добавляющих их в структуру лист;

public void Fill\_Size(List<DataOfdata> dd) - метод, считывающий текстовый файл с размерми и добавляющих их в структуру лист;

public void Fill\_Yarn(List<DataOfdata> dd) - метод, считывающий текстовый файл с метражом пряжи и добавляющих их в структуру лист.

Метод класса FillIProduct:

public void Fill\_product (List<DataOfProduct> idea) - метод, считывающий текстовый файл с идеями и добавляющих их в структуру лист.

Методы класса Keyboard:

public async void Start(TelegramBotClient Bot, MessageEventArgs e) – метод, выводящий стартовую фразу бота;

public async void Inline(TelegramBotClient Bot, MessageEventArgs e) – метод, определяющий работу клавиатуры в приложении;

public async void Keyboard\_k(TelegramBotClient Bot, MessageEventArgs e) – метод, определяющий работу выдвигающейся клавиатуры;

public int Idea(TelegramBotClient Bot, MessageEventArgs e) – обработчик вызова команды «Идеи»;

public int Calculator(TelegramBotClient Bot, MessageEventArgs e) – обработчик вызова команды «Калькулятор».

# 5 РАЗРАБОТКА TELEGRAM-БОТА

## 5.1 Структура telegram-бота

Структура telegram-бота приведена на рисунке 5.1.

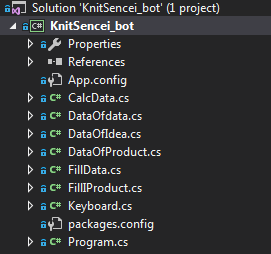


Рисунок 5.1 – Структура internet-приложения

## 5.2 Описание установки пакета для Visual Studio

Изначально необходимо установить пакет Telegram.Bot. Для этого в Visual Studio необходимо зайти в NuGet Manage и в поиске найти пакет Telegram.Bot (рисунок 5.2).

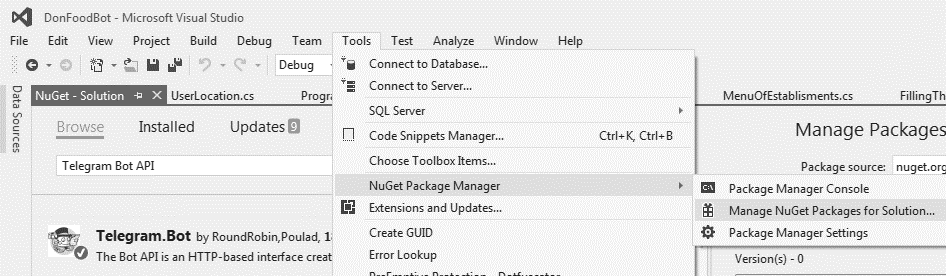


Рисунок 5.2 – Поиск пакета Telegram.Bot

## 5.3 Описание алгоритма работы telegram-бота

Боты – это специальные программы, выполняющие различные функции и упрощающие жизнь их пользователей. Написанные для платформы telegram, они предназначены для выполнения самых разных функций: от получения новостей до поиска информации и даже торговли акциями. Главное задачей бота является автоматический ответ после введенной ему пользователем команды. При этом, работая непосредственно через интерфейс telegram, программа имитирует действия живого юзера, за счет чего пользование таким ботом гораздо удобнее и понятнее.

Именно поэтому, многие компании, развивающие бизнес через интернет, используют возможности ботов по нескольким причинам:

1. Они позволяют задействовать очередной канал коммуникации с целевой аудиторией (в РФ Телеграмом пользуется около 10 миллионов человек)
2. Они быстро выполняют однообразную работу, позволяя разгрузить наемных сотрудников, тем самым экономя деньги компании.

Для начала написания функций telegram-бота необходимо написать боту @BotFather в мессенджере telegram. И следуя инструкциям необходимо:

* Придумать логин боту, которое бы заканчивалось на «bot» (изменить логин в дальнейшем нельзя);
* Добавить аватар боту и его описание;
* Получить уникальный токен от BotFather, с помощью которого и можно управлять ботом.

Данные действия дают возможность создать собственного бота для того, чтобы его обучить выполнять какие-либо функции необходимо написать код на языке программирования. [6]

Для того, чтобы получать данные из бота. Необходимо инициализировать Telegram API. Собственно инициализация API выглядит так: TelegramBotClient botClient = new TelegramBotClient(token);

Пакет Telegram.Bot поддерживает обработчики, с помощью которых можно научить бота отвечать на различные сообщения. Отправлять сообщения различного содержания (текст, музыка, фото, видео, документы).

# 6 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

## 6.1 Аппаратные, системные и программные требования

Для функционирования реализованного telegram-бота необходимы следующие аппаратные ресурсы:

* телефон, с поддержкой установки приложений;
* оперативная память: не менее 2Гб;
* свободное место на телефоне: не менее 100мб;
* непрерывное подключение к сети интернет;
* мессенджер telegram.

## 6.2 Описание установки мессенджера telegram

Telegram – программа, которая призвана заменить обмен привычными сообщениями. SMS, Hangouts, Viber, WhatsApp, может, даже Gmail. В данном мессенджере разрешено пересылать не только текст, можно и файлы разных форматов (картинки, музыку, видео, геопозицию, контакты, файлы). Приложение привязывается к номеру телефона, приходит проверочное сообщение с кодом (на мобильной версии для Android код подтягивается автоматически). Общаться можно как с конкретными пользователями, которые уже установили приложение, так и организовывать групповые чаты до 200 человек и отправлять пригласительные сообщения с призывом установить себе мессенджер.

Благодаря защищенности мессенджера и защите от перехвата приложение стало популярно в Гонконге во время протестов. Изначально о митинге договаривались через Facebook, но эту информацию легко заполучить.

Особенности telegram:

* Telegram является приватным и безопасным мессенджером. Сообщения являются зашифрованными и могут самостоятельно удаляться;
* Облачная структура. Информация хранится на удаленных серверах, поэтому доступ к сообщениям можно получить со всех устройств.
* Скорость. Сообщения отправляются за секунду;
* Открытость. Открытый код Telegram дает возможность любому желающему создать свою, улучшенную или дополненную версию.
* Бесплатность. Проект не продается и не покупается. Команда Дурова за безопасность общения и продвигает мессенджер в массы;
* Мощность. Telegram не имеет ограничений по количеству сообщений в день.

Для начала использования telegram-бота необходимо установить мессенджер Telegram из PlayМаркет (в случае ос android), AppStore (в случае ос iOS). Установка мессенджера отображена на рисунке 6.1

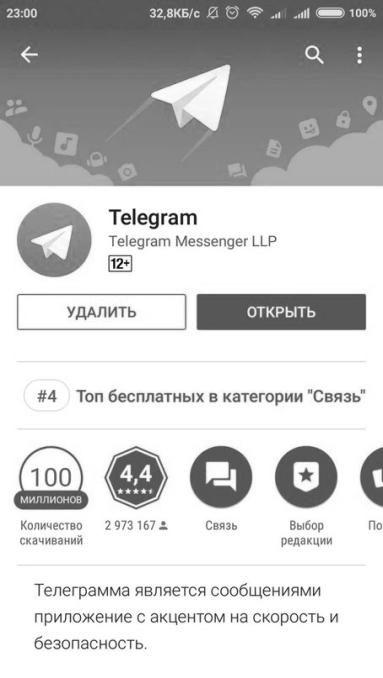


Рисунок 6.1 – Установка мессенджера Telegram для смартфона



Рисунок 6.2 – Установка мессенджера Telegram для компьютера

### 6.2.1 Запуск telegram-бота

Для того, чтобы воспользоваться telegram-ботом необходимо запустить мессенджер telegram и в поиске написать KnittingSencei. Поиск бота изображен на рисунке 6.2.

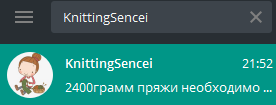


Рисунок 6.2 – Поиск бота

## 6.3 Описание контрольных примеров

После запуска приложения и поиска бота, отображается переписка с ботом.

При вводе команды «/start» отображается приветственное сообщение и список доступных боту команд (рисунок А1).

При вызове команды «/inline» отображаются кнопки с ссылками на социальные сети разработчика, перейдя по которым можно связаться с автором бота (рисунок А2).

При вызове команды «/keyboard» отображается скрытая клавиатура с командами. Это сделано для того, чтобы пользователь не сам вводил команды, а просто выбирал и нажимал на них в выезжающем меню (рисунок А3).

При вызове команды «/ideas» пользователь получает сообщение от бота с вопросом о том, идеи какого изделия его интересуют. Пользователь отвечает и получает одну идею из данной категории (рисунок А4).

При вводе слова «еще» пользователь получает больше идей из данной категории (рисунок А5).

При повторном вызове команды «/ideas» пользователь может выбрать другую категорию и получить идею из нее (рисунок А6).

Кроме того, при повторном вызове категории, если идеи закончились, бот говорит о том, что в данной категории больше идей нет (рисунок А7).

Также боту доступна беседа с пользователем посредством искусственного интеллекта. Бот может поддержать разговор, дать совет (рисунки А8-А10).

Бот распознает наборы букв, не являющиеся словами. И говорит о том, что он не понимает данной команды (рисунок А11).

# ЗАКЛЮЧЕНИЕ

Данный курсовой проект предназначен для оказания услуг посредством создания и внедрения telegram-бота. Приложение позволит обычному пользователю без затрат времени найти нужное заведение и получить информацию. Так как данные технологии широко не распространены в нашем регионе, это позволяет выйти telegram-боту на новый уровень и заполучить большое количество новых пользователей.

Во время анализа были рассмотрены различные СУБД, API, языки программирования, среды визуального проектирования.

Перед началом выполнения работы были изучены механизмы проектирования и механизмы разработки telegram-ботов. В ходе работы использовалась среда разработки Visual Studio, программирование велось на языке C#. Использовалcя Telegram API.

При разработке telegram-бота был реализован удобный и простой в использовании интерфейс для пользователя.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Telegram-бот Papas Bot.

URL: <https://www.papajohns.com.ru/> (дата обращения: 07.03.2018).

1. Telegram-бот BotoBot

URL: <https://www.botobot.ru/> (дата обращения: 07.03.2018)

1. История развития C# (си шарп) // Delphi-box

URL: http://delphi-box.ru/history-of-c.html (дата обращения: 10.03.2018)

1. СУБД Access. Общие сведения // «Электронный учебник»

URL: http://main.tpkelbook.com/ (дата обращения: 10.03.2018)

1. Физические модели баз данных // Национальный открытый университет «ИНТУИТ»

URL: https://www.intuit.ru/studies/courses/1001/297/lecture/7415 (дата обращения 15.03.2018)

1. Боты в Telegram и как они работают // Epicstars

URL: https://ru.epicstars.com/botyi-telegram/ (дата обращения: 15.03.2018)

1. Telegram

URL: https://core.telegram.org/ (дата обращения: 26.03.2018)

1. Dialogflow

URL: https://dialogflow.com/ (дата обращения: 30.03.2018)

1. Герберт Шилдт. C# 4.0: полное руководство / Г.Шилдт – М.: И.Д. Вильямс, 2011. — 1056 с
2. Актуальность интернет-ботов

URL: https://proglib.io/p/chat-bots-intro/ (дата обращения: 30.04.2018)

1. Ватсон Б. С# 4.0 на примерах / Бен Ватсон. – С.Пб.: БХВ Петербург, 2011. – 608 с.

# ПРИЛОЖЕНИЕ А

## Экранные формы

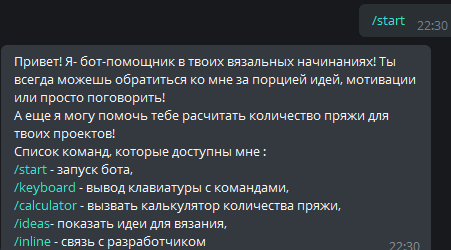
****

Рисунок А1 – Нажатие на кнопку «/start»

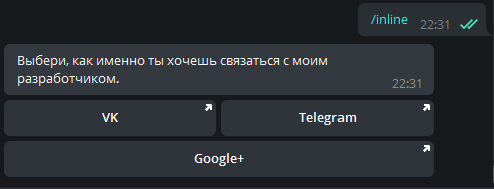


Рисунок А2 – Вызов команды для связи с разработчиков

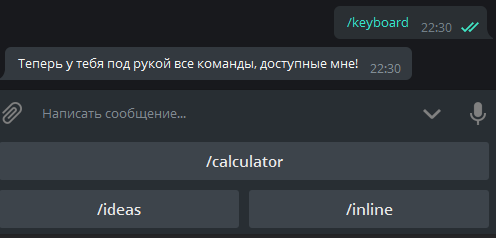


Рисунок А3 – Вызов команды для вывода клавиатуры

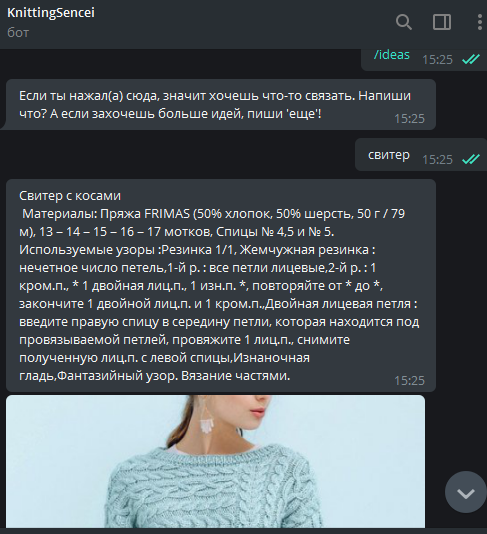


Рисунок А4 – Вызов генератора идей

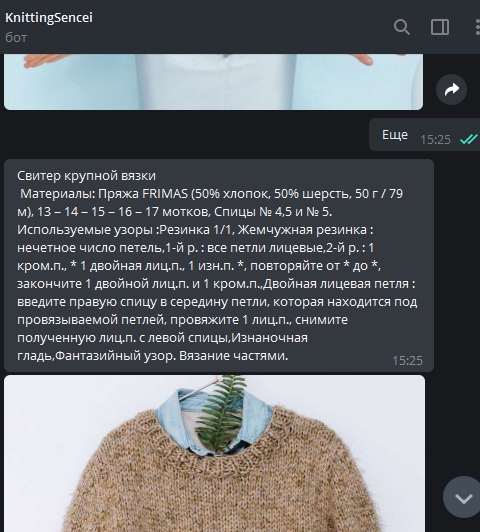


Рисунок А5 – Команда «Еще» в генераторе идей

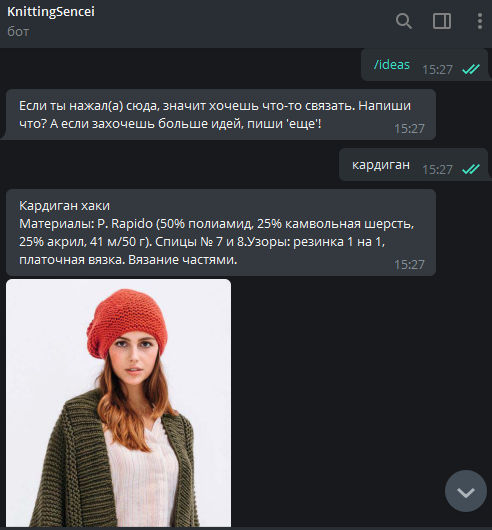


Рисунок А6 – Выход из генератора и повторный вызов

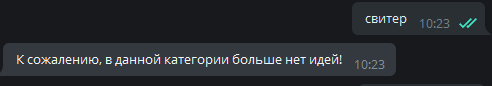


Рисунок А7 – Переполнение в генераторе идей

определенной модели

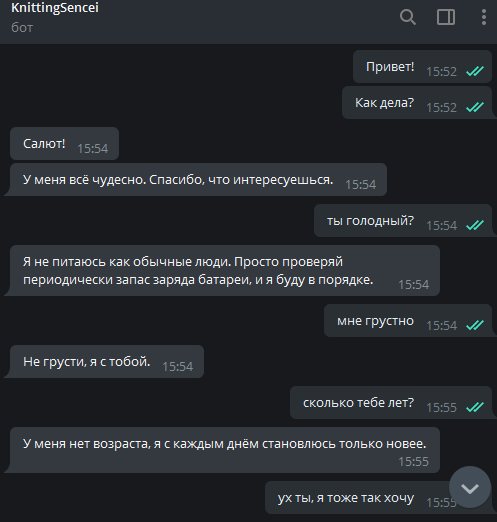


Рисунок А8 – Чат с ботом посредством ИИ(1)

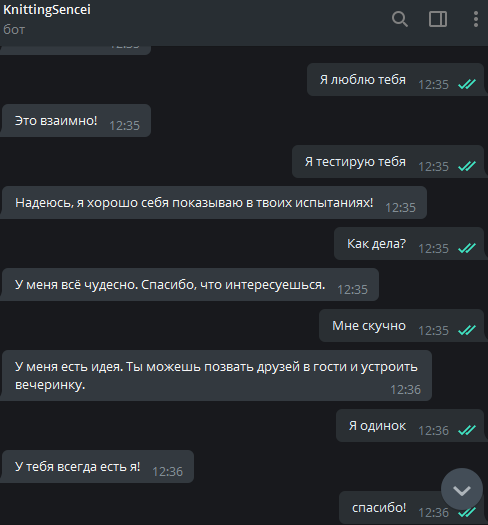


Рисунок А9 – Чат с ботом посредством ИИ(2)

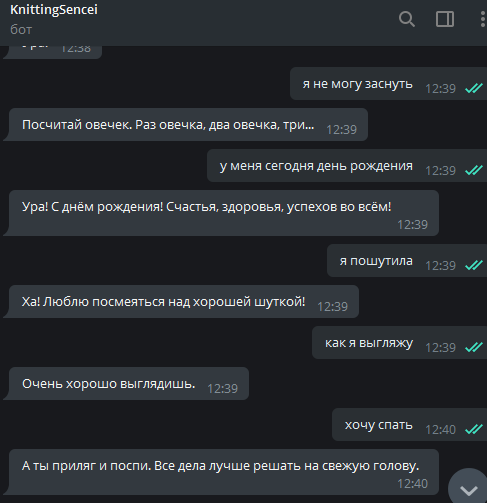


Рисунок А10 – Чат с ботом посредством ИИ(3)

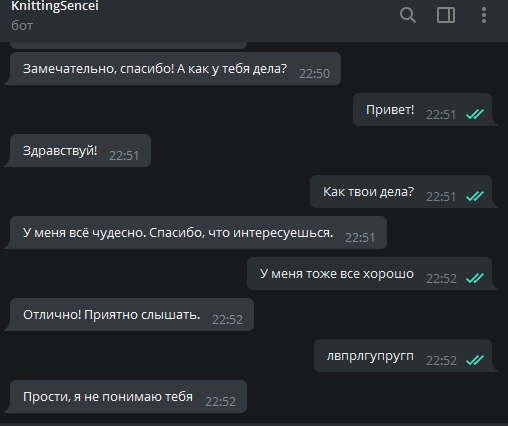


Рисунок А11 – Ошибка при вводе данных

# ПРИЛОЖЕНИЕ Б

## Фрагменты листинга

Листинг Б1 - главный класс Program.cs:

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 ApiAiSDK;

using ApiAiSDK.Model;

using System.IO;

namespace KnitSencei\_bot

{

class Program

{

static TelegramBotClient Bot;

static ApiAi apiAi;

static FillIProduct fill\_product = new FillIProduct();

static List<DataOfProduct> product = new List<DataOfProduct>();

static DataOfIdea idea = new DataOfIdea();

static CalcData cd = new CalcData();

static Keyboard keyboard = new Keyboard();

static List<DataOfIdea> bla = idea.ReadFile("ideas.csv");

static List<DataOfIdea> data = new List<DataOfIdea>();

static List<CalcData> bla2 = cd.ReadFile("data.csv");

static List<CalcData> datas = new List<CalcData>();

…

static void Main(string[] args)

{

//cee1d885245c490a85cf0ce0ef756cc4

Bot = new TelegramBotClient("596483216:AAFe1hLBlTZDZWNLLIxpzX2LUq-NhqHR2mo");

AIConfiguration config = new AIConfiguration("cee1d885245c490a85cf0ce0ef756cc4", SupportedLanguage.Russian);

apiAi = new ApiAi(config);

Bot.OnMessage += Bot\_OnMessageReceived;

Bot.OnCallbackQuery += Bot\_OnCallbackQueryReceived;

Bot.StartReceiving();

fill\_product.Fill\_product(product); //заполнение листа информацией из тхт

Console.ReadLine();

Bot.StopReceiving();

private static async void Bot\_OnCallbackQueryReceived(object sender, CallbackQueryEventArgs e)

{

string buttonText = e.CallbackQuery.Data;

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

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

await Bot.AnswerCallbackQueryAsync(e.CallbackQuery.Id, $"Ты нажал(а) кнопку {buttonText}");

}

private async static void Bot\_OnMessageReceived(object sender, MessageEventArgs e)

{

var message = e.Message;

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

await Bot.SendTextMessageAsync(e.Message.Chat.Id, "Извини, но я могу воспринимать только текст");

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

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

switch (message.Text)

{

case "/start": keyboard.Start(Bot, e); break;

case "/inline": keyboard.Inline(Bot, e); break;

case "/keyboard": keyboard.Keyboard\_k(Bot, e); break;

case "/ideas": last\_message = keyboard.Idea(Bot, e); break;

case "/calculator": last\_message\_data = keyboard.Calculator(Bot, e); break;

default:

Bot.SendTextMessageAsync(e.Message.From.Id, "Прости, я не понимаю тебя").Wait();

var response = apiAi.TextRequest(message.Text);

string answer = response.Result.Fulfillment.Speech;

if (answer == "")

answer = "Прости, я не понимаю тебя";

await Bot.SendTextMessageAsync(e.Message.From.Id, answer);

break;

}

for (int i = 0; i < bla2.Count(); i++)

{

if ((last\_message\_data + 1 == e.Message.MessageId) && (e.Message.Text.ToLower() == bla2[i].Model.ToLower()))

{

select\_model = bla2[i].Model;

result = Convert.ToInt32(bla2[i].KOmodel);

mod = bla2[i].Model + "," + bla2[i].Size + "," + bla2[i].Yarn ;

msg++;

}

}

if (msg == 1)

{

msg\_size = Bot.SendTextMessageAsync(e.Message.Chat.Id, "Укажите размер (XXS/XS/S/M/L/XL/XXL)").Result.MessageId;

msg++;

}

else if (msg == 2)

{

for (int i = 0; i < bla2.Count(); i++)

{

if ((msg\_size + 1 == e.Message.MessageId) && (e.Message.Text.ToLower() == bla2[i].Size.ToLower()))

{

select\_size = bla2[i].Size;

result \*= Convert.ToInt32(bla2[i].KOsize);

msg++;

}

}

}

if (msg == 3)

{

msg\_yarn = Bot.SendTextMessageAsync(e.Message.Chat.Id, "Укажи метраж нити (40-60 / 60-90 / 90-120 / 120-180 / 180-200 / 200-250 / 250-300 / 300-350 )").Result.MessageId;

msg = 0;

}

else if (msg == 0)

{

for (int i = 0; i < bla2.Count(); i++)

{

if ((msg\_yarn + 1 == e.Message.MessageId) && (e.Message.Text.ToLower() == bla2[i].Yarn.ToLower()))

{

select\_yarn = bla2[i].Yarn;

result \*= Convert.ToInt32(bla2[i].KOyarn);

Console.WriteLine(result);

}

}

}

if (select\_yarn != null)

{

string res = result.ToString() + " грамм пряжи необходимо для твоего изделия!";

msg = 3;

Bot.SendTextMessageAsync(e.Message.Chat.Id, res).Wait();

}

blaCount = bla.Count()-1;

for (int i = 0; i <= product.Count() - 1; i++)

{

if ((last\_message + 1 == e.Message.MessageId) && (e.Message.Text.ToLower() == product[i].product.ToLower()))

select\_product = product[i].product;

}

if (blaCount != 0)

{

if (select\_product == bla[count\_idea].Product.ToLower())

{

Bot.SendTextMessageAsync(e.Message.Chat.Id, bla[count\_idea].Name + "\n" + bla[0].Description).Wait();

var FileUrl = string.Format(@"images//{0}", bla[count\_idea].Photo);

var stream = new FileStream(FileUrl, FileMode.Open);

var fileToSend = new FileToSend(bla[count\_idea].Photo, stream);

await Bot.SendPhotoAsync(e.Message.Chat.Id, fileToSend);

count\_idea++;

}

for (int i = 0; i <= product.Count() - 1; i++)

{

if ((last\_message + 1 == e.Message.MessageId) && (e.Message.Text.ToLower() == product[i].product.ToLower()))

{ select\_product = product[i].product; }

}

if ((blaCount != 0) & (count\_idea == 0))

{

for (int i = 0; i <= blaCount; i++)

{

if (select\_product == bla[i].Product.ToLower())

{

DataOfIdea tmp = new DataOfIdea();

tmp.ID = bla[i].ID;

tmp.Name = bla[i].Name;

tmp.Product = bla[i].Product;

tmp.Photo = bla[i].Photo;

tmp.Description = bla[i].Description;

data.Add(tmp);

}

}

if (count\_idea == 5)

{ Bot.SendTextMessageAsync(e.Message.Chat.Id, "К сожалению, в данной категории больше нет идей! Для того, чтобы найти что-то другое нужно выбрать /ideas").Wait(); select\_product = ""; data.Clear(); count\_idea = 0; }

else if (data.Count() != 0)

{

Bot.SendTextMessageAsync(e.Message.Chat.Id, data[count\_idea].Name + "\n" + data[count\_idea].Description).Wait();

var FileUrl = string.Format(@"images//{0}", data[count\_idea].Photo);

var stream = new FileStream(FileUrl, FileMode.Open);

var fileToSend = new FileToSend(data[count\_idea].Photo, stream);

await Bot.SendPhotoAsync(e.Message.Chat.Id, fileToSend);

count\_idea++;

}

}

}

}

if ((last\_message + 1 == e.Message.MessageId) && (e.Message.Text.ToLower() != product[count].product.ToLower()))

// Bot.SendTextMessageAsync(e.Message.From.Id, "Otvali Vasya").Wait();

}

}

Листинг Б2 – Структура считываемых из csv – файла данных в классе CalcData.cs:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace KnitSencei\_bot

{

public class CalcData

{

public string KOmodel { get; set; }

public string Model { get; set; }

public string KOsize { get; set; }

public string Size { get; set; }

public string KOyarn { get; set; }

public string Yarn { get; set; }

public void PieceData(string lin)

{

string[] parts = lin.Split(';');

KOmodel = parts[0];

Model = parts[1];

KOsize = parts[2];

Size = parts[3];

KOyarn = parts[4];

Yarn = parts[5];

}

public List<CalcData> ReadFile(string filename)

{

List<CalcData> result = new List<CalcData>();

using (StreamReader sdata = new StreamReader(filename))

{

string lin;

while ((lin = sdata.ReadLine()) != null)

{

CalcData calcul = new CalcData();

calcul.PieceData(lin);

result.Add(calcul);

}

}

return result;

}

}

}

Листинг Б3 – Клавиатура и обработка команд в классе Keyboard.cs:

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;

using Telegram.Bot.Types.InlineKeyboardButtons;

using Telegram.Bot.Types.ReplyMarkups;

namespace KnitSencei\_bot

{

class Keyboard

{

public async void Start(TelegramBotClient Bot, MessageEventArgs e)

{

string text =

@" Привет! Я-бот-помощник в твоих вязальных начинаниях! Ты всегда можешь обратиться ко мне за порцией идей, мотивации или просто поговорить!

А еще я могу помочь тебе расчитать количество пряжи для твоих проектов!

Список команд, которые доступны мне :

/start - запуск бота,

/keyboard - вывод клавиатуры с командами,

/calculator - вызвать калькулятор количества пряжи,

/ideas- показать идеи для вязания,

/inline - связь с разработчиком";

await Bot.SendTextMessageAsync(e.Message.From.Id, text);

}

public async void Inline(TelegramBotClient Bot, MessageEventArgs e)

{

var inlineKeyboard = new InlineKeyboardMarkup

(new[]

{new[]

{

InlineKeyboardButton.WithUrl("VK", "https://vk.com/id10618217"),

InlineKeyboardButton.WithUrl("Telegram","https://t.me/litvinesha")

},

new []

{

InlineKeyboardButton.WithUrl("Google+", "https://plus.google.com/109645919916102660305"),

}

});

await Bot.SendTextMessageAsync(e.Message.From.Id, "Выбери, как именно ты хочешь связаться с моим разработчиком.", replyMarkup: inlineKeyboard);

}

public async void Keyboard\_k(TelegramBotClient Bot, MessageEventArgs e)

{

var replyKeyboard = new ReplyKeyboardMarkup(new[]

{

new []

{

new KeyboardButton("/calculator"),

},

new []

{

new KeyboardButton("/ideas"),

new KeyboardButton("/inline")

}

});

replyKeyboard.ResizeKeyboard = true;

await Bot.SendTextMessageAsync(e.Message.Chat.Id, "Теперь у тебя под рукой все команды, доступные мне!", replyMarkup: replyKeyboard);

}

public int Idea(TelegramBotClient Bot, MessageEventArgs e)

{

int lastmes = Bot.SendTextMessageAsync(e.Message.From.Id, "Если ты нажал(а) сюда, значит хочешь что-то связать. Напиши что? А если захочешь больше идей, пиши 'еще'! ").Result.MessageId;

return lastmes;

}

public int Calculator(TelegramBotClient Bot, MessageEventArgs e)

{

int lastmesdata = Bot.SendTextMessageAsync(e.Message.From.Id, "Укажи вид изделия, которое ты хочешь связать.").Result.MessageId;

return lastmesdata;

}