**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ ………………………………………………………………………..6](#_Toc168255314)

[1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧИ 7](#_Toc168255315)

[1.1 Описание предметной области 7](#_Toc168255316)

[1.2 Сравнительный анализ аналогов и прототипов 8](#_Toc168255317)

[1.3 Постановка задачи проектирования 10](#_Toc168255318)

[1.4 Анализ требования к проекту 10](#_Toc168255319)

[1.5 Выбор и обоснование средств и методов решения задач 11](#_Toc168255320)

[1.6 Разработка технического задания 13](#_Toc168255321)

[2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 14](#_Toc168255322)

[2.1 Функциональная структура приложения 14](#_Toc168255323)

[2.2 Проектирование диаграммы вариантов использования 14](#_Toc168255324)

[2.3 Информационное обеспечение проекта 15](#_Toc168255325)

[2.4 Разработка структуры конфигурации 15](#_Toc168255325)

[2.5 Разработка концепции пользовательского интерфейса 16](#_Toc168255327)

[2.6 Безопасность и защита данных 17](#_Toc168255328)

[3 РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ 18](#_Toc168255329)

[3.1 Результаты реализации серверной части 18](#_Toc168255330)

[3.2 Результаты реализации клиентской части 24](#_Toc168255330)

[3.3 Разработка программной документации 31](#_Toc168255331)

[3.4 Тестирование 32](#_Toc168255332)

[4 ЭКОНОМИЧЕСКАЯ ЧАСТЬ 18](#_Toc168255329)

[4.1 Обоснование необходимости выведения продукта на рынок 18](#_Toc168255330)

[4.2 Структура (этапы) работ по созданию программного обеспечения 24](#_Toc168255330)

[4.3 Составление сметы затрат на разработку программного обеспечения 31](#_Toc168255331)

[4.4 Расчет экономического эффекта разработчика ПО 32](#_Toc168255332)

[4.5 Вывод по экономической части 32](#_Toc168255332)

[ЗАКЛЮЧЕНИЕ 33](#_Toc168255341)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 34](#_Toc168255342)

[ПРИЛОЖЕНИЕ А (обязательное) Техническое задание 35](#_Toc168255343)

[ПРИЛОЖЕНИЕ Б (справочное) Диаграмма вариантов использования 39](#_Toc168255353)

[ПРИЛОЖЕНИЕ В (обязательное) Программа и методика испытаний 40](#_Toc168255355)

# ВВЕДЕНие

Сейчас уже трудно представить себе жизнь без интернета. В нем ежесекундно происходят миллионы событий начиная от простого поиска информации, общения и просмотра новостей заканчивая глобальными событиями по типу аукционов или введения государственных интернет услуг. В настоящее время трудно найти кампанию не имеющую свою группу в социальных сетях или собственный сайт. Использование социальных сетей и других технологий в интернете является удобным инструментом для всех организаций

Несмотря на удобство использования интернета, появилось множество интернет-мошенничества. Особенно прогрессирует мошенничество в социальных сетях. Мошенничество в социальных сетях – это онлайн-мошенничество, осуществляемое через социальные медиа-платформы, в которых мошенники размещают фальшивые рекламные акции, рассылают вредоносные ссылки или выдают себя за онлайн-аккаунт, чтобы украсть личную информацию пользователя[1].

Актуальность дипломной работы обусловлена высокой популярностью мессенджеров и таких средств автоматизации как чат-боты среди пользователей сети Интернет. Чат-боты позволяют упростить ежедневные рутинные задачи, такие как получение информации о погоде, пробках, последних новостях и другие. Главным достоинством относительно классических приложений является возможность совмещения всех возможностей на платформе одного мессенджера.

Данная работа посвящена созданию телеграмм бота, которые является администратором групп, и не дают мошенникам рассылать спам. В свою очередь все сообщения, которые бот удалит посчитав из спамом, будут сохранены в базе данных.

# АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧИ

## Описание предметной области

Свое развитие интернет-сервисы для общения начали с чатов, потом мессенджеры, потом социальные сети, но с недавних пор мессенджеры вновь возглавили список самых перспективных сервисов. Это подтверждено исследованиями компаний «We Are Social» и «Hootsuite» в 2021 году: количество пользователей мессенджеров в мире увеличилось на 13%. [2] Одновременно с этим среднее количество мессенджеров, используемых одним пользователем, увеличилось за 2021 год с 3 до 4 штук. Исследования компании «Билайн показали, что мессенджеры используются преимущественно взрослыми, так на долю лиц до 18 лет приходится 7,7% от общего количества аудитории, пользователи в возрасте от 18 до 25 лет составляют 9,8%, от 25 до 35 лет – 32,3%, от 35 до 45 лет – 2б,8%. При этом 13,1% пользователей мессенджеров находятся в возрасте от 45 до 55 лет, а 7,6% – от 55 до 64 лет. И всего 2,7% аудитории мессенджеров находится в возрасте старше 64 лет. [3]

Причина повторной волны популярности мессенджеров – самоизоляция которая заставила большее количество людей использовать мессенджеры для общения с близкими и коллегами, что сильно повлияло на рост трафика внутри приложений, за год он увеличился более чем в четыре раза. [3]

Telegram является приложением созданным на языке программирования С++. Клиентские приложения могут быть установлены на различные платформы: Windows, Linux, MacOS, IOS, Android так же существует веб- версия. Пользователи могут обмениваться сообщениями и различными файлами. Для работы приложение использует серверную часть с закрытым кодом, который расположен на серверах, которые находятся в разных концах планеты.

Главной особенностью Telegram считается специально разработанный протокол шифрования MTProto и возможность создания секретных чатов.

Секретные чаты требуются в тех случаях, когда посланное сообщение должно быть максимально защищено и даже быть удалено спустя некоторое время, например, при передачи каких-нибудь паролей или важных файлов содержащих интеллектуальную собственность. В этих чатах реализовано «Оконченное шифрование» которое гарантирует невозможность перехватки сообщений даже если вас взломают, ибо секретные чаты привязаны к самим устройствам. Сообщения из таких чатов не поддаются пересылке и не оставляют следов на сервере Telegram. Так же существуют некоторые особенности у этих чатов, а именно:

В том случае если ваш партнер или друг, с которым вы общались в секретном чате сменит режим чата на обычный то сообщения начнут шифроваться стандартным методом шифрования и это произойдет со всеми сообщениями в чате что были не удалены.

Если ваш собеседник сделает фото экрана, на котором будет секретный чат то уведомление о фото придет на ваше устройство.

Когда вы выйдете из аккаунта, но забудете закрыть секретный чат он удалится автоматически без возможности восстановления.

В данный момент Telegram может работать на многих устройствах: телефонах, компьютерах и даже доступен в веб версии. Так же он не обошел стороной и Linux так как изначальные пользователи были из сферы ИТ и больше пользовались Linux для своей работы. Сейчас Telegram переведен на многие языки: русский, французский, немецкий, итальянский, английский, польский и т. д.

Какими преимуществами обладает Telegram над другими мессенджерами:

* Имеет Porbale и веб-версию
* открытость – использование открытого протокола MТProto и API, бесплатных для всех;
* скорость отправки сообщений быстрее остальных мессенджеров;
* ограничение на размер отправляемых файлов до 2 ГБ.
* В Telegram отсутствует реклама;
* все сообщения будут зашифрованы, а сообщения из секретных чатов удалены без следов на сервере Telegram;

Чат-боты – это программа в основе, которой ИИ, которая способна принимать от пользователя различную информацию: сообщение, команды, файлы и т. п. после обработать их и выдать результат.

В основном выделяют чат-ботов двух типов:

1. Декларативные чат-боты, ориентированные на задания

Задействуют заранее прописанные инструкции и используются в основном для обслуживания и поддержки в формате 24\7.

1. Предиктивные чат-боты на основе данных, работающие в режиме диалога

Задействуют машинное обучение и являются более сложной формой декларативных чат-ботов. Они способны учитывать контекст отправленного пользователем сообщения и отправить более персонализированный ответ.

Чат-боты могут применятся для самых различных задач: простые ответы на вопросы по командам, различные тесты, отправка файлов, проверка файлов на вирусы, построение маршрутов по карте, вызов такси и т. п.

## Сравнительный анализ аналогов и прототипов

Сравним разрабатываемого бота DaysandboxBot с telemetrmebot и QuizBot.

DaysandboxBot предназначен для мониторинга новых участников группового чата в Telegram и предотвращения распространения нежелательного контента.

С момента добавления в чат, бот отслеживает события захода новых пользователей в группу и тем самым знает, когда кто зашёл в чат и сколько времени он уже провёл в чате. В силу ограничений telegram API я не могу узнать, когда к чату присоединились те пользователи, которые уже были на момент добавления бота в чат. Бот никак не анализирует сообщения от пользователей, существующи на момент добавления бота. Итак, бот знает, когда новые пользователи зашли в чатик. Если бот видит, пользователь провёл в чате меньше суток и запостил сообщение, удовлетворяющее определению спама, то бот удаляет такое сообщение. Он не банит пользователя, не ставит ему read-only права, бот просто удаляет сообщение со спамом. Если пользователь запостит не-спам сообщение, то оно будет опубликовано.

Спам-сообщения определяются по следующим критериям:

* содержат ссылку (URL, email, username);
* представляют собой пересланные (forward) сообщения;

Для правила [username](https://habr.com/ru/users/username/) есть исключение, если [username](https://habr.com/ru/users/username/) ссылается на пользователя, то такое сообщение разрешается. Бот удаляет только те сообщения с [username](https://habr.com/ru/users/username/), которые ссылаются на группу или канал. Это сделано для того, чтобы не было ложных срабатываний, когда новый пользователь просто пытается обратиться к какому-то участнику чата по его username[4].

Telemetrmebot служит инструментом анализа Telegram-каналов. Это простой бот-помощник сервиса Telemetr.me, который отображает динамику подписчиков и просмотров канала Телеграм. Ведет учет количества упоминаний канала и метрики ER. Ко всему дополнительно прилагается текстовая и графическая статистика.

Telemetrmebot имеет ряд преимуществ, таких как:

* простота при регистрации и в дальнейшем использовании;
* в формате изображения выводит все нужные метрики;
* ведет учет динамики и роста;
* подсчитывает упоминания непосредственно в Телеграм (а также и в YouTube, ВК и Инстаграм);
* при этом остается бесплатным.

Чтобы получить данные, необходимо:

* зайти в бот;
* нажать кнопку «Запустить»;
* в нужном формате отправить адрес канала.

После обработки запроса, бот отправит вам результат статистики[5].

QuizBot – это бот, который помогает быстро создавать тесты в Telegram. Алгоритм работы QuizBot:

* Откройте QuizBot в Telegram и нажмите «Начать».
* Создайте новый тест, указав его название и описание.
* Добавьте вопросы и варианты ответов. Для каждого вопроса выберите правильный вариант.
* Настройте время для прохождения теста.
* Готово! Вы можете отправить тест в любой чат или канал, где участники смогут его пройти.

Бот автоматически проверяет ответы и показывает результаты как вам, так и участникам, что экономит ваше время и силы, особенно если у вас много участников[6].

Как мы видим существует множество ботов в telegram, которые для улучшения качества групп. Эти три бота небольшое количество из всех возможных ботов. Но в наше время, очень много происходит мошенничества в интернете, и часто мошенники используют ссылка для кражи денег, для того что бы взламывать аккаунты и тд.

Выбранный бот проектируемый в рамках дипломной работы, является крайне необходимым в наше время, так как обеспечивает безопасность в telegram чатах, в отличии от других ботов, которые обеспечивают простоту управления чатами.

## Постановка задачи проектирования

Постановка задачи проектирования Daysandbox заключается в мониторинге всех участников вступивших в чат. На основании этой информации бот принимает решение об удалении сообщения. Так же данный дипломный проект включает в себя создании приложении по просмотру всех удалённых сообщений.

Основные требования к системе включают:

Реагирование на сообщения в реальном времени. При отправки сообщения пользователем бот должен сразу же реагировать на это сообщения. И в то же самое время, мгновенно сохранять эти сообщения в базе данных, для дальнейшего просмотра тех сообщений и от кого они были отправлены.

Пользовательский интерфейс. Данный пункт является необходимым для приложения по отображению удалённых сообщений.

Техническая поддержка и обновляемость. Приложения должны быть разработаны с учетом возможности регулярного обновления и модернизации.

## Анализ требования к проекту

При разработке telegram-bot Daysandbox необходимо решить следующие задачи:

* Определение логико-функциональной структуры бота;
* Проведение анализа и оптимизации основных программных модулей;
* разработка современного и интуитивно понятного интерфейса;
* интеграция с API Telegram;
* получении информации о пользователе;
* соединение с базой данных.

Информация о пользователе доступна при каждом пришедшем сообщении, если пользователь является администратором, бот не обрабатывает его сообщения

При обработке сообщений бот обращается в базу данных для получения информации о дате, когда пользователь был добавлен.

## Выбор и обоснование средств и методов решения задач

При разработке дипломного проекта происходил выбор языка программирования среди современных высокоуровневых языков.

Основными вариантами были следующие языки программирования:

* Python;
* C#.

Python представляет популярный высокоуровневый язык программирования, который предназначен для создания приложений различных типов. Это и веб-приложения, и игры, и настольные программы, и работа с базами данных. Довольно большое распространение питон получил в области машинного обучения и исследований искусственного интеллекта.

Впервые язык Python был анонсирован в 1991 году голландским разработчиком Гвидо Ван Россумом. С тех пор данный язык проделал большой путь развития. В 2000 году была издана версия 2.0, а в 2008 году - версия 3.0. Несмотря на вроде такие большие промежутки между версиями постоянно выходят подверсии. Так, текущей актуальной версией на момент написания данного материала является 3.12, которая вышла в октябре 2023 года.

Основные особенности языка программирования Python:

* Скриптовый язык. Код программ определяется в виде скриптов.
* Поддержка самых различных парадигм программирования, в том числе объектно-ориентированной и функциональной парадигм.
* Интерпретация программ. Для работы со скриптами необходим интерпретатор, который запускает и выполняет скрипт.

Выполнение программы на Python выглядит следующим образом. Сначала мы пишим в текстовом редакторе скрипт с набором выражений на данном языке программирования. Передаем этот скрипт на выполнение интерпретатору. Интерпретатор транслирует код в промежуточный байткод, а затем виртуальная машина переводит полученный байткод в набор инструкций, которые выполняются операционной системой.

Здесь стоит отметить, что хотя формально трансляция интерпретатором исходного кода в байткод и перевод байткода виртуальной машиной в набор машинных команд представляют два разных процесса, но фактически они объединены в самом интерпретаторе.

Python - очень простой язык программирования, он имеет лаконичный и в то же время довольно простой и понятный синтаксис. Соответственно его легко изучать, и собственно это одна из причин, по которой он является одним из самых популярных языков программирования именно для обучения. В частности, в 2014 году он был признан самым популярным языком программирования для обучения в США.

Python также популярен не только в сфере обучения, но в написании конкретных программ в том числе коммерческого характера. В немалой степени поэтому для этого языка написано множество библиотек, которые мы можем использовать.

Кроме того, у данного языка программирования очень большое сообщество программистов, в интернете можно найти по данному языку множество полезных материалов, примеров, получить квалифицированную помощь специалистов[7].

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей. [8]

Еще одним существенным преимуществом C# является его мощная библиотека классов, включающая множество предварительно написанных фрагментов кода и функций, которые можно использовать для быстрого создания сложных игровых систем. В C# также реализовано автоматическое управление памятью, что означает, что разработчикам не нужно беспокоиться о выделении и освобождении памяти вручную, поскольку сборщик мусора языка обрабатывает это автоматически.

Однако, пожалуй, наиболее значительным преимуществом C# для разработки игр является его кроссплатформенная совместимость. Код C# можно скомпилировать для работы на нескольких платформах, включая Windows, macOS, Linux, Android, iOS и даже веб-браузеры. Такая кроссплатформенная совместимость облегчает разработчикам создание игр для различных устройств и операционных систем, что может увеличить потенциальную аудиторию игры.

C# можно использовать для различных аспектов разработки игр, включая игровые движки, инструменты и сценарии. Многие популярные игровые движки, такие как Unity и Godot, используют C# в качестве основного языка сценариев. Например, в Unity C# используется для создания игровой логики, пользовательских интерфейсов и других игровых систем.

## Разработка технического задания

При разработке дипломного проекта были определены следующие требования

Требование к серверной части:

* Использование python 3.11, для обеспечении легкой поддержки.

Требования к клиентской части:

* Разработка клиентского приложения на основе WPF.
* Обеспечение взаимодействия с серверной частью через базу дынных PostgreSQL.

Технология WPF (Windows Presentation Foundation) является часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов.

Если при создании традиционных приложений на основе WinForms за отрисовку элементов управления и графики отвечали такие части ОС Windows, как User32 и GDI+, то приложения WPF основаны на DirectX. В этом состоит ключевая особенность рендеринга графики в WPF: используя WPF, значительная часть работы по отрисовке графики, как простейших кнопочек, так и сложных 3D-моделей, ложиться на графический процессор на видеокарте, что также позволяет воспользоваться аппаратным ускорением графики.

Первая версия - WPF 3.0 вышла вместе с .NET Framework 3.0 и операционной системой Windows Vista в 2006 году. И с тех пор платформа WPF является частью экосистемы .NET и развивается вместе с фреймворком .NET. Например, на сегодняшний день последней версией фреймворка .NET является .NET 8, и WPF полностью поддерживается этой версией фреймворка[9].

# Проектирование ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## Функциональная структура приложения

Приложение Daysandbox представляет собой Telegram-бота, основная задача которого – автоматическое удаление сообщений от новых пользователей в чатах и сохранение этих сообщений в базе данных. В данном разделе будет описана функциональная структура приложения, включая основные компоненты и их взаимодействие. Telegram API используется для взаимодействия с платформой Telegram и обрабатывает входящие сообщения, отправленные пользователями в чат. Основная логика приложения, реализованная на языке программирования, обрабатывает события, такие как получение новых сообщений и добавление пользователей в чат. База данных в приложении содержит таблицы для хранения информации о сообщениях и пользователях.

Перечислим основные функции приложения. Обработка новых сообщений, то есть бот анализирует все сообщения из чата проверяя, является ли отправитель новым пользователем. Удаление сообщений, то есть если сообщение отправлено новым пользователем, бот автоматически удаляет его из чата, а также удалённые сообщения передаются в базу данных для хранения. Сохранение сообщений в базу данных, то есть после удаления сообщение сохраняется в базе данных с соответствующей информацией. Кроме этого сделана возможность просмотра удалённых сообщений. Администратор может получить доступ к интерфейсу для просмотра всех удалённых сообщений.

Функциональная структура приложения Daysandbox позволяет эффективно управлять сообщениями от новых пользователей в Telegram-чате, обеспечивая автоматизацию процессов удаления и хранения информации о сообщениях. Это делает бот полезным инструментом для администраторов групповых чатов, стремящихся поддерживать порядок и чистоту в обсуждениях.

Данная функциональная структура позволяет охватить все основные аспекты работы telegram-bot.

## Проектирование диаграммы вариантов использования

Диаграмма вариантов использования для чат-бота будет отображать самого бота, взаимодействие пользователей с системой и функциональные задачи, которые бот должен выполнять. Актером выступает бот, находящийся в чате, вместе с пользователями. При этом диаграмма позволяет визуализировать основные процессы, происходящие в системе, и определить ключевые сценарии взаимодействия. Суть работы в том, что когда пользователь отправляет сообщение, то бот на него во всяком случае реагирует. Если пользователь находится больше суток в чате, то бот просто игнорирует сообщение. Иначе же удаляет сообщение, если это текстовое сообщение с ссылкой. Если пользователь меньше часа находится в чате, то бот будет удалять все сообщения, а именно текст, фото, видео.

## Информационное обеспечение проекта

Информационное обеспечение проекта включает использование современных технологий и инструментов, которые способствуют эффективной разработке, тестированию и поддержке информационной системы.

Для разработки клиентской части приложения использовалась технология WPF.

Процесс разработки серверной части проекта выполнялся в интегрированной среде разработки PyCharm Community 2023.

Процесс разработки клиентской части проекта выполнялся в интегрированной среде разработки Visual Studio 2022, которая предоставила мощные инструменты для написания кода, отладки и управления проектом. Visual Studio 2022 поддерживает интеграцию с системами контроля версий, что позволило эффективно управлять изменениями в коде.

PyCharm – это интегрированная среда разработки (IDE), созданная компанией JetBrains специально для языка программирования Python. IDE PyCharm включает набор инструментов и функций, которые облегчают разработку и отладку программного обеспечения на Python[10].

Интегрированная среда разработки Visual Studio является творческой стартовой площадкой, которую можно использовать для редактирования, отладки и сборки кода, а также для публикации приложения. В дополнение к стандартному редактору и отладчику, предоставляемых большинством интегрированных сред разработки, Visual Studio включает компиляторы, средства завершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки программного обеспечения [11].

Исходный код проекта будет сохраняться с помощью системы контроля версий git.

## Разработка структуры конфигурации

Для создания чат-бота, который будет автоматически удалять сообщения пользователей на основе заданных критериев, мы будем использовать различные технологии и инструменты, каждый из которых будет выполнять свою уникальную функцию.

На этапе проектирования функциональных требований и взаимодействий системы мы применим UML (Unified Modeling Language). Этот стандарт визуализации поможет создать диаграммы, которые отобразят архитектуру бота, его взаимодействие с пользователями, а также внутренние процессы обработки сообщений.

Серверная часть бота будет разрабатываться на языке Python. Выбор этого языка обусловлен его гибкостью, обширной поддержкой библиотек для работы с API и обработки данных, а также возможностью быстрого прототипирования. Python идеально подходит для разработки чат-ботов благодаря своей простоте и богатому экосистемному окружению.

Программирование и тестирование кода будут проводиться в интегрированной среде разработки PyCharm 2023. Эта IDE предоставляет мощные инструменты для написания и отладки кода на Python, а также поддерживает различные плагины, которые помогут ускорить процесс разработки и улучшить качество кода.

Для управления версиями и совместной работы над проектом мы будем использовать систему контроля версий Git. Она позволит отслеживать изменения в коде, упрощая совместную разработку, а также обеспечит безопасность данных, позволяя в любой момент откатиться к предыдущим версиям проекта при необходимости.

Таким образом, выбранный стек технологий и инструментов обеспечит создание эффективного и надежного чат-бота, который будет соответствовать всем требованиям по удалению сообщений пользователей.

## Разработка концепции пользовательского интерфейса

Разработка пользовательского интерфейса (UI) является одной из ключевых задач при создании любого программного обеспечения, так как именно от этого компонента зависит удобство и эффективность взаимодействия пользователя с системой.

При разработке проекта для отображения удалённых сообщений пользовательского интерфейса должен быть интуитивно понятным, минималистичным и при этом функциональным, чтобы пользователи могли с лёгкостью находить нужные функции и взаимодействовать с системой.

Согласно техническому заданию, программное обеспечение разрабатывается под Windows 10.

Для отображения всей информации будет использоваться таблица, которая является достаточно понятной для любого пользователя. Для просмотра определённого сообщения, будет создана отдельная кнопка в данной таблице.

## Безопасность и защита данных

## Для обеспечения безопасности и защиты данных при разработке чат-бота будет применен комплекс мер, направленных на предотвращение утечек информации и обеспечение надёжности работы системы. Основные аспекты безопасности включают:

## Шифрование данных: Весь обмен данными между пользователем и сервером будет защищён с использованием современных технологий шифрования, таких как HTTPS, что предотвратит перехват данных в процессе передачи.

## Обработка ошибок: В критически важных точках системы будут реализованы обработчики ошибок, которые обеспечат корректное поведение в случае возникновения исключений. Эти обработчики предоставят пользователю подробную информацию о проблемах, что поможет понять источник ошибки и улучшить взаимодействие с ботом.

* Логирование и мониторинг: Все действия, связанные с важными изменениями в системе (например, создание/изменение данных пользователя), будут логироваться. Это поможет оперативно выявлять возможные нарушения безопасности и принимать меры по их устранению.

‒ Обновления и патчи: Регулярные обновления программного обеспечения и своевременная установка патчей безопасности помогут предотвратить возможные уязвимости в системе.

Кроме того, защита данных будет включать соблюдение актуальных стандартов безопасности и рекомендаций, таких как GDPR, для обеспечения конфиденциальности пользовательских данных и соблюдения юридических требований.

# РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ

## Результаты реализации серверной части

Серверная часть приложения была разработана с использованием языка программирования python. Для обработки разных видов сообщений были разработаны различные модули, которые разделены по файлам.

Для обработки присоединения или удаления пользователя в чат был создан метод, для добавления и удаления пользователя в базу данных. Данный метод представлен в листинге 3.1.

Листинг 3.1– обработка присоединения или удаления пользователя в чат

async def greet\_chat\_members(update: Update, context: ContextTypes.DEFAULT\_TYPE) -> None:  
 result = extract\_status\_change(update.chat\_member)  
 if result is None:  
 return  
 was\_member, is\_member = result  
 cause\_name = update.chat\_member.from\_user.mention\_html()  
 member\_name = update.chat\_member.new\_chat\_member.user.mention\_html(  
 new\_user = update.chat\_member.new\_chat\_member.user  
  
 conn = psycopg2.connect(database="BotAdministrator",  
 host="localhost",  
 user="postgres",  
 password="postgres",  
 port="5432")  
 conn.autocommit = True  
 cursor = conn.cursor()  
 if not was\_member and is\_member:  
 await update.effective\_chat.send\_message(  
 f"{member\_name} was added by {cause\_name}. Welcome!",  
 parse\_mode=ParseMode.HTML,  
 )  
 now = datetime.now()  
 cursor.execute(sqlInsertDateJoined.format(new\_user.id, f"'{now}'"))  
 cursor.execute(sqlSelectIdUsers.format(new\_user.id))  
 fetchone = cursor.fetchone()  
 if fetchone is None:  
 username = new\_user.username if new\_user.username is not None else "null"  
 cursor.execute(sqlInsertUsers.format(  
 new\_user.id,  
 f"'{username}'",  
 f"'{new\_user.full\_name}'"  
 ))  
 elif was\_member and not is\_member:

Продолжение листинга 3.1

await update.effective\_chat.send\_message(  
 f"{member\_name} is no longer with us. Thanks a lot, {cause\_name} ...",  
 parse\_mode=ParseMode.HTML,  
 )  
 cursor.execute(sqlSelectDateJoined.format(new\_user.id))

cursor.execute(sqlDeleteDateJoined.format(new\_user.id))  
 conn.close()

Для того что бы выяснить, пользователь был добавлен или удален, был создан метод, который определяет добавлен пользователь или удален. Функция возвращает два булевых значение: был пользователь в чате и является ли сейчас он участником чата. Данный метод представлен в листинге 3.2.

Листинг 3.2– получение статуса пользователя

def extract\_status\_change(chat\_member\_update: ChatMemberUpdated) -> Optional[Tuple[bool, bool]]:  
  
 status\_change = chat\_member\_update.difference().get("status")  
 old\_is\_member, new\_is\_member = chat\_member\_update.difference().get("is\_member", (None, None))  
  
 if status\_change is None:  
 return None  
 old\_status, new\_status = status\_change  
 was\_member = old\_status in [  
 ChatMember.MEMBER,  
 ChatMember.OWNER,  
 ChatMember.ADMINISTRATOR,  
 ] or (old\_status == ChatMember.RESTRICTED and old\_is\_member is True)  
 is\_member = new\_status in [  
 ChatMember.MEMBER,  
 ChatMember.OWNER,  
 ChatMember.ADMINISTRATOR,  
 ] or (new\_status == ChatMember.RESTRICTED and new\_is\_member is True)

return was\_member, is\_member

Для обработки события, когда пользователь отправил сообщение в виде изображения, был создан метод, для сохранения локально изображения, добавления пути текущего изображения в базу данных и удаления сообщения из чата. Данный метод представлен в листинге 3.3.

Листинг 3.3 – удаление изображения

async def delete\_photo\_message(cursor, update: Update, reason\_for\_deletion: str):  
 photo = update.message.photo[-1]  
 file = await photo.get\_file()  
 file\_path = f"{name\_dir\_images}/{photo.file\_id}.jpg"

Продолжение листинга 3.3

await file.download\_to\_drive(file\_path)  
 cursor.execute(sqlInsertDeletedMessages  
 .format(update.message.from\_user.id,  
 f"'{datetime.now()}'",  
 "'PHOTO'",  
 f"'{file\_path}'",  
 f"'{reason\_for\_deletion}'"))  
 await update.message.delete()  
  
async def handle\_photo\_message(update: Update, context: CallbackContext):  
 conn = psycopg2.connect(database="BotAdministrator",  
 host="localhost",  
 user="postgres",  
 password="postgres",  
 port="5432")  
 conn.autocommit = True  
 cursor = conn.cursor()  
cursor.execute(sqlSelectDateJoined.format(update.message.from\_user.id))  
 datetime\_joined\_user = cursor.fetchone()[0]  
 time\_user\_is\_in\_the\_chat = datetime.now() - datetime\_joined\_user  
 if time\_user\_is\_in\_the\_chat.total\_seconds() < seconds\_in\_hour:  
 await delete\_photo\_message(cursor, update,

error\_message\_send\_photo\_less\_an\_hour)  
  
 conn.close()

Для обработки события, когда пользователь отправил видео, был создан метод, для сохранения видел локально на компьютер, добавления пути текущего видео-сообщения в базу данных и удаления самого видео-сообщения. Сообщения удаляется только после сохранения локально. Данный метод представлен в листинге 3.4.

Листинг 3.4 – удаления видео-сообщения

import psycopg2  
from telegram import Update, error  
from telegram.ext import (  
 CallbackContext,  
)  
from datetime import datetime  
from constants import \*  
import asyncio  
  
async def delete\_video\_message(cursor, update, reason\_for\_deletion):  
 video = update.message.video  
 file = await video.get\_file()  
 file\_path =

Продолжение листинга 3.4

f"{name\_dir\_videos}/{video.file\_id}.{video.mime\_type.split('/')[1]}"  
 await download\_file\_with\_retry(file, file\_path)  
 cursor.execute(sqlInsertDeletedMessages  
 .format(update.message.from\_user.id,  
 f"'{datetime.now()}'",  
 "'VIDEO'",  
 f"'{file\_path}'",  
 f"'{reason\_for\_deletion}'"))  
 await update.message.delete()  
  
async def handle\_video\_message(update: Update, context: CallbackContext):  
 conn = psycopg2.connect(database="BotAdministrator",  
 host="localhost",  
 user="postgres",  
 password="postgres",  
 port="5432")  
 conn.autocommit = True  
 cursor = conn.cursor()  
 cursor.execute(sqlSelectDateJoined.format(update.message.from\_user.id))  
 datetime\_joined\_user = cursor.fetchone()[0]  
 time\_user\_is\_in\_the\_chat = datetime.now() - datetime\_joined\_user

if time\_user\_is\_in\_the\_chat.total\_seconds() < seconds\_in\_hour:  
 await delete\_video\_message(cursor, update, error\_message\_send\_video\_less\_an\_hour)  
  
 conn.close()

Для сохранения видео локально был создан метод, который, при неудачном скачивании, пытается скачать видео снова. Данный метод представлен в листинге 3.5.

Листинг 3.5 – сохранение видео-сообщения

async def download\_file\_with\_retry(file, file\_path, retries=10, delay=5):  
 for attempt in range(retries):  
 try:  
 await file.download\_to\_drive(file\_path)  
 return   
 print("timeout")  
 if attempt < retries – 1:  
 await asyncio.sleep(delay)   
 else:  
 raise

Для обработки события, когда пользователь отправил текстовое сообщение, был создан метод, для добавления текущего сообщения в базу данных и удаления самого сообщения. При исследовании ботом текстового сообщения, проверяется содержит ли сообщение ссылку. Проверка осуществляется с помощью регулярных выражений. Данный метод представлен в листинге 3.6.

Листинг 3.6 – удаление текстового сообщения

async def delete\_text\_message(cursor, update: Update, reason\_for\_deletion: str):  
 cursor.execute(sqlInsertDeletedMessages  
 .format(update.message.from\_user.id,  
 f"'{datetime.now()}'",  
 f"'TEXT'",  
 f"'{update.message.text}'",  
 f"'{reason\_for\_deletion}'"))  
 await update.message.delete()  
  
async def handle\_text\_message(update: Update, context: CallbackContext):  
 chat\_admins = await update.effective\_chat.get\_administrators()  
 if update.effective\_user in (admin.user for admin in chat\_admins):

print("user is admin")  
 return  
 conn = psycopg2.connect(database="BotAdministrator",  
 host="localhost",  
 user="postgres",  
 password="postgres",  
 port="5432")  
 conn.autocommit = True  
 cursor = conn.cursor()  
 cursor.execute(sqlSelectDateJoined.format(update.message.from\_user.id))  
 datetime\_joined\_user = cursor.fetchone()[0]  
 time\_user\_is\_in\_the\_chat = datetime.now() - datetime\_joined\_user  
 if time\_user\_is\_in\_the\_chat.total\_seconds() < seconds\_in\_hour:  
 await delete\_text\_message(cursor, update, error\_message\_send\_message\_less\_an\_hour)  
 elif re.search("(?P<url>https?://[^\s]+)", update.message.text):  
 if time\_user\_is\_in\_the\_chat.total\_seconds() < seconds\_in\_day:  
 await delete\_text\_message(cursor, update, error\_message\_send\_message\_with\_reference)  
 conn.close()

Для работы всех, выше показанных, функций, был создан файл с различными константами: sql запросы, количество секунд в одном часе и в одних сутках, сообщения об ошибках и пути директорий для сохранения сообщений. Данные константы находятся в файле constants.py, который представлен в листинге 3.7.

Листинг 3.7 – constants.py

import os  
from dotenv import load\_dotenv, find\_dotenv  
config\_path = find\_dotenv('config.env')  
load\_dotenv(config\_path)  
  
sqlInsertDateJoined = os.getenv("sqlInsertDateJoined")  
sqlSelectDateJoined = os.getenv("sqlSelectDateJoined")  
sqlDeleteDateJoined = os.getenv("sqlDeleteDateJoined")  
  
sqlInsertDeletedMessages = os.getenv("sqlInsertDeletedMessages")  
sqlInsertUsers = os.getenv("sqlInsertUsers")  
sqlSelectIdUsers = os.getenv("sqlSelectIdUsers")  
  
seconds\_in\_hour = 60 \* 60  
seconds\_in\_day = seconds\_in\_hour \* 24  
  
error\_message\_send\_message\_less\_an\_hour = "the user has been in the chat for less than an hour"  
error\_message\_send\_message\_with\_reference = "the message contains a link"

error\_message\_send\_photo\_less\_an\_hour = "this photo was sent by a user, who has been in the chat for less than an hour"  
error\_message\_send\_video\_less\_an\_hour = "this video was sent by a user, who has been in the chat for less than an hour"  
name\_dir\_images = "../images"  
name\_dir\_videos = "../videos"

Для работы файла с константами, был создан файл environment, который содержал в себе запросы в базу данных, а также токен для бота. Данный файл представлен в листинге 3.8.

Листинг 3.8 – config.env

TOKEN = "8093292418:AAEPYVGW4B-qrCLD7ridWiKhZSryLZTaEKI"  
  
sqlInsertDateJoined = "INSERT INTO dateJoined(id, datetime)  
 VALUES({}, {})"  
sqlSelectDateJoined = "Select datetime FROM dateJoined  
 WHERE id = {}"  
sqlDeleteDateJoined = "Delete FROM dateJoined  
 WHERE id = {}"  
sqlInsertDeletedMessages = "INSERT INTO deletedMessages(id, user\_id, datetime\_of\_message\_sending, type\_message, message\_or\_path, reason\_for\_deleted)  
 VALUES(DEFAULT, {}, {}, {}, {}, {})"  
sqlInsertUsers = "INSERT INTO users(id, username, name)  
 VALUES({}, {}, {})"  
sqlSelectIdUsers = "Select id FROM users  
 WHERE id = {}"

## Результаты реализации клиенткой части

Клиентская часть приложения была разработана с помощью WPF с использованием языка программирования C#. Для работы с удаленными сообщениями в базе данных, EntityFramework Core автоматически сгенерировал все необходимые классы моделей и контекста. Было добавлен функционал для отображения необходимой информации на форме.

Сгенерированный класс Datejoined имеет поля: id пользователя и время добавления его в чат. Класс представлен в листинге 3.9.

Листинг 3.9 – Datejoined.cs

public partial class Datejoined

{

public int Id { get; set; }

public DateTime Datetime { get; set; }

}

Сгенерированный класс Deletedmessage имеет поля: id записи, id пользователя, username, который получаем с таблицы User по UserId, время отправки сообщения, тип сообщения, путь или текст, если это текстовое сообщение и причина удаления сообщения. Класс представлен в листинге 3.10.

Листинг 3.10 – Deletedmessage.cs

public partial class Deletedmessage

{

public int Id { get; set; }

[Browsable(false)]

public int UserId { get; set; }

public string Username

{

get

{

string name = BotAdministratorContext.getInstance().Users.Where(u => u.Id == UserId).Single().Name;

if (name == null)

return BotAdministratorContext.getInstance().Users.Where(u => u.Id == UserId).Single().Username;

return name;

}

}

public DateTime? DatetimeOfMessageSending { get; set; }

public string TypeMessage { get; set; } = null!;

Продолжение листинга 3.10

public string MessageOrPath { get; set; } = null!;

public string ReasonForDeleted { get; set; } = null!;

public virtual User User { get; set; } = null!;}

Сгенерированный класс User имеет поля: id пользователя, этот id telegram присваивает сам, псевдоним (никнейм) и имя пользователя. Класс представлен в листинге 3.11.

Листинг 3.11 – User.cs

public partial class User

{

public int Id { get; set; }

public string? Username { get; set; }

public string? Name { get; set; }

public virtual ICollection<Deletedmessage> Deletedmessages { get; set; } = new List<Deletedmessage>();

}

Сгенерированный класс BotAdministratorContext показан в листинге 3.12.

Листинг 3.12– BotAdministratorContext.cs

using System;

using System.Collections.Generic;

using Microsoft.EntityFrameworkCore;

namespace DisplayDeletedMessages.Models;

public partial class BotAdministratorContext : DbContext{

private BotAdministratorContext()

{

}

private static BotAdministratorContext \_context;

public static BotAdministratorContext getInstance()

{

if(\_context == null)

\_context = new BotAdministratorContext();

return \_context;

}

private BotAdministratorContext(DbContextOptions<BotAdministratorContext> options)

: base(options)

{

}

public static BotAdministratorContext getInstance(DbContextOptions<BotAdministratorContext> options)

Продолжение листинга 3.12

{

if (\_context == null)

\_context = new BotAdministratorContext(options);

return \_context;

}

public virtual DbSet<Datejoined> Datejoineds { get; set; }

public virtual DbSet<Deletedmessage> Deletedmessages { get; set; }

public virtual DbSet<User> Users { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

=> optionsBuilder.UseNpgsql("Host=localhost;Port=5432;Database=BotAdministrator;Username=postgres;Password=postgres");

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Datejoined>(entity =>

{

entity.HasKey(e => e.Id).HasName("datejoined\_pkey");

entity.ToTable("datejoined");

entity.Property(e => e.Id)

.ValueGeneratedNever()

.HasColumnName("id");

entity.Property(e => e.Datetime)

.HasColumnType("timestamp without time zone")

.HasColumnName("datetime");

});

modelBuilder.Entity<Deletedmessage>(entity =>

{

entity.HasKey(e => e.Id).HasName("deletedmessages\_pkey");

entity.ToTable("deletedmessages");

entity.Property(e => e.Id).HasColumnName("id");

entity.Property(e => e.DatetimeOfMessageSending)

.HasColumnType("timestamp without time zone")

.HasColumnName("datetime\_of\_message\_sending");

entity.Property(e => e.MessageOrPath)

.HasColumnType("character varying")

.HasColumnName("message\_or\_path");

entity.Property(e => e.ReasonForDeleted)

.HasMaxLength(120)

.HasColumnName("reason\_for\_deleted");

entity.Property(e => e.TypeMessage)

.HasMaxLength(5)

Продолжение листинга 3.12

.HasColumnName("type\_message");

entity.Property(e => e.UserId).HasColumnName("user\_id");

entity.HasOne(d => d.User).WithMany(p => p.Deletedmessages)

.HasForeignKey(d => d.UserId)

.OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("deletedmessages\_user\_id\_fkey");

});

modelBuilder.Entity<User>(entity =>

{

entity.HasKey(e => e.Id).HasName("users\_pkey");

entity.ToTable("users");

entity.Property(e => e.Id)

.ValueGeneratedNever()

.HasColumnName("id");

entity.Property(e => e.Name)

.HasMaxLength(70)

.HasColumnName("name");

entity.Property(e => e.Username)

.HasMaxLength(70)

.HasColumnName("username");

});

OnModelCreatingPartial(modelBuilder);

}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);

}

Для отображения данных с базы данных, была создана окно MainWindow. Разметка данного окна, в виде xaml, представлена в листинге 3.13.

Листинг 3.13 – MainWindow.xaml

<Window x:Class="DisplayDeletedMessages.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:DisplayDeletedMessages"

mc:Ignorable="d"

WindowState="Maximized"

Title="MainWindow" Height="450" Width="800" Loaded="Window\_Loaded">

<Grid>

<DataGrid x:Name="dateGridListMessages"

d:ItemsSource="{d:SampleData ItemCount=1}"

SelectionChanged="dateGridListMessages\_SelectionChanged"

AutoGenerateColumns="False">

Продолжение листинга 3.13

<DataGrid.Columns>

<DataGridTextColumn Header="Id" Binding="{Binding Id}"/>

<DataGridTextColumn Header="Username" Binding="{Binding Username}"/>

<DataGridTextColumn Header="DatetimeOfMessageSending" Binding="{Binding DatetimeOfMessageSending}"/>

<DataGridTextColumn Header="TypeMessage" Binding="{Binding TypeMessage}"/>

<DataGridTextColumn Header="ReasonForDeleted" Binding="{Binding ReasonForDeleted}"/>

<DataGridTemplateColumn Width="50">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Click="BtnShowContentInDataGridClicked">show</Button>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTextColumn Header="MessageOrPath" Binding="{Binding MessageOrPath}"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</Window>

В файле MainWindow.xaml.cs находятся вся необходимая логика для отображения удаленных сообщений. Данный файл представлен в листинге 3.14.

Листинг 3.14 – MainWindow.xaml.cs

using DisplayDeletedMessages.Models;

using System.Diagnostics;

using System.Windows;

using System.Windows.Controls;

using System.IO;

namespace DisplayDeletedMessages

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

BotAdministratorContext context = BotAdministratorContext.getInstance();

private Deletedmessage? \_deletedmessage = null;

private int \_countOfMessages;

private string \_addToPathInFront = "../../../../";

Продолжение листинга 3.14

public MainWindow()

{

InitializeComponent();

}

private void dateGridListMessages\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (dateGridListMessages.SelectedIndex < \_countOfMessages)

\_deletedmessage = (Deletedmessage)dateGridListMessages.SelectedItem;

else

\_deletedmessage = null;

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

List<Deletedmessage> list = context.Deletedmessages.ToList();

\_countOfMessages = list.Count;

dateGridListMessages.ItemsSource = list;

Console.WriteLine();

}

private void BtnShowContentInDataGridClicked(object sender, RoutedEventArgs e)

{

if (\_deletedmessage == null)

return;

User user = context.Users.Where(u => u.Id == \_deletedmessage.UserId).Single();

if(\_deletedmessage.TypeMessage == "TEXT")

{

MessageBox.Show(\_deletedmessage.MessageOrPath,

$"{(user.Name != null ? user.Name : user.Username)} прислал сообщение",

MessageBoxButton.OK,

MessageBoxImage.None);

}

if (\_deletedmessage.TypeMessage == "PHOTO")

{

string pathToPhoto = \_addToPathInFront + \_deletedmessage.MessageOrPath;

pathToPhoto = Path.GetFullPath(pathToPhoto);

var startInfo = new ProcessStartInfo

{

FileName = pathToPhoto,

UseShellExecute = true

};

Продолжение листинга 3.14

try

{

Process.Start(startInfo);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

if (\_deletedmessage.TypeMessage == "VIDEO")

{

string pathToVideo = \_addToPathInFront + \_deletedmessage.MessageOrPath;

pathToVideo = Path.GetFullPath(pathToVideo);

var startInfo = new ProcessStartInfo

{

FileName = pathToVideo,

UseShellExecute = true

};

try

{

Process.Start(startInfo);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

}

## Разработка программной документации

В результате создания программного продукта была разработана следующая программная документация:

* техническое задание (приложение А);
* описание программы (приложение Б);
* программа и методика испытаний (приложение В).

## Тестирование

Тестирование программного обеспечения (Software Testing) – проверка соответствия реальных и ожидаемых результатов поведения программы, проводимая на конечном наборе тестов, выбранном определённым образом [12].

Тестирование программного обеспечения является неотъемлемой частью процесса разработки, направленной на обеспечение качества и надежности создаваемых систем. В этой главе будут рассмотрены основные виды тестирования, методологии и инструменты, используемые для проверки правильности работы программного обеспечения, выявления дефектов и предотвращения возможных ошибок.

Существует несколько основных видов тестирования, которые различаются по своим целям и методам проведения. Модульное тестирование проводится на уровне отдельных компонентов или модулей программы и проверяет правильность работы каждого компонента в отдельности. Интеграционное тестирование проверяет взаимодействие между модулями и компонентами системы, чтобы убедиться в правильности передачи данных и совместной работе различных частей приложения. Системное тестирование осуществляется на уровне всей системы в целом и проверяет корректность работы всего программного продукта в реальных условиях эксплуатации. Приемочное тестирование выполняется с целью проверки соответствия программного обеспечения требованиям заказчика и готовности продукта к выпуску, часто включая участие конечных пользователей.

Существуют различные подходы к тестированию, которые позволяют проверять как внутренние структуры и логику кода, так и его внешнее поведение с точки зрения пользователя. В данной работе будут рассмотрены методы тестирования белого и черного ящиков, которые обеспечивают разный уровень видимости и охвата тестируемого кода.

Методы тестирования белого и черного ящиков представляют собой два подхода к проверке программного обеспечения. Белый ящик, также называемый стеклянным или прозрачным, предполагает, что тестировщик имеет доступ к исходному коду и внутренним структурам системы. Это позволяет проверять внутреннюю логику и структуру кода, обеспечивая высокий уровень покрытия. Среди техник белого ящика можно выделить анализ потока управления, который проверяет логические ветвления и циклы, и анализ потока данных, отслеживающий переменные и их значения во время выполнения. Также используется покрытие кода, чтобы убедиться, что все строки и условия протестированы, и тестирование пути, создающее тесты для различных логических маршрутов в коде.

# ЭКОНОМИЧЕСКАЯ ЧАСТЬ

## Обоснование необходимости выведения продукта на рынок

Цель экономического раздела – рассчитать затраты на разработку программного обеспечения и определить экономическую эффективность от его внедрения.

После проведенного в главе 1.2 сравнительного анализа telegram бот аналогов был сделан вывод о имеющемся ряде недостатков данных платформ. Данный анализ позволяет разрабатываемому продукту найти своих потребителей.

Реализованный продукт имеет достаточно широкое применение в связи с тем, что в нынешнее время появилось много «спамеров», которых нужно пересекать в чатах.

Telegram бот daysandbox предназначено для администраторов групп в telegram, разрабатываемому продукту, требуется подключение к интернету, в свою очередь, как пользователем, не обязательно подключение к интернету, по-скольку бот работает постоянно, не под надзором пользователя.

Таким образом, разрабатываемую интеллектуальную систему можно назвать актуальной для рынка «telegram-bot», так как данное программное обеспечение не только не теряет актуальности, но и набирает обороты на сегодняшний день.

## Структура (этапы) работ по созданию программного обеспечения

Жизненный цикл программного обеспечения (ПО) – период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации.

Для разработки программного продукта требуется время и структура этапов разработки, в которой будет выражаться количество недель, затраченных на его реализацию. Структура этапов и времени, выделенного для каждого этапа, необходимого для реализации Daysandbox бот представлена в таблице 4.1.

Таблица 4.1 – Таблица суммарного времени разработки программного обеспечения

| Этапы разработки | Затраты времени, недели | Процент от общего  времени, процент |
| --- | --- | --- |
| Проектирование | 2 недели | 16,67 |
| Реализация | 5 недель | 41,66 |

Продолжение таблицы 4.1

| Этапы разработки | Затраты времени, недели | Процент от общего  времени, процент |  |
| --- | --- | --- | --- |
| Тестирование | 3 недели | 25 |  |
| Доработка | 2 недели | 16,67 |  |
| Итого | 12 недель | 100 |  |

Процент от общего времени находим по следующей формуле:

, (4.1)

где Tp – процент от общего времени;

Wi – число недель одного этапа;

Ws – количество недель разработки.

*,*

Диаграмма выполнения работ по разработке программного обеспечения (рисунок 4.1).

Рисунок 4.1 – Диаграмма распределения времени работы при разработке программного обеспечения

Кроме деления жизненного цикла программного обеспечения на стадии и фазы, можно выделить восемь видов работ, которые выполняются в процессе создания программного продукта:

* анализ требований и планирование – предусматривает разработку спецификаций, анализ и модификацию функциональных, технических, интерфейсных требований;
* проектирование изделий – включает в себя определение, спецификацию, анализ и модификации аппаратно-программной архитектуры проекта программы и базы данных;
* программирование – детальное проектирование, кодирование, автономная отладка и комплексирование отдельных компонентов программы, а также планирование работ программистов, документирование отдельных компонентов и организация программирования;
* планирование отладки;
* верификация – процесс проверки корректности требований, отладки изделий и приемные испытания;
* управление проектом – планирование и контроль проекта, контроль и регулирование договоров, связь с пользователями;
* контроль качества – разработка и контроль, стандартные и технические проверки программных средств и процессов разработки;
* документирование – разработка и корректировка руководства для пользователя и операторов.

Деление жизненного цикла программных средств на стадии, фазы и этапы, а также более подробное описание работ, выполняемых в процессе их разработки, производства и эксплуатации, необходимо для детального определения затрат, требуемых на эти цели.

## Составление сметы затрат на разработку программного обеспечения

Расчет полной стоимости разработки и внедрения программного продукта учитывает следующие основные статьи затрат:

* материалы и комплектующие (*М*);
* электроэнергия (*Э*);
* основная заработная плата разработчиков (*Зо*);
* дополнительная заработная плата разработчиков (*Зд*);
* отчисления на социальные нужды (*Осн*);
* амортизация основных средств и нематериальных активов (*А*);
* расходы на спецоборудование (*Рс*);
* прочие прямые расходы (*Пз*).

1. Расходы по статье «Материалы и комплектующие» (*М*) отражают расходы на магнитные носители, бумагу и другие материалы, необходимые для разработки ПО.

Таблица 4.2 – Стоимость материалов, необходимых для разработки программного обеспечения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Ед. измерения | Количество | Цена(руб./шт.) | Сумма(руб.) |
| Бумага формата А4 | шт. | 500 | 0,04 | 18,86 |
| Компакт-диск | шт. | 1 | 2 | 2 |
| Всего | | | | 20,86 |

Общая стоимость основных и вспомогательных материалов *М* = 20,86 белорусских рубля.

2. Статья «Электроэнергия». Стоимость электроэнергии, необходимой для разработки программного обеспечения, представлена в таблице 4.3. Для расчета данного показателя необходим иметь в наличии следующие данные:

* продолжительность недель разработки;
* кол-во часов потребления электроэнергии за весь период разработки;
* стоимость одного кВт/ч электроэнергии (тариф).

Таблица 4.3 – Стоимость электроэнергии необходимой для разработки программного обеспечения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Статья расходов | Длительность периода, недели | Количество кВт/ч | Тариф за 1 кВт/ч. | Расход (бел. руб.) |
| Электроэнергия | 12 | 480 | 0,45659 | 219,16 |

Затраты электроэнергии формируются из затрат электроэнергии за один рабочий день, количество дней в неделю, количество недель разработки. Максимальная потребляемая энергия компьютера составляет 1 кВт/ч. При 8- часовом рабочем дне и 5-дневной неделе получаем в неделю 40 часов, за 12 недель разработки 480 часов.

(4.2)

где *Hw*– количество рабочего времени, затраченного на разработку;

*h –* продолжительность рабочего дня (часы);

*Wd –* количество рабочих дней;

*Ws* – сумма недель разработки.

(кВт/ч)

Базовый тариф для прочих потребителей с 01.01.2025 г. составляет 0,45659 руб. за 1 кВт/ч (согласно Приказа Министерства антимонопольного регулирования и торговли Республики Беларусь от 18.01.2018 №11).

Рассчитаем расходы на электроэнергию:

3. Статья «Основная заработная плата». Основной статьей расходов на создание ПО является заработная плата разработчиков (исполнителей) проекта, в число которых принято включать.

Таблица 4.4 – Расходы на специалистов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Специалисты | Количество (чел.) | Количество месяце в разработки | Основная зарплата (с учетом премии) за мес. (бел. руб.) | Основная с учетом дополнительной за месяц (бел. руб.) | Общая сумма заработной платы (бел. руб.) | Общая дополнительная (бел. руб.) |
| Инженеры програ-ммисты | 1 | 3 | 890,19 | 977,01 | 2670,57 | 2991,03 |
| Итого: | | | | | | 2991,03 |

С 1 января 2020 года в Беларуси появится новая система начисления зарплаты работникам бюджетной сферы. Правовое основание для этого – Указ Президента от 18 января 2019 г. № 27 «Об оплате труда работников бюджетных организаций». Для оплаты труда бюджетников применяется базовая ставка. С 01.01.2025 базовая ставка составляет 270 рублей.

Рассчитаем зарплату месячную тарифную ставку (оклад). Для программистов, работающих в бюджетной организации представлены коэффициенты в таблице 4.5.

Таблица 4.5– Тарифные ставки

|  |  |  |
| --- | --- | --- |
| Категории работников | Разряд | Коэффициент |
| Специалисты, занимающие должности, квалификационными требованиями которых предусматривается наличие высшего, среднего специального образования | 4 – 8 | 1.21 – 1.57 |

Месячная тарифная ставка каждого исполнителя (*Тм*) определяется путем умножения действующей базовой ставки ставки (*Тм1*) на тарифный коэффициент (*Тк*), соответствующий установленному тарифному разряду:

(4.3)

Основная заработная плата исполнителей рассчитывается по формуле:

(4.4)

где *n* – количество исполнителей, занятых разработкой конкретного ПО;

*ТМ* – месячная тарифная ставка (бел. руб.);

*К* – коэффициент премирования;

*t* – период времени, затраченный на разработку ПО.

Рассчитаем основную заработную плату исполнителя:

Основная заработная плата исполнителя составила 2670,57 белорусских рубля.

4. Статья «Дополнительная заработная плата» на конкретное ПО (Зд) включает выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей), и определяется по нормативу в процентах к основной заработной плате:

(4.5)

где *Здi* – дополнительная заработная плата исполнителей на конкретное ПО (руб.);

*Нд* – норматив дополнительной заработной платы (принимается в размере 7 - 15 %).

Дополнительная заработная плата на конкретное ПО составит:

5. Статья «Отчисления на социальные нужды». К данным затратам относят отчисления в фонд социальной защиты и отчисления на обязательное страхование от несчастных случаев (*Нстр* – 0,1 – 0,6%).

Отчисления на социальные нужды определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей, определенной по нормативу, установленному в целом по организации:

(4.6)

где *Нсз* – норматив отчислений в фонд социальной защиты населения (%);

*Нстр* – норматив отчислений на обязательное страхование от несчастных случаев (%).

Тариф отчислений в фонд социальной защиты населения установлен в размере 35%: 29% – пенсионное страхование и 6% – социальное страхование.

Страховой тариф составляет: для страхователей бюджетных организаций - 0,1 процента; для иных страхователей – 0,6 процента.

Исходя из выше перечисленного, отчисления на социальные нужды составят:

6. Статья «Амортизация основных средств и нематериальных активов». Определим стоимость основных средств (ноутбук) и нематериальных активов (программное обеспечение), на которые распространяются амортизационные отчисления (таблица 4.6).

Таблица 4.6 – Данные по амортизационной стоимости

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Активы | Стоимость актива, бел.р. | Период амор-тизации (лет) | За 1 год, бел. р. | За период  разработки (3 мес.) бел.р. |
| Gateway GWTN141-10BK | 1305,76 | 4 | 326,44 | 81,61 |
| Программное обеспечение | 1244,55 | 8 | 311,14 | 77,79 |
| Итого: | 159,4 | | | |

Амортизация – это процесс периодического переноса начальной стоимости основного средства или нематериального актива на производственные, коммерческие или общехозяйственные расходы – в зависимости от того, как этот актив используется [13].

При расчете амортизационных отчислений для начала необходимо определить стоимость основных средств и нематериальных активов. Под материальным активом в данном случае понимается ноутбук Gateway GWTN141-10BK. Под нематериальными активами – программное обеспечение: Window 10, Microsoft Office и среда разработки JetBrains PyCharm.

Расчет амортизационных отчислений проведен на основе линейного способа. Линейный способ заключается в том, что в течение всего срока полезного использование основное средство или нематериальный актив списываются равными долями. Амортизацию начисляют ежемесячно, начиная со следующего месяца после введения имущества в эксплуатацию, и до тех пор, пока не амортизируют полностью первоначальную стоимость основного средства или нематериального актива [13].

Для того чтобы рассчитать сумму амортизации линейным способом, можно использовать один из двух вариантов:

‒ исходя из годовой или месячной нормы амортизации в процентах (предусматривает умножение амортизируемой стоимости объекта на норму амортизации);

‒ без использования нормы амортизации (предусматривает распределение амортизируемой стоимости на срок амортизации).

В данном случае износ рассчитывается без использования нормы амортизации путем деления стоимости актива на количество лет его службы

формула (4.8).

(4.7)

где А – сумма амортизации в год;

t – период амортизации (лет);

– первоначальная стоимость.

Амортизационная стоимость за период разработки считалась по формуле 4.9:

(4.8)

где Аi – сумма амортизации за период разработки;

А – сумма амортизации в год;

ti – период амортизации разработки проекта (месяц).

Период амортизации разработки проекта составляет 3 месяца. Стоимость материального актива составила 1305,76 белорусских рулей, нематериального актива – 1244,55 белорусских рублей. Ниже представлены расчёты для каждог вида актива за 1 год.

После этого были произведены расчёты для периода разработки, который составил 3 месяца.

Итоговая амортизационная стоимость составила 159,4 белорусских рублей.

7. Статья «Расходы на спецоборудование» (*Pci*) включает затраты средств на приобретение вспомогательных технических и программных средств специального назначения, необходимых для разработки конкретного программного обеспечения, включая расходы на их проектирование, изготовление, отладку, установку и эксплуатацию. Затраты по этой статье определяются в соответствии со сметой расходов, которая составляется перед разработкой ПО.

Данная статья включается в смету расходов на разработку программного обеспечения в том случае, когда приобретаются специальное оборудование или специальные программы, предназначенные для разработки и создания только данного ПО.

Сумма начисленной амортизации за 1-й год составит:

|  |  |  |
| --- | --- | --- |
|  |  | (4.9) |
|  |  |  |

где *Цсi*– годовая норма амортизации;

*n –* остаточная или первоначальная стоимость актива.

Расходы на спецоборудование не понадобились.

8. Статья «Прочие прямые расходы» (*Пз*) на конкретное ПО включает затраты: на оплату услуг связи, Интернета, транспортные расходы, канцтовары, приобретение и подготовку специальной научно-технической информации и специальной литературы. Расходы определяются студентом самостоятельно по действующим на момент написания работы ценам и тарифам.

Тарифный план 4G не лимитируемого доступа в сеть Интернет («Безлимитище X» (скорость прием/передача до 112 Мбит/с)), в месяц 29,90 белорусских рублей.

Общая сумма расходов по смете (плановая себестоимость) (*С*) на ПО рассчитывается по формуле:

*С = М + Э + Зо + Зд + Осн + А + Рс + П3*(4.10)

Результаты расчетов представлены в таблице 4.7.

Таблица 4.7 −Расчет плановой себестоимости программного обеспечения

|  |  |  |
| --- | --- | --- |
| Статья затрат | Затраты, бел. руб. | Удельный вес, % |
| 1. Материалы и комплектующие (*М*); | 20,86 | 0,46 |
| 2. Электроэнергия; | 219,16 | 4,84 |
| 3. Основная заработная плата исполнителей (*Зо*) | 2670,57 | 58,95 |
| 4. Дополнительная заработная плата исполнителей (*Зд*); | 320,47 | 7,07 |
| 5. Отчисления на социальные нужды (*Осн*); | 1049,85 | 23,18 |
| 6. Амортизация (*А*); | 159,4 | 3,52 |
| 7. Расходы на спецоборудование (*Рс*); | - |  |
| 8. Прочие прямые расходы (*Пз*); | 89,9 | 1,98 |
| Общая сумма расходов по смете (*С*) | 4530,21 | 100 |

Представим затраты из таблицы 4.7 в графическом виде (рисунок 4.2).

Рисунок 4.2 − Структура затрат на разработку программного обеспечения

## Расчет экономического эффекта разработчика программного обеспечения

**4.4.1 Экономический эффект у разработчика программного обеспечения**

Рассчитаем отпускную цену программного обеспечения.

С учетом действующих в республике нормативных документов отпускная цена на продукцию предприятия рассчитывается по формуле:

*ОЦ = С + П,*  (4.11)

где *ОЦ* – отпускная цена изготовителя, бел. руб.;

*С* – плановая себестоимость, руб.;

*П* – прибыль руб.

4530,21 + 906,04 = 5436,25

Прибыль закладывается в цену по нормативу рентабельности (устанавливается самостоятельно), расчет производится по следующей формуле:

*П = R · C,* (4.12)

где *R* – норматив рентабельности (например, если рентабельность 20 %, то при расчете переводим в коэффициент и получаем 0,2).

*С* – плановая себестоимость, руб;

Отпускная цена изготовителя с налогом на добавленную стоимость:

Ставка налога на добавленную стоимость установлена в размере 20 %.

*НДС = (С + П) · ставка НДС(%) / 100 %*, (4.13)

где *С* – плановая себестоимость, руб.;

*П* – прибыль, руб.; ставка *НДС* = 20 %;

*(4530,21* *+* 906,04*) · 20 / 100 = 1087,25*

Прогнозируемая отпускная цена на программное обеспечение с *НДС*:

*ОЦ с НДС= С + П + НДС*, (4.14)

*4509,77* *+* 901,95 *+ 1087,25 = 6523,5*

**4.4.2 Экономический эффект от использования программного обеспечения у пользователя (заказчика)**

При сравнении базового и нового варианта программного обеспечения в качестве экономического эффекта будет выступать общая сумма экономии всех видов ресурсов относительно базового варианта.

Приведем расчет экономии некоторых видов ресурсов в связи с использованием нового программного обеспечения.

Экономия затрат на заработную плату (*Эз*) при использовании нового программного обеспечения в расчете на объем выполненных работ определяется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.15) |

где *Эзэ* – экономия затрат на заработную плату при решении задач с использованием нового программного обеспечения в расчете на 1 задачу, руб.;

*Q* – количество задач, решаемых за год с использованием нового программного обеспечения.

Экономия затрат на заработную плату в расчете на 1 задачу (*Эзэ*) рассчитывается по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.16) |
|  |  |  |

где *Зсм* – среднемесячная заработная плата одного программиста, руб.;

*Тс1, Тс2 –* снижение трудоемкости на одну задачу в базовом и новом варианте соответственно, чел.-ч.;

*Тч –* количество часов работы в день, ч.;

*Др –* среднемесячное количество рабочих дней.

Экономия с учетом начислений на заработную плату (*Эн*):

|  |  |  |
| --- | --- | --- |
|  |  | (4.17) |

где *Кнз* – коэффициент начислений на заработную плату. Рассчитывается по данным заказчика.

Экономия за счет сокращения простоя сервиса (*Эс*):

|  |  |  |
| --- | --- | --- |
|  |  | (4.18) |

где П1, П2 – время простоя сервиса, обусловленное программным обеспечением в день;

*Дрг –* плановый фонд работы сервиса (дней);

*Сп –* стоимость одного часа, руб.

Общая экономия от применения нового программного обеспечения рассчитывается по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.19) |

где Кнз – коэффициент начислений на заработную плату. Рассчитывается по данным заказчика.

Плановый объем выполняемых работ в год составляет 400 задач. Среднемесячная заработная плата программиста в бюджетной организации на данный момент в Республике Беларусь составляет около 2 601,8 руб. Средняя трудоемкость работ в расчете на 1 задачу 5 и 0,6 человеко-часов в базовом и новом варианте программного обеспечения соответственно. Количество часов работы программиста в день 8 часов. Среднемесячное количество рабочих дней 21 дня. Время простоя сервиса, обусловленное применением программного обеспечения 0,5 и 0,2 дней в базовом и новом варианте соответственно. Плановый фонд работы сервиса 365 дней. Стоимость одного часа простоя 300 руб. (информация по данным предприятия-заказчика).

Экономия затрат на заработную плату в расчете на 1 задачу составит:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Экономия затрат на заработную плату на весь объем запланированных задач составит:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Экономия с учетом начислений на заработную плату:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Экономия за счет сокращения простоя сервиса:

|  |
| --- |
|  |

Рассчитаем сумму общей экономии денежных средств:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Исходя из выше подсчитанных цифр, общая экономия от применения разработанного программного обеспечения составит 34742,7 рублей в год.

## Выводы по экономической части

В данной экономической части проекта представлено обоснование о необходимости выведения программного продукта на рынок с приведением следующих расчетов по разработке ПО: затраты времени на этапы разработки, затраты на материалы, комплектующие, затраты на электроэнергию, заработную плату персоналу, амортизацию материальных и нематериальных активов, приобретение оборудования для разработки, расчет прибыли, отчисления на социальные нужды и налоговые отчисления. Представлен расчет экономической эффективности разработки ПО для (пользователя) заказчика.

Капитальные вложения заказчика на приобретение и внедрение ПО составляют 6523,5 бел. руб. Экономия денежных средств составила для заказчика 34742,7 бел. руб, что позволяет практически в 6 раза сэкономить на разработке разрабатываемого продукта. Исходя из показателей можно сделать вывод, что приобретение и внедрение ПО заказчиком экономически целесообразно.

## 

## ЗАКЛЮЧЕНИЕ

По окончании разработки дипломного проекта была спроектирована и создана два приложения, серверная и клиентская. Серверная часть – telegram daysandbox bot, для удаления сообщений, клиентская часть – для отображения удалённых сообщений.

Серверная часть была выполнена на языке программирования python. Версия python 3.11. Основная библиотека, которая была подключена python-telegram-bot.

Клиентская часть была выполнена на языке программирования C#, на WPF. В проекте использовалась база данных PostgreSQL. Для работы были установлены следующие nuget package: Npgsql.EntityFrameworkCore.PostgreSQL, Microsoft.EntityFrameworkCore.Tools, данные пакеты весьма упростили выполнение миграции базы данных.

В ходе выполнения данной работы использовались две интегрированные среды разработки Visual Studio 2022 и PyCharm 2023. Они значительно упростили процесс написания кода, отладки и управления проектом, предоставив мощные инструменты для разработки качественного программного обеспечения

В результате был создан telegram-bot, который соответствует всем требованиям для корректной работы. Telegram-бот эффективно управляет сообщениями в группах и каналах, удаляя сообщения от старых пользователей. Этот инструмент отвечает всем современным требованиям к безопасности и удобству использования. Бот автоматически отслеживает активность участников и удаляет сообщения, если пользователь не проявлял активности в течение заданного периода. Это позволяет поддерживать порядок в чате и улучшать взаимодействие между активными участниками.

Второе приложение были создано для просмотра удалённых сообщений в группе. Это приложение предоставляет возможность администраторам и участникам группы просматривать информацию о ранее отправленных сообщениях, что позволяет сохранить важные данные и поддерживать прозрачность общения. В сочетании с Telegram-ботом, это решение обеспечивает полный контроль над коммуникацией в сообществе, позволяя эффективно управлять информационными потоками.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Типы онлайн-мошенничества [Электронный ресурс]. – Режим доступа: https://www.keepersecurity.com/blog/ru/2023/11/28/the-types-of-online-scams/# – Дата доступа: 22.03.2025;
2. Telegram: новый мессенджер от Павла Дурова [Электронный ресурс]. – Режим доступа: https://republic.ru/posts/l/978067 – Дата доступа: 22.03.2025;
3. Мессенджер [Электронный ресурс]. – Режим доступа: https://www.tadviser.ru/index.php/Статья:Мессенджеры\_(Іnstant\_Мessenger,\_IM) – Дата доступа: 23.03.2025;
4. Телеграм бот для удаления спама [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/348800/ – Дата доступа: 27.03.2025;
5. Сервисы для анализа Telegram-каналов [Электронный ресурс]. – Режим доступа: https://collaborator.pro/ru/blog/tg-analytiks-tools – Дата доступа: 29.03.2025;
6. Создание тестов в Telegram с помощью QuizBot [Электронный ресурс]. – Режим доступа: https://vc.ru/telegram/1582007-sozdanie-testov-v-telegram-s-pomoshyu-quizbot-prosto-i-udobno – Дата доступа: 01.04.2025;
7. Введение в Python [Электронный ресурс]. – Режим доступа: https://metanit.com/python/tutorial/1.1.php – Дата доступа: 03.04.2025;
8. Язык C# и платформа .NET – METANIT.COM [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/tutorial/1.1.php – Дата доступа: 04.04.2025;
9. Введение в WPF [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/1.php – Дата доступа: 05.04.2025;
10. Сравнение версий PyCharm [Электронный ресурс]. – Режим доступа: [https://ru.hexlet.io/blog/posts/sravnenie-versiy-pycharm-community-edition-vs-professional-edition#](https://ru.hexlet.io/blog/posts/sravnenie-versiy-pycharm-community-edition-vs-professional-edition) – Дата доступа: 10.04.2025;
11. Visual Studio [Электронный ресурс]. – Режим доступа: https://visualstudio.microsoft.com/ru/ – Дата доступа: 20.04.2025;
12. Фундаментальная теория тестирования – Хабр [Электронный ресурс]. – Режим доступа: <https://vc.ru/u/326339-dev-house/848499-razrabotka-igr-na-javascript> – Дата доступа: 10.05.2025;
13. Как считать амортизацию – КонтурЭльба [Электронный ресурс]. – Режим доступа: <https://e-kontur.ru/enquiry/130> – Дата доступа: 29.05.2025;

# ПРИЛОЖЕНИЕ А

# (обязательное)

**Техническое задание**

## Введение

Наименование программного продукта – daysandbox bot. Он, удаляет спам и другие сообщения от новых пользователей.

Данный бот создан для управления чатами в телеграмме. Он помогает фильтровать сообщения в чате, что способствует уменьшению мошенничества через рассылки.

## А.1 Основание для разработки

Бот для управления чатами в телеграмме разрабатывается в рамках дипломного проекта студента учреждения образования «Полоцкий государственный университет имени Евфросинии Полоцкой» Чиникайло А.П. Основанием для разработки является выданное задание к дипломному проекту по теме разработки бота для управления чатами в телеграмме.

## А.2 Назначение разработки

Функциональное и эксплуатационное назначение бота для управления чатами в телеграмме, удалять сообщения от новых пользователей.

Данный бот должен облегчить безопасное нахождение в чате каждого участника чата. В дальнейшем возможен просмотр удалённых сообщений

## А.3 Требования к программе или программному изделию

**А.3.1 Требования к функциональным характеристикам**

Требования к функциональным характеристикам telegram-bot:

* Удалять посты с ссылками от участников, вступивших в чат меньше суток назад.
* Удалять изображения и видео от новых пользователей.
* Удалять любые сообщения от новых пользователей в течение заданного времени.

**А.3.2 Требования к надежности**

Данный бот для управления чатами должен надежно функционировать и обеспечивать стабильность работы. Бот должен быть доступен пользователям 24/7 без значительных простоев. Все данные о пользователях, сообщения которых были удалены, должны надежно сохраняться в базе данных. Так же и сами сообщения должны быть сохранены в базе данных. Бот должен корректно обрабатывать сообщения, что бы удалял только сообщения, от новых пользователей. Данные пользователей и их удалённые сообщения должны быть защищены от несанкционированного доступа и изменений.

**А.3.3 Условия эксплуатации**

Данное программное средство не требует предварительной подготовки.

**А.3.4 Требования к составу и параметрам технических средств**

Для обеспечения устойчивости работы программного средства требуется:

* x64 процессор с тактовой частотой от 1 ГГц и выше;
* 2 ГБ ОЗУ;
* не менее 50 МБ свободного места на жестком диске;
* графический адаптер с минимум 512 МБ видеопамяти, совместимый с DirectX 11 или выше либо с OpenGL 4.6 или выше;

**А.3.5 Требования к информационной и программной совместимости**

Программное средство должно удовлетворять следующему требованию: операционная система Windows 10 и выше, MacOS 12 или выше либо Linux (Ubuntu 22.04 или выше), либо android 4.4.4 и выше, либо IOS 12 и выше. В свою очередь, клиентская часть должна иметь операционную систему Windows 10.

**А.3.6 Требования к маркировке и упаковке**

Требования к маркировке и упаковке отсутствуют.

**А.3.7 Требования к транспортированию и хранению**

Программное средство находится в Telegram и не требует хранения.

## А.4 Требования к программной документации

Программная документация по боту для управления личными финансами с возможностью прогноза бюджета должна быть предоставлена в следующем составе:

* техническое задание. Согласно ГОСТ 19.201-78;
* пояснительная записка. Согласно ГОСТ 19.101-77.

Требования к перечисленным программным документам устанавливаются государственными стандартами ЕСПД.

## А.5 Технико-экономические показатели

Эффективность данного программного продукта обуславливается совокупностью следующих факторов:

‒ Увеличение популярности мессенджеров, таких как Telegram, что создает спрос на дополнительные инструменты для управления сообщениями и взаимодействия с пользователями;

‒ Актуальность функции удаления сообщений от старых пользователей, что позволяет поддерживать чистоту и порядок в чатах, повышая комфорт общения;

‒ Разработка приложения, отображающего удаленные сообщения, предоставляет пользователям возможность сохранять важную информацию и облегчает доступ к ранее удаленному контенту.

Кроме того, постоянная доступность Telegram на мобильных устройствах обеспечивает пользователям легкий доступ к функционалу бота и приложения, что способствует повышению их вовлеченности и удовлетворенности.

## А.6 Этапы разработки

Этапы разработки:

* постановка задачи;
* анализ исходных данных;
* разработка программного средства;
* тестирование программного средства;
* разработка технической документации к данному программному средству.

## А.7 Порядок контроля и приемки

Контроль и приемка программного средства осуществляется в соответствии с программой и методикой испытаний.

Для проверки корректности приложения применялись следующие программные средства:

* ОС Windows 10, Linux (Ubuntu), macOS;
* среда разработки Visual Studio 2022 Community Edition.

Тестирование программы состояло из проверки корректности работы ранее перечисленных функций. Это включало в себя функциональное тестирование, где проверялась каждая функция приложения, а также интеграционное тестирование, где проверялось, как функции взаимодействуют друг с другом.

Основным методом испытания программы является визуальный контроль выполнения программой требующихся функций. Это включает в себя проверку пользовательского интерфейса, анимаций, графического дизайна и общего визуального представления.

Также было проведено тестирование на устойчивость к ошибкам, чтобы убедиться, что приложение может эффективно обрабатывать и восстанавливаться после возникновения ошибок или сбоев.

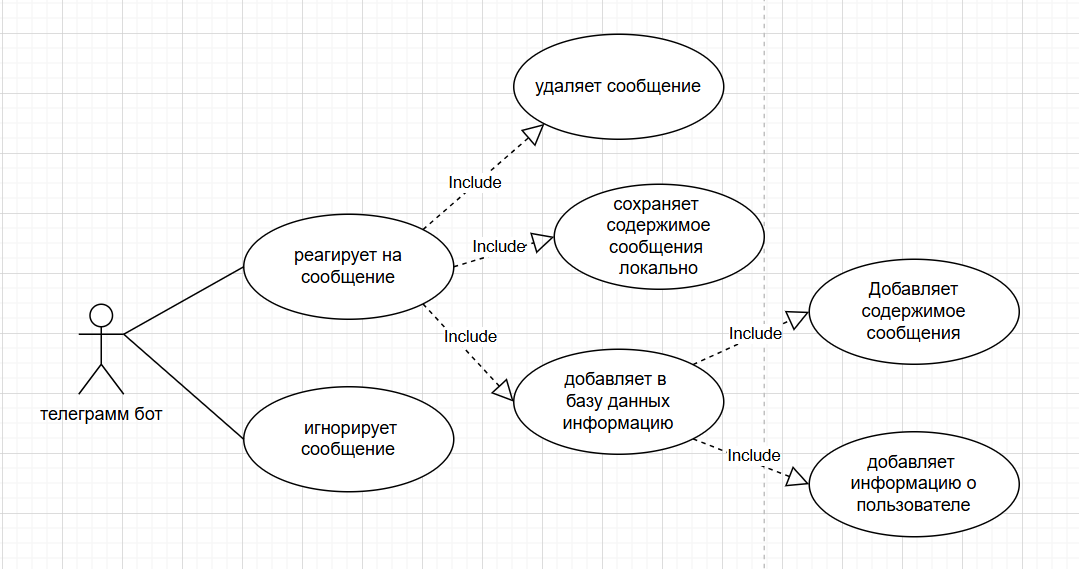
Кроме того, было проведено тестирование совместимости, чтобы убедиться, что приложение корректно работает в различных операционных системах и на различном оборудовании.

Все обнаруженные в процессе тестирования ошибки и недоработки были зарегистрированы, а затем исправлены. После исправления ошибок было проведено повторное тестирование для убеждения в том, что все проблемы были устранены.

# ПРИЛОЖЕНИЕ Б

(справочное)

**Диаграмма вариантов использования**



# ПРИЛОЖЕНИЕ В

(обязательное)

**Программа и методика испытаний**

**В.1 Объект испытаний**

Объектом испытаний является telegram-bot daysandbox.

**В.2 Цель испытаний**

Цель испытаний - проверка программного продукта, telegram-bot, на предмет работоспособности в соответствии с техническим заданием и другой программной документации, а также отказоустойчивости ошибкам в процессе работы приложения.

**В.3 Требования к программе**

Проверка программного продукта на соответствие требованиям технического задания и программной документации включает следующие пункты:

* Работоспособность функциональных модулей;
* Производительность;
* Устойчивость к ошибкам;
* Безопасность;
* Удобство интерфейса.

**В.4 Требования к программной документации**

Дипломный проект должен содержать следующие элементы программное документации:

* техническое задание;
* описание программы;
* руководство пользователя.

**В.5 Средства и порядок испытаний**

**В.5.1 Средства для испытаний**

Для проведения испытания над приложением необходимо наличие персонального компьютера, с установленным на него .NET фреймворком. Для проверки корректности функционирования разработанного программного продукта должны применяться следующие программные средства:

* процессор с тактовой частотой от 2 ГГц;
* оперативная память 4 Гб+;
* наличие .NET фреймворка версии, на которой разрабатывался программный продукт;
* операционная система Windows 10 или 11 версии;
* 100 Мб свободного места;
* оперативная память 1 Гб+.

**В.5.2 Порядок испытаний**

Проверка приложения выполнялась на операционной системе Windows 10 в соответствии с тест-планом, представленным в таблицу В.1.

Таблица В.1 – Тест-план проверки мобильного приложения

|  |  |  |  |
| --- | --- | --- | --- |
| Тестовый вариант | Входные данные | Ожидаемый результат | Результат тестирования |
| Удаление  текстового сообщения | Пользователи, который находится в чате меньше одного часа, присылает текстовое сообщение | Сообщение  удаленно | Тест пройден успешно |
| Удаление фото сообщения | Пользователи, который находится в чате меньше одного часа, присылает фото сообщение | Сообщение удаленно | Тест пройден успешно |

Продолжение таблицы В.1

|  |  |  |  |
| --- | --- | --- | --- |
| Тестовый вариант | Входные данные | Ожидаемый результат | Результат тестирования |
| Удаление видел сообщения | Пользователи, который находится в чате меньше одного часа, присылает видео сообщение | Сообщение удаленно | Тест пройден успешно |
| Удаление  текстового сообщения с ссылкой | Пользователи, который находится в чате меньше суток, присылает текстовое сообщение с ссылкой | Сообщение удаленно | Тест пройден успешно |
| Игнорирование  текстового сообщения | Пользователи, который находится в чате больше одного часа, присылает текстовое сообщение | Сообщение  было  проигнорировано | Тест пройден успешно |
| Игнорирование  фото сообщения | Пользователи, который находится в чате больше одного часа, присылает фото сообщение | Сообщение  было  проигнорировано | Тест пройден успешно |
| Игнорирование  видел сообщения | Пользователи, который находится в чате больше одного часа, присылает видео сообщение | Сообщение  было  проигнорировано | Тест пройден успешно |

Продолжение таблицы В.1

|  |  |  |  |
| --- | --- | --- | --- |
| Тестовый вариант | Входные данные | Ожидаемый результат | Результат тестирования |
| Игнорирование  текстового сообщения с ссылкой | Пользователи, который находится в чате больше суток, присылает текстовое сообщение с ссылкой | Сообщение  было  проигнорировано | Тест пройден успешно |

**В.6 Методы испытаний**

Проверка программного продукта должна осуществляться методом «Черного ящика», который предназначен для проверки функционального назначения программы.

Корректность выполнения определяется сравнением фактического и ожидаемого результата. Если результаты совпали, то тестирование пройдено успешно. Обратные результаты будут говорить о проблемах и сбоях в работе программного продукта и необходимости устранения неполадок на определенных участка, соответственно. Исходя из этого можно будет сделать вывод, что тестирование программы не было пройдено успешно.