СОДЕРЖАНИЕ

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 курсовой 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.

1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧИ

1.1 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 б4 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 C++. 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 MTProto 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.

2) 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.

1.2 Сравнительный анализ аналогов и прототипов

Сравним нашего бота Daysandbox-бот с telemetrmebot и QuizBot.

Принцип работы бота Daysandbox.

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. Какие сообщения считаются спамом? Во-первых, любые сообщения содержащие ссылку: URL, email или username. Во-вторых, любые forward-сообщения т.е. сообщения пересланные из другого чата. Для правила username есть исключение, если username ссылается на пользователя, то такое сообщение разрешается. Бот удаляет только те сообщения с 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].

Принцип работы telemetrmebot – 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. Процесс простой:

– 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 курсовой работы, является крайне необходимым в наше время, так как обеспечивает безопасность в telegram чатах, в отличии от других ботов, которые обеспечивают простоту управления чатами.

1.3 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 курсовой 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.

1.4 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 получении информации о дате, когда пользователь был добавлен.

1.5 Выбор и обоснование средств и методов решения задач

При разработке курсового 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 C++. 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.

1.6 Разработка технического задания

При разработке курсового 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].

2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 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. Бот основная логика приложения, реализованная на языке программирования и 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 получает все сообщения из чата и проверяет, является ли отправитель новым пользователем. Удаление сообщений, то есть если сообщение отправлено новым пользователем, бот автоматически удаляет его из чата, а так же 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. Так же 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.

2.2 Проектирование диаграммы вариантов использования

Диаграмма вариантов использования для чат-бота будет отображать ключевых пользователей, их взаимодействие с системой и функциональные задачи, которые бот должен выполнять. Актером будет любой пользователь, находящийся в чате, вместе с чат-ботом. 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.

2.3 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.

2.4 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.

2.5 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.

2.6 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.

3 РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ

3.1 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 были созданы методы, для добавления и удаления пользователя в базу данных. Данные методы находятся в файле, который представлен в листинге 3.1.

Листинг 3.1– handleJoinOrRemoveFromChat.py

```
from datetime import datetime
from typing import Optional, Tuple

import psycopg2
from telegram import ChatMember, ChatMemberUpdated, Update
from telegram.constants import ParseMode
from telegram.ext import (
    ContextTypes,
)

from constants import *


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


async def greet_chat_members(update: Update, context:
ContextTypes.DEFAULT_TYPE) -> None:
    """Greets new users in chats and announces when someone
leaves"""
    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()
    # cursor.execute("SELECT * FROM DateJoined")
    # print(cursor.fetchone())

    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:
        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))
        # datetime_user = cursor.fetchone()
        # print(datetime_user[0])
        # print(datetime.datetime.now() - datetime_user[0])
        cursor.execute(sqlDeleteDateJoined.format(new_user.id))

    conn.close()
```

Для обработки события, когда пользователь отправил сообщение в виде изображения, были созданы методы, для добавления текущего сообщения в базу данных и удаления самого сообщения. Данные методы находятся в файле, который представлен в листинге 3.2.

Листинг 3.2 – handlePhotoMessage.py

```
from datetime import datetime

import psycopg2
from telegram import Update
from telegram.ext import (
    CallbackContext,
)

from constants import *

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"
    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 сообщение в виде видео, были созданы методы, для добавления текущего сообщения в базу данных и удаления самого сообщения. Данные методы находятся в файле, который представлен в листинге 3.3.

Листинг 3.3 – handleVideoMessage.py

```
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 =
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()


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 сообщение в виде текста, были созданы методы, для добавления текущего сообщения в базу данных и удаления самого сообщения. Данные методы находятся в файле, который представлен в листинге 3.4.

Листинг 3.4 – handleTextMessage.py

```
from datetime import datetime

import re
import psycopg2
from telegram import Update
```

```
from telegram.ext import (
    CallbackContext,
)
from constants import *


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. Данные константы находятся в файле, который представлен в листинге 3.5.

Листинг 3.5 – 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 так же токен для бота. Данный файл представлен в листинге 3.6.

Листинг 3.6 – config.env

```
TOKEN = "8093292418:AAEPYVGW4B-qrCLD7ridWiKhZSryLZTaEKI"

sqlInsertDateJoined = "INSERT INTO dateJoined(id, datetime)
            VALUES({}, {})"

sqlSelectDateJoined = "Select datetime FROM dateJoined
            WHERE id = {}"

sqlDeleteDateJoined = "Delete FROM dateJoined
```

Продолжение листинга 3.6

```
            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 = {}"
```

3.2 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 BotAdministratorContext показан в листинге 3.7.

Листинг 3.7– 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)
    {
        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=BotAdmi
nistrator;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)
                .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);
}
```

Сгенерированный класс Datejoined представлен в листинге 3.8.

Листинг 3.8 – Datejoined.cs

```
namespace DisplayDeletedMessages.Models;

public partial class Datejoined
{
    public int Id { get; set; }

    public DateTime Datetime { get; set; }
}
```

Сгенерированный класс Deletedmessage представлен в листинге 3.9.

Листинг 3.9 – Deletedmessage.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;

namespace DisplayDeletedMessages.Models;

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!;

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

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

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

Сгенерированный класс User представлен в листинге 3.10.

Листинг 3.10 – User.cs

```
using System;
using System.Collections.Generic;

namespace DisplayDeletedMessages.Models;

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>();
}
```

Для отображения данных с базы данных, была создана окно MainWindow. Разметка данного окна, в виде xaml, представлена в листинге 3.11.

Листинг 3.11 – 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">
            <DataGrid.Columns>
                <DataGridTextColumn Header="Id"  Binding="{Binding
Id}"/>
                <DataGridTextColumn Header="Username"
Binding="{Binding Username}"/>
                <DataGridTextColumn
Header="DatetimeOfMessageSending" Binding="{Binding
DatetimeOfMessageSending}"/>
```

Продолжение листинга 3.11

```
                <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 3.12.

Листинг 3.12 – 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 = "../../../../";
        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
                };

                try
                {
                    Process.Start(startInfo);
                }
```

Продолжение листинга 3.12

```
                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);
                }
            }
        }

    }
}
```

3.3 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).

3.4 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.

ЗАКЛЮЧЕНИЕ

По окончании разработки курсового 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: 22.10.2024;

2 Telegram: новый мессенджер от Павла Дурова [Электронный ресурс]. – Режим доступа: https://republic.ru/posts/l/978067 – Дата доступа: 22.10.2024;

3 Мессенджер [Электронный ресурс]. – Режим доступа: https://www.tadviser.ru/index.php/Статья:Мессенджеры_(Instant_Messenger,_IM) – Дата доступа: 23.10.2024;

4 Телеграм бот для удаления спама [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/348800/ – Дата доступа: 27.10.2024;

5 Сервисы для анализа Telegram-каналов [Электронный ресурс]. – Режим доступа: https://collaborator.pro/ru/blog/tg-analytiks-tools – Дата доступа: 29.10.2024;

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: 01.11.2024;

7 Введение в Python [Электронный ресурс]. – Режим доступа: https://metanit.com/python/tutorial/1.1.php – Дата доступа: 03.11.2024;

8 Язык C# и платформа .NET – METANIT.COM [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/tutorial/1.1.php – Дата доступа: 04.11.2024;

9 Введение в WPF [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/1.php – Дата доступа: 05.11.2024;

10 Сравнение версий PyCharm [Электронный ресурс]. – Режим доступа: https://ru.hexlet.io/blog/posts/sravnenie-versiy-pycharm-community-edition-vs-professional-edition# – Дата доступа: 10.11.2024;

11 Visual Studio [Электронный ресурс]. – Режим доступа: https://visualstudio.microsoft.com/ru/ – Дата доступа: 20.11.2024;

12 Фундаментальная теория тестирования – Хабр [Электронный ресурс]. – Режим доступа: https://vc.ru/u/326339-dev-house/848499-razrabotka-igr-na-javascript – Дата доступа: 25.11.2024;

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 курсового проекта студента учреждения образования «Полоцкий государственный университет имени Евфросинии Полоцкой» Чиникайло А.П. Основанием для разработки является выданное задание к курсовому 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.

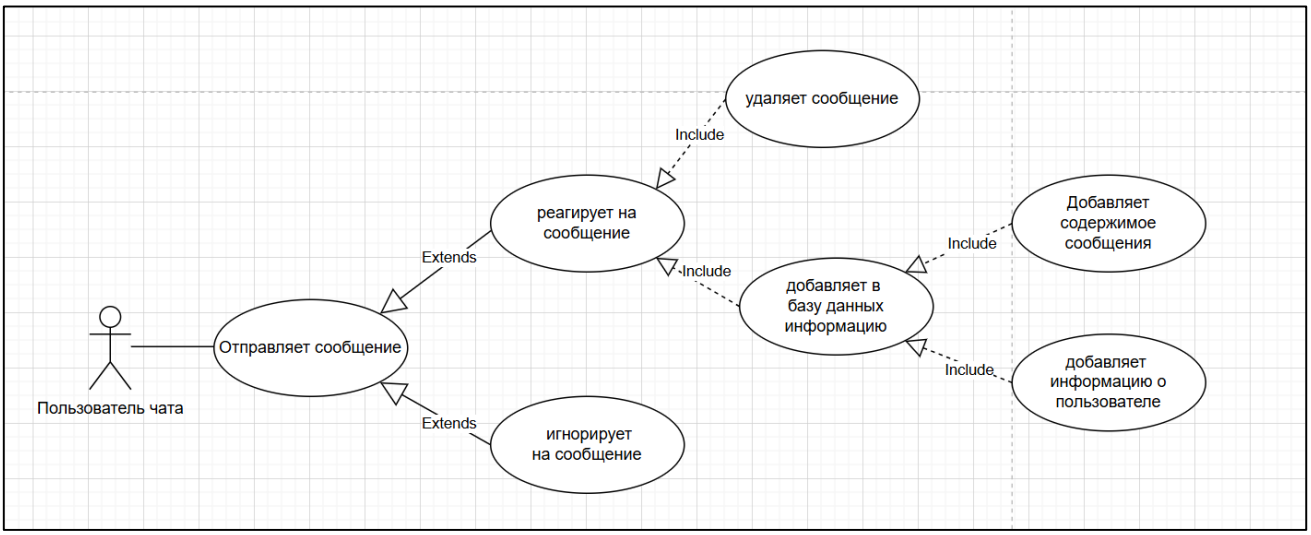
Основным методом испытания программы является визуальный контроль выполнения программой требующихся функций. Это включает в себя проверку пользовательского интерфейса, анимаций, графического дизайна и общего визуального представления.

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

Курсовой 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.