Министерство науки и высшего образования Российской Федерации ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных технологий

наименование факультета

Кафедра Кафедра компьютерной безопасности и математического обеспечения информационных систем

наименование кафедры

**КУРСОВАЯ РАБОТА**

по Языки программирования

наименование дисциплины, модуля

**Проектирование и разработка системы логирования событий в мессенджере Telegram**

наименование темы

ОГУ 10.05.01 3024 ОО

Руководитель

К.Р. Джукашев

подпись инициалы фамилия

Студент группы 23КБ(с)РЗПО-1

А.А. Коновалов

подпись инициалы фамилия

« » 20 г.

Оренбург 2024

Утверждаю

Заведующий кафедрой КБМОИС

                           И.В. Влацкая

«        »                         2024 г.

# ЗАДАНИЕ

**на выполнение курсовой работы**

студенту Коновалову Андрею Алексеевичу

специальности 10.05.01 Компьютерная безопасность

по дисциплине «Языки программирования»

1 Тема работы Проектирование и разработка системы логирования событий в мессенджере Telegram

2 Срок сдачи студентом проекта (работы) «    »   \_\_\_\_\_\_\_      2024 г.

3 Цель и задачи работы Разработать программный продукт для сохранения, упорядочивания и отображения истории событий аккаунта Telegram.

Задачи:

1. Разработать структуру хранения логов.
2. Разработать серверную часть системы логирования
3. Спроектировать и разработать интерфейсную часть с использованием технологии WebApp.
4. Реализовать авторизацию пользователя(ей) в аккаунт Telegram через веб-интерфейс.
5. Обеспечить защиту передаваемых данных посредством протокола https.

4 Исходные данные к проекту (работе): Учебная и научная литература содержащая:

- теоретические и практические сведения о платформе разработки .NET и языке C#;

- теоретические и практические сведение о языке разработки JavaScript;

- информацию об основах криптографии и о системах сквозного шифрования;

- документация Telegram Bot Api и Telegram Api

5 Перечень вопросов, подлежащих разработке:

1. Освоить на практике разработку с использованием языков C# и JavaScript.
2. Изучить методы Telegram Api, позволяющие осуществить логирование событий.
3. Изучить методы Telegram Bot Api, позволяющие встроить WebApp в бота Telegram.
4. Разработать и реализовать программный продукт, обеспечивающий сохранение, упорядочивания и отображения истории событий аккаунта Telegram.

Дата выдачи и получения задания

Руководитель «       »                      2024 г.                      К.Р. Джукашев

подпись инициалы фамилия

Студент «       »                      2024 г.                      А.А. Коновалов

подпись инициалы фамилия

# Аннотация

Курсовая работа посвящена созданию серверного приложения, цель которого записывать все события, происходящие на аккаунте Телеграм, а также разработку веб-интерфейса для просмотра сохранённых записей и управлением конфигурацией сохранения. Основные задачи включают разработку серверного приложения, которое отвечает за сохранение или «логирование» событий, Телеграм бота для управления серверным приложением прямо из мессенджера, а также веб-интерфейса встроенного в Телеграм бота.

Работа содержит … листов текста, …

# Содержание

# Введение

# Терминология

**Пользователь** – Телеграм аккаунт, с которого происходит взаимодействие с веб-приложением.

**Аккаунт** – Телеграм аккаунт, с которым происходит взаимодействие.

**Сервер** – серверная часть приложения или “бэкенд”.

**Бот** – Телеграм бот с помощью которого пользователь будет открывать веб-приложение.

**Логи** – журнал. Под журналом имеется в виду структура хранения записей в базе данных.

**Лог** – запись в журнале. Под записью имеется в виду строка в одной из таблиц базы данных.

# Техническое задание

Разработать программный продукт интегрированный в мессенджер Телеграм по средствам Telegram API и Telegram Bot Api.

* Создать Телеграм бота.

Управление должно быть реализовано с использованием веб-приложения или Web-App. Через это приложение можно будет ввести свои данные для входа в аккаунт Телеграм и отслеживать события, происходящие на этом аккаунте через это веб-приложение. Один пользователь должен иметь возможность добавить несколько аккаунтов.

* Разработать веб-сайт для мобильного формата.
* Создать кнопку в Телеграм боте для открытия сайта как веб-приложения внутри Телеграм.

Авторизация и аутентификация. Определять от какого пользователя пришёл запрос и проверять не самозванец ли этот пользователь.

* Создание сессии
* Проверка подписи

Между пользователями должно быть разграничение прав.

* Реализовать передачу доступа к аккаунту другому пользователю.
* Реализовать белый и черный списки пользователей.

Сервер должен запускать сессии пользовательских аккаунтов и логировать (запоминать) события, пришедшие на них. Логи должны быть дополняемыми. *Пример: пришло сообщение – лог; сообщение было отредактировано – новый лог со ссылкой на старый (родительский) лог.*

* Разработать систему хранения событий.
* Написать приложение для логирования событий.
* Написать веб-сервер (бэкенд) для предоставления информации (логов) для веб-приложения.
* Реализовать поддержку изменения конфигурации «на горячую» через веб-приложение. В конфигурацию входят: аккаунты для отслеживания, доступ к аккаунтам, изменение логируемых чатов. Должна быть возможность редактировать эти параметры без перезапуска приложения.

# Выбор методов, способов и средств разработки

Серверное приложение будет разработано на платформе ASP.NET8.0. Благодаря веб-серверу можно будет легко взаимодействовать с серверной частью приложения.

Для реализации получения событий с Телеграм аккаунта будет использоваться библиотека WTelegram для C#. Её код находится в открытом доступе и её использование бесплатно. Библиотека предоставляет удобный интерфейс для взаимодействия с Telegram API.

Для работы с базой данных будет использоваться Entity Framework, позволяющий удобно работать с данными из БД в представлении обычных объектов классов.

Взаимодействие пользователя с программой будет в веб-приложении, встроенном в Телеграм бота. Веб-приложение будет написано с использованием библиотеки React для JavaScript.

Веб-приложение будет развёрнуто на GitHub pages. Это бесплатный и простой способ развёртывания веб-страницы с SSL сертификатом.

Серверная инфраструктура (приложение и СУБД) будет упакована в docker-compose, что позволит быстро разворачивать его на любом сервере.

# Описание алгоритмов

## Использование технологии WebApp

Открывающееся в Телеграме веб-приложение это по сути своей обычный сайт. Отличие лишь в том, что подключена библиотека, которая выполняет функции авторизации и вызовом некоторого функционала Телеграм. Подробнее о функционале можно узнать в официальной документации Telegram mini apps, но сейчас нас интересует именно авторизация.

Авторизация через Телеграм позволит не реализовывать свою авторизацию, позволит не хранить хеши паролей и т.д. В таблице БД с пользователями мы будет просто хранить user\_id для каждого пользователя, который взаимодействовал с ботом.

Звучит довольно просто, но как этим пользоваться? В подключаемой библиотеке есть поле *initData*. Это поле храни строку с данными о пользователе, откуда и когда он открыл веб-приложение и хеш этой *initData*. Этот хеш используется для аутентификации данных. Благодаря нему, посторонний не сможет воспользоваться нашим API.

Для того, чтобы проверить пришедшую на бэкенд *initData*, нужно выполнить следующие шаги:

* Получить data\_check\_stirng путём сортировки всех полей полученных данных по ключу и записи их в виде:

<Ключ[1]>=<Значение[1]>

<Ключ[2]>=<Значение[2]>

…

<Ключ[n]>=<Значение[n]>

* Получить хеш для *secretKey* по алгоритму HMAC-SHA256, где ключ — это константная строка *“WebApp”*, а хешируемые данные – секретный API ключ Телеграм бота.
* Получить финальный хеш по алгоритму HMAC-SHA-256, где ключ это secretKey, а данные это dataCheckString.

Финальный хеш нужно сравнить с тем, что находится в полученной *initData*. При несовпадении этих хешей, отбрасываем запрос с этой *initData*.

## Авторизация и аутентификация

Авторизация будет осуществляться по заголовку userId в запросе к серверу, а аутентификация в свою очередь по подписи пакета.

Схема такая: при открытии веб-приложения, первым запросом отправляется auth/logIn с данными initData. Важно, что в заголовках должно присутствовать userId: -1, для обозначения того, что клиент ещё не авторизован. В ответе будет находиться строковое поле sessionCode и объектом me. Объект me – это объект информации о пользователе, который открыл веб-приложение. Там и будет тот самый userId, взятый в свою очередь из отправленной initData. Следующие запросы должны содержать в заголовках userId с полученными id.

Аутентификация будет осуществляться через подпись отправляемых данных. Подписывать данные нужно строкой sessionCode. В свою очередь sessionCode имеет время жизни, и сбросить это время жизни можно вызовом любого метода. Если в данный момент нет необходимости вызывать какой-то метод, а сбросить время жизни нужно, вызывается метод auth/ping.

## Длинные и короткие задачи

Короткая задача – это задача результат которой будет достигнут за один запрос к серверу. Длинная задача – это задача требующая для результата несколько запросов. Пример: вход в аккаунт. Для выполнения этой задачи, нужно разбить её на несколько маленьких задач, это:

* Клиент отправляет номер телефона на сервер, а сервер отправляет запрос для входа на сервера телеграм.
* Клиент отправляет код для входа и сервер отправляет его на сервера телеграм.
* (Опционально) клиент отправляет серверу пароль от аккаунта и сервер отправляет его не сервера телеграм.

Разделение на такие задачи нужны для простой вещи: после выполнения маленькой подзадачи, основная задача может быть ещё не завершена, а значит что за ней нужно следить и в случае сбоя или прекращения её выполнения пользователем нужно освободить память от этой задачи или «забыть» её. Также у большой задачи может быть время жизни. Например то время, которое код для входа в аккаунт действителен. Сервер должен это отслеживать, поэтому и вводиться понятие «Длинная задача».

У длинной задачи будут состояния. В состояниях хранятся данные, необходимые для выполнения следующей подзадачи. Например, после отправки номера телефона, сервер будет хранить в состоянии соединение с серверами телеграм для которого он инициировал вход в аккаунт для этого конкретного пользователя, чтобы после получения кода для входа сервер мог отправить этот код в это соединение, чтобы сервера телеграм приняли этот код.

После выполнения длинной задачи, она очищается из памяти, т.е. данные о состояниях забываются.