Министерство образования Республики Беларусь

Учреждение образования  
 БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
 ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Рефакторинг и оптимизация программного кода

Отчет

по практической работе №4

на тему:

Реализация системы аутентификации и авторизации пользователей ПС и механизмов обеспечения безопасности данных

|  |  |  |  |
| --- | --- | --- | --- |
| Проверил | |  | А.В. Шелест |
|  | (подпись) | |  |
| зачтено |  | |  |
|  | (дата защиты) | |  |
|  |  | |  |
| Выполнил |  | | Н.В. Верховодко  гр. 114302 |
|  | (подпись) | |  |

Минск, 2024

**Содержание**

[1 Реализация системы аутентификации и авторизации пользователей ПС и механизмов обеспечения безопасности данных 3](#_Toc177988985)

2 Список использованных источников……..……………….…………....9

# **1** РЕАЛИЗАЦИЯ СИСТЕМЫ АУТЕНТИФИКАЦИИ И АВТОРИЗАЦИИ ПОЛЬЗОВАТЕЛЕЙ ПС И МЕХАНИЗМОВ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ДАННЫХ

В данном практическом занятии основной акцент делается на внедрении механизма аутентификации и авторизации с использованием технологии *JSON Web Token* (*JWT*). *JWT* — это современный стандарт для передачи данных в виде компактных и безопасных *JSON*-объектов, который широко используется для аутентификации в веб-приложениях. Он позволяет реализовать надежную систему контроля доступа и легко интегрируется в современные архитектуры программных систем.

*JWT* представляет собой токен, содержащий информацию о пользователе и его правах доступа, подписанный с использованием ключа, что гарантирует целостность данных и их защиту от подделки. При аутентификации сервер генерирует *JWT*-токен и отправляет его клиенту, который затем использует его для доступа к защищённым ресурсам. Важным преимуществом использования *JWT* является то, что проверка токена может происходить без необходимости постоянного обращения к серверу, так как токен содержит всю необходимую информацию.

В ходе выполнения задания необходимо реализовать аутентификацию и авторизацию с помощью *JWT*. Также потребуется описать и продемонстрировать использование сторонних компонентов, если они применялись, и обеспечить безопасность данных с помощью методов шифрования и механизма контроля прав доступа. В случае изменений в архитектуре системы, связанных с внедрением *JWT*, необходимо их описать.

Для реализации аутентификации и авторизации с использованием *JSON* *Web* *Token* (*JWT*) в .*NET* требуется выполнить несколько шагов, начиная с настройки инфраструктуры токенов и заканчивая их использованием в контроллерах для защиты *API* [1].

Для начала необходимо добавить зависимость *Microsoft*.*AspNetCore*.*Authentication*.*JwtBearer* --*version* 8.0.1

После чего Добавить в файл *appsettings*.*json* секцию с параметрами для *JWT*, пример изображен на рисунке 1.

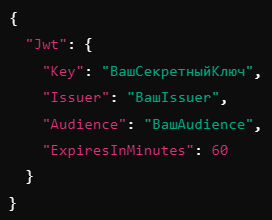


Рисунок 1 – *Файл appsettings.json* с параметрами для *JWT*

Теперь необходимо настроить *JWT*-аутентификацию в файле *Program.cs*, пример на рисунке 2.

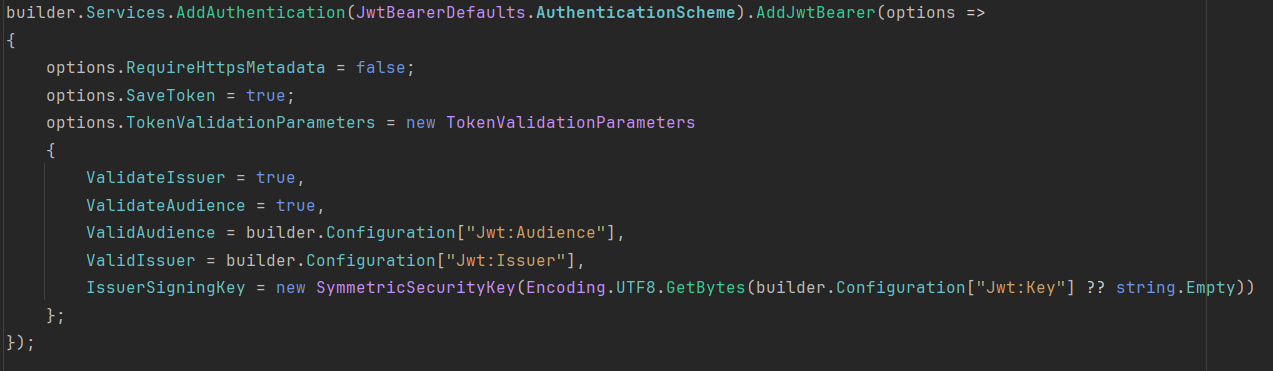


Рисунок 2 – Конфигурация *JWT*

Теперь нужно создать метод для генерации *JWT*-токена. Этот метод можно реализовать в сервисе, который будет заниматься аутентификацией, пример кода на рисунке 3.

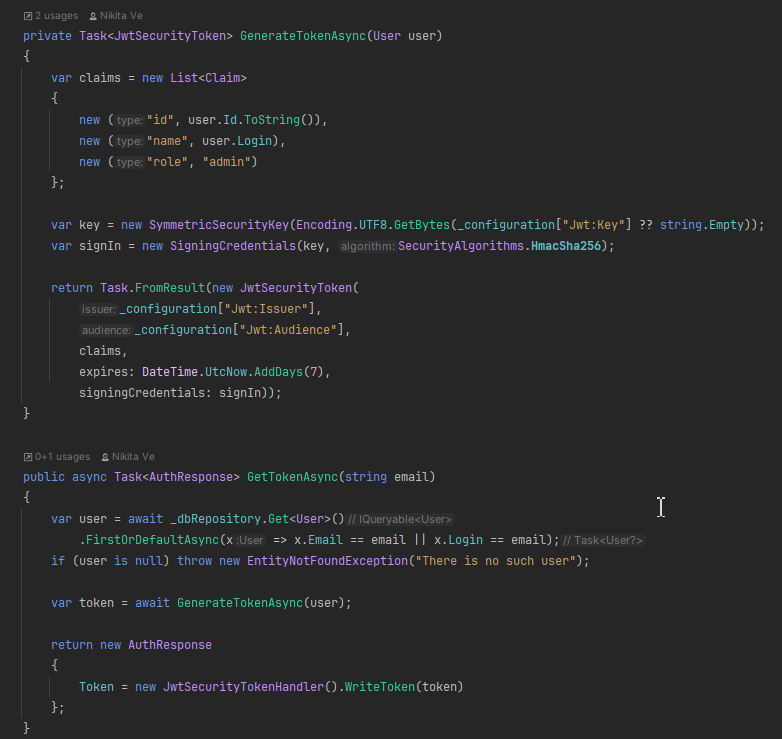


Рисунок 3 – Генерация *JWT*

Пример использования аутентификации в контроллере, где происходит проверка учетных данных и генерация *JWT*-токена. Достаточно повесить атрибут *Authorize* над контроллером или методами отдельно, пример код на рисунке 4

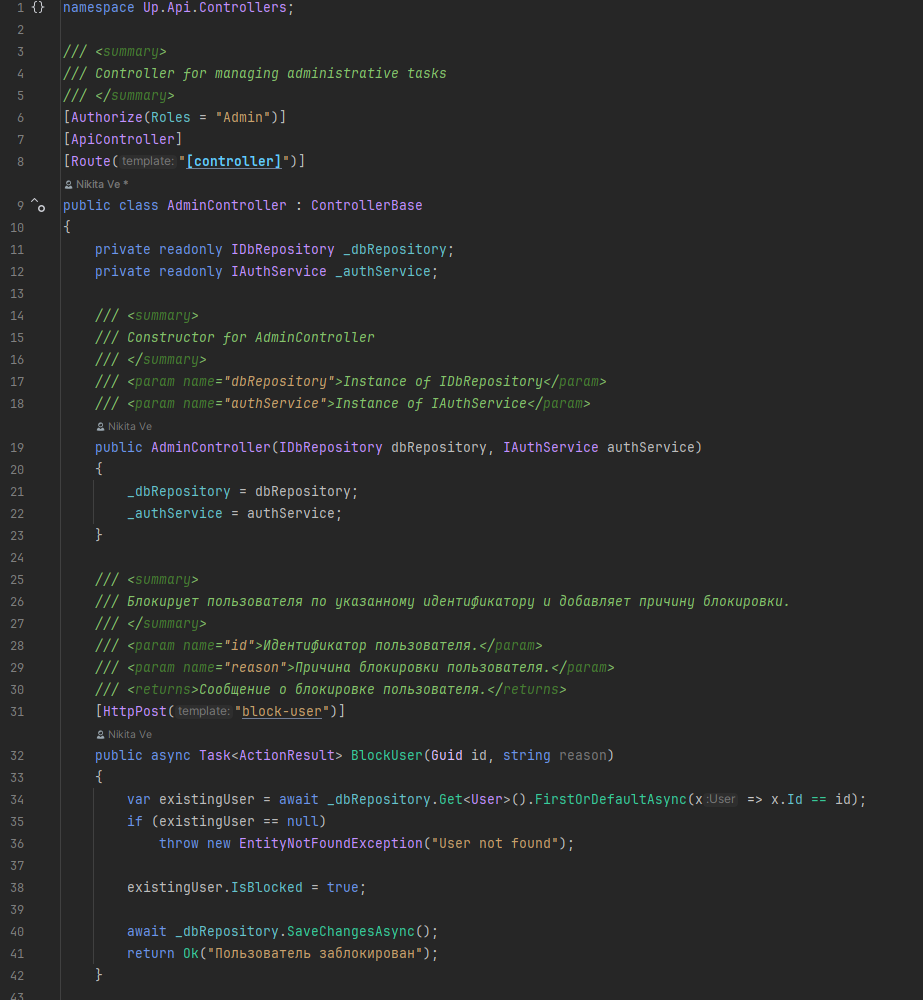


Рисунок 4 – Пример использования атрибута *Authorize*

*JWT* является мощным инструментом для реализации аутентификации и авторизации в современных веб-приложениях. В .*NET* его легко интегрировать и настроить с использованием встроенных средств, что позволяет быстро обеспечить безопасность *API.*

Также в программной системе (ПС) предусмотрено хеширование паролей с солью для повышения безопасности данных пользователей. Хеширование пароля с добавлением соли – это важный шаг для защиты паролей в базе данных от потенциальных атак, таких как атака по словарю или *brute*-*force*.

Хеширование – это процесс преобразования пароля в фиксированную строку символов (хеш). При этом, даже если два пользователя вводят одинаковые пароли, результат хеширования без соли будет одинаковым, что создает уязвимость [2].

Соль – это случайная строка данных, которая добавляется к паролю перед хешированием. Это делает результат хеширования уникальным даже для одинаковых паролей. Соль предотвращает использование предварительно вычисленных таблиц хешей.

Пример реализации хеширования паролей с солью в .*NET* изображен на рисунке 5.

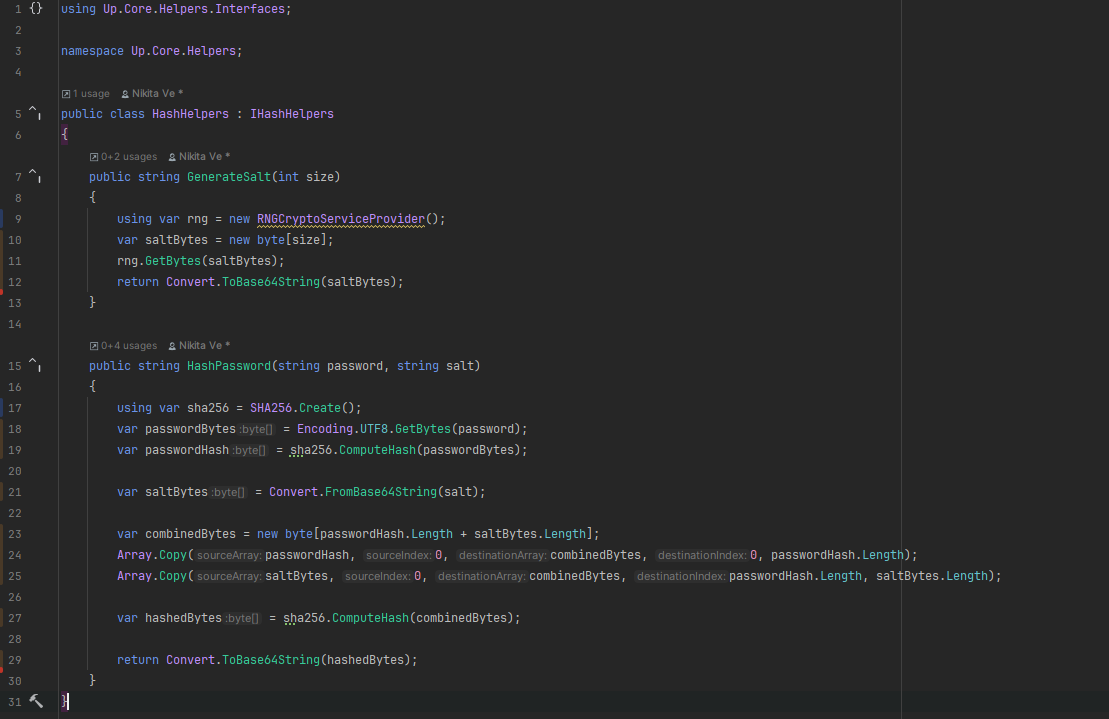


Рисунок 5 – Пример реализации хеширования

Хеширование паролей с солью повышает безопасность, так как злоумышленникам сложнее подобрать пароли даже при наличии хешей. В сочетании с использованием *JWT* для аутентификации, ПС получает более надежную защиту данных пользователей.

Вывод:

В ходе выполнения работы были исследованы различные подходы к реализации аутентификации и авторизации в программной системе (ПС), включая использование сторонних компонентов для внедрения стандартов, таких как JWT. Реализованная система аутентификации и авторизации успешно интегрировалась в архитектуру ПС, что обеспечило безопасное управление доступом пользователей.

Кроме того, были предусмотрены дополнительные механизмы защиты данных, такие как шифрование и разграничение прав доступа, что значительно повысило уровень безопасности ПС. Результаты работы, включая примеры кода и описание компонентов, были добавлены в репозиторий.

Ссылка на репозиторий *GitHub*: https://github.com/NVERKHOVODKO/crypto-wallet

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] JSON Web Token [Электронный ресурс]. – Режим доступа https://en.wikipedia.org/wiki/JSON\_Web\_Token. – Дата доступа: 23.10.2024.

[2] SHA256 Klasa [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/pl-pl/dotnet/api/system.security.cryptography.sha256?view=net-8.0. – Дата доступа: 23.10.2024.