Министерство образования Республики Беларусь

Учреждение образования  
 БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
 ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Анализ и оптимизация бизнес-процессов

Отчет

по результатам выполнения задания к практическому занятию №4

на тему:

## 

**МЕТОДЫ РЕФАКТОРИНГА КОДА**

|  |  |  |
| --- | --- | --- |
| Проверил |  | А.В. Шелест |
|  | (подпись) |  |
| зачтено |  |  |
|  | (дата защиты) |  |
|  |  |  |
| Выполнил |  | Ю-Ф.А. Тюленев  гр. 114301 |
|  | (подпись) |  |

Минск, 2024

Цель: *реализация системы аутентификации и авторизации пользователей ПС и механизмов обеспечения безопасности данных*

Ссылка на репозиторий *GitHub*: [RSAS](https://github.com/FranGranDev/RSAS)

**ASP.NET Core Identity** — это фреймворк для управления аутентификацией и авторизацией в приложениях. Он предоставляет готовые механизмы для работы с пользователями, ролями, входом в систему, управлением паролями, а также для интеграции со сторонними провайдерами (например, Google, Facebook, Twitter).

Основные компоненты ASP.NET Core Identity:

1. Пользователи (Users):
   * Identity работает с пользовательскими учетными записями.
   * Каждый пользователь представлен объектом класса IdentityUser, который можно расширить для добавления дополнительных свойств.
2. Роли (Roles):
   * Роли предоставляют способ группировать пользователей и управлять их доступом.
   * Роли управляются через классы IdentityRole и RoleManager.
3. Хранилище данных (Stores):
   * Identity поддерживает хранилища, такие как базы данных (по умолчанию Entity Framework Core), где сохраняются данные пользователей, ролей и другие сведения.
   * Также можно создать собственные хранилища (Custom Stores).
4. Хеширование паролей:
   * Identity автоматически хеширует пароли перед их сохранением.
   * Используется алгоритм PBKDF2, а в более новых версиях .NET Core — ASP.NET Core Data Protection для повышения безопасности.
5. Аутентификация:
   * ASP.NET Core поддерживает различные способы аутентификации: Cookie, JWT, OAuth2 и OpenID Connect.
   * По умолчанию Identity использует cookie-based аутентификацию.
6. Авторизация:
   * Identity предоставляет механизмы проверки прав доступа через роли и политики.

**Аутентификация**

Аутентификация — это процесс проверки подлинности пользователя. В ASP.NET Core Identity она реализована следующим образом:

1. Cookie-based аутентификация:
   * После успешного входа создается аутентификационный cookie.
   * Cookie хранит зашифрованные данные, представляющие пользователя (например, его идентификатор).
2. Встроенные механизмы входа:
   * Поддерживаются методы аутентификации с помощью имени пользователя и пароля.
   * Возможна интеграция со сторонними провайдерами, такими как Google, Microsoft, Facebook, или GitHub, через AddAuthentication().
3. Настройка аутентификации:
   * Конфигурация аутентификации происходит в Startup.cs или файле конфигурации (рисунок 1):

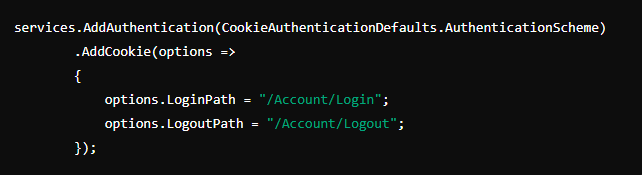


Рисунок 1 – Конфигурация аутентификации

**Авторизация**

Авторизация проверяет, имеет ли пользователь доступ к определенным ресурсам или действиям.

1. Роли (Roles):
   * Использование ролей — базовый способ авторизации.
   * Например, можно назначить пользователю роль "Admin" и ограничить доступ к определенным частям приложения(рисунок 2):

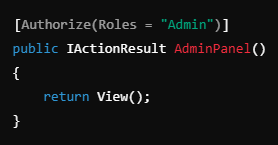


Рисунок 1 – Использование аттрибута [Authorize]

1. Политики (Policies):
   * Политики предоставляют более гибкий способ авторизации(рисунок 3):

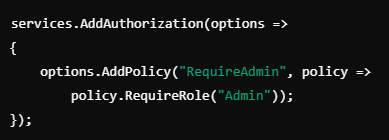


Рисунок 3 – Пример политик

1. Пример проверки на уровне кода:
   * Проверка прав доступа в коде контроллера(рисунок 4):

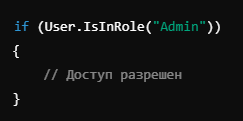


Рисунок 4 – Проверка прав доступа в коде

**Хеширование паролей**

ASP.NET Core Identity использует современные алгоритмы для безопасного хеширования паролей:

1. Алгоритм PBKDF2:
   * Используется для создания сильных хешей паролей с солью.
   * Соль генерируется автоматически и добавляется к каждому паролю для защиты от атак типа «радужные таблицы».
2. Интеграция с Data Protection:
   * Хеши паролей также могут быть защищены механизмами ASP.NET Core Data Protection для дополнительной безопасности.
3. Пример хеширования пароля:
   * Identity хеширует пароли автоматически при использовании методов UserManager(рисунок 5):



Рисунок 5 – Хеширование пароля

1. Проверка пароля:
   * Identity сравнивает введенный пользователем пароль с сохраненным хешем(рисунок 6):



Рисунок 6 – Сравнение хешей паролей

**Особенности ASP.NET Core Identity**

* Готовая реализация входа/регистрации: Identity предоставляет готовые Razor Pages для обработки входа, регистрации, восстановления пароля и управления учетной записью.
* Гибкость и кастомизация: Все аспекты Identity можно кастомизировать, включая модели пользователей, хранилища данных, и UI.
* Поддержка многофакторной аутентификации (MFA): Identity поддерживает двухфакторную аутентификацию через SMS, электронную почту или приложения-аутентификаторы.

Вывод. В рамках практической работы были реализованы системы аутентификации и авторизации, а так же механизмы обеспечения безопасности данных в разрабатываемом программном средстве.