



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

*Факультет Информационных технологий
Кафедра Информатики и информационных технологий*

направление подготовки
09.03.02 «Информационные системы и технологии»

КУРСОВАЯ РАБОТА

Дисциплина: Back-end

Тема: WEB-сайт “AUTOLUXERY”

Выполнил(а): студент(ка) группы 221-3711

Малыгин Александр Сергеевич/Мусатов Тимофей
Эдуардович
(Фамилия И.О.)

Дата, подпись _____
(Дата) (Подпись)

Проверил: _____
(Фамилия И.О., степень, звание) (Оценка)

Дата, подпись _____
(Дата) (Подпись)

Замечания: _____

Москва

2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ	5
1.1 Инструментальные программные средства	5
1.1.1 Visual Studio 2022 и Visual Code	5
1.1.2 React и Vite	6
1.2 Основы разработки веб-приложений с использованием React и базы данных.....	6
1.2.1 Проектирование структуры	6
1.2.2 Интеграция базы данных	7
1.2.3 Реализация авторизации и безопасности	7
1.2.4 Создание интерфейса	7
1.2.5 Оптимизация и тестирование.....	8
2. ПРАКТИЧЕСКАЯ ЧАСТЬ	9
ЗАКЛЮЧЕНИЕ	15
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	17

ВВЕДЕНИЕ

React —это мощная библиотека для создания пользовательских интерфейсов, активно используемая в веб-разработке и играх. С ее помощью разработчики могут создавать динамичные и отзывчивые интерфейсы, которые адаптируются к различным экранам и устройствам. В игровой индустрии **React** позволяет эффективно управлять состоянием и взаимодействием с пользователем, обеспечивая плавный и быстрый отклик на действия игрока. Его компонентная архитектура способствует легкости в поддержке и расширении проекта, что особенно важно в больших и сложных играх. **React** активно используется в играх для создания меню, панелей инвентаря и других интерфейсных элементов.

Vite — это современный инструмент для сборки и разработки, который значительно ускоряет процесс создания веб-приложений, включая игры и мультимедийные проекты. Благодаря использованию продвинутых технологий, таких как модульная система и горячая перезагрузка, **Vite** позволяет разработчикам быстро и эффективно разрабатывать игровые интерфейсы и динамичные веб-страницы. В игровых проектах это способствует сокращению времени загрузки и улучшению общей производительности, что критически важно для обеспечения комфортного игрового опыта. **Vite** оптимизирует сборку и улучшает совместимость с современными веб-технологиями, позволяя интегрировать различные 3D-движки и визуальные эффекты с минимальными затратами времени.

Главной целью данной курсовой работы является разработка сайта с добавлением базы данных для авторизации пользователей, работы с каталогом продукции и улучшения процесса авторизации через внедрение JWT токенов в куки файлы. В рамках проекта необходимо будет реализовать проверку данных на сервере, а не на фронт-энде, что обеспечит более высокий уровень безопасности. Особое внимание уделяется архитектуре системы, интеграции с базой данных и оптимизации процесса авторизации для удобства пользователей. Для достижения поставленной цели определены следующие задачи:

1. Изучить возможности React для разработки интерфейса сайта и интеграции с базой данных, чтобы обеспечить удобную работу с каталогом продукции и авторизацией пользователей.
2. Провести анализ и подбор подходящих библиотек и инструментов для работы с JWT токенами и куки файлами для улучшенной авторизации.

3. Разработать базовую структуру сайта, учитывая требования к безопасности, удобству интерфейса и взаимодействию с базой данных.
4. Реализовать регистрацию и авторизацию пользователей с использованием JWT токенов, обеспечив безопасность хранения данных в куки файлах.
5. Провести проектирование и создание структуры базы данных для хранения информации о пользователях и продукции.
6. Разработать логику работы с каталогом продукции, включая добавление, удаление и обновление товаров в базе данных.
7. Реализовать проверку данных на сервере, обеспечив безопасность авторизации и предотвращение возможных атак.
8. Провести тестирование и оптимизацию всех элементов сайта, включая работу с базой данных и безопасность авторизации, а также подготовить финальную документацию проекта.

Реализация проекта позволяет не только освоить ключевые аспекты разработки веб-приложений, но и применить полученные знания в создании системы авторизации и работы с базой данных. Итоговая работа может быть полезна как для обучения, так и для дальнейшего использования в реальных веб-проектах, где важна безопасность и эффективная работа с пользователями. Процесс разработки станет хорошей основой для изучения современных подходов в создании безопасных и оптимизированных веб-систем и поможет развить навыки, востребованные в быстро развивающейся сфере веб-разработки.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Инструментальные программные средства

В процессе выполнения курсовой работы по разработке сайта с добавлением базы данных для авторизации пользователей, работы с каталогом продукции и улучшения авторизации с использованием JWT токенов применялись следующие ключевые инструменты: Visual Studio 2022, Visual Code, React и Vite. Visual Studio 2022 и Visual Code использовались для разработки и отладки серверной и клиентской части приложения. React и Vite были выбраны как основные средства для создания динамичного интерфейса сайта с эффективной работой с данными и безопасной авторизацией пользователей.

Эти программные средства обеспечили все необходимые функции для реализации поставленных задач и создания качественного веб-приложения.

1.1.1 Visual Studio 2022 и Visual Code

Visual Studio 2022 и Visual Code — это два мощных средства разработки, которые активно используются в создании современных веб-приложений. Visual Studio 2022 является полнофункциональной IDE, которая предоставляет мощные инструменты для разработки на различных языках программирования, включая C# и JavaScript, что удобно для работы с серверной частью проекта. Она включает в себя функции отладки, тестирования и управления версиями, что значительно ускоряет процесс разработки и улучшает качество кода.

Visual Code — это легкий, но мощный редактор, который идеально подходит для написания клиентского кода и работы с современными фреймворками, такими как React. Он поддерживает большое количество расширений, которые помогают в работе с JavaScript, JSX, а также предлагают интеграцию с системами контроля версий и инструментами сборки. В данном проекте Visual Studio 2022 использовалась для создания и настройки серверной части приложения, а Visual Code — для разработки и отладки клиентского интерфейса, написанного с использованием React.

1.1.2 React и Vite

React — это популярная библиотека JavaScript, используемая для создания динамичных пользовательских интерфейсов. В данном проекте React был выбран для создания компонента авторизации, работы с каталогом продукции и других интерактивных элементов сайта. Благодаря своей компонентной архитектуре React позволяет эффективно управлять состоянием приложения и обновлением данных на странице, что критично для динамичного взаимодействия с пользователем и сервером.

Vite — это современный инструмент для сборки, который значительно ускоряет процесс разработки благодаря использованию горячей перезагрузки и модульной системы. Он был использован в проекте для настройки сборки и обеспечения быстрого отображения изменений при работе с React. Vite помог оптимизировать процесс разработки и значительно улучшить производительность проекта, что также включало создание безопасной системы авторизации с использованием JWT токенов.

С использованием данных инструментов был создан функциональный веб-сайт с надежной авторизацией, интеграцией с базой данных и удобным пользовательским интерфейсом, что стало основой успешной реализации курсовой работы.

1.2 Основы разработки веб-приложений с использованием React и базы данных

1.2.1 Проектирование структуры

Проектирование структуры веб-приложения представляет собой начальный этап разработки, на котором формируются основные компоненты и логика работы приложения. На этом этапе используются базовые элементы, такие как страницы, компоненты и роутеры, чтобы задать общую структуру и взаимодействие между частями сайта. Основная цель этого этапа — выявить и

устранить возможные ошибки в архитектуре и взаимодействии, что обеспечит стабильную работу на более поздних стадиях разработки.

1.2.2 Интеграция базы данных

Процесс интеграции базы данных заключается в подключении системы хранения данных, которая будет использоваться для хранения информации о пользователях и продукции. На этом этапе проектируются таблицы и связи между ними, чтобы обеспечить корректную работу с данными. Важно создать правильные запросы и механизмы обработки данных для обеспечения эффективной и безопасной работы с базой данных, что является основой функциональности сайта.

1.2.3 Реализация авторизации и безопасности

Авторизация и безопасность — это ключевые элементы веб-приложения, которые обеспечивают защиту данных пользователей. На этом этапе реализуется система регистрации, авторизации и проверки данных через JWT токены. Для этого используется куки-файлы, которые позволяют хранить токены безопасности на клиентской стороне, а проверку данных выполняет сервер. Важно на этом этапе интегрировать механизм проверки подлинности данных, чтобы повысить уровень безопасности.

1.2.4 Создание интерфейса

Создание интерфейса сайта включает разработку всех интерактивных элементов, таких как формы регистрации и входа, каталог продукции, а также различные кнопки и панели. На этом этапе используется библиотека React, которая позволяет создавать динамичные и отзывчивые интерфейсы. Основной задачей является создание удобного пользовательского опыта, который позволит легко взаимодействовать с сайтом, управлять данными и проходить авторизацию.

1.2.5 Оптимизация и тестирование

Оптимизация завершает процесс разработки, обеспечивая максимальную производительность и стабильность работы сайта. Для этого проводятся тестирования, исправляются баги и оптимизируется работа с сервером и базой данных. Также проводится тестирование системы безопасности для предотвращения утечек данных. Результатом работы является готовый и стабильный веб-сайт с интегрированным функционалом авторизации и работы с базой данных, который готов к использованию в реальных проектах.

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

Практическая часть курсовой работы включает в себя создание веб-приложения с использованием технологий C#, ASP.NET Core, Entity Framework, а также внедрение системы авторизации через JWT токены. В данном разделе описан процесс разработки и реализации ключевых функций веб-приложения с примерами кода.

1. Настройка базы данных

В этой части работы используется Entity Framework для создания базы данных и работы с таблицами клиентов и продукции. Настроена строка подключения и контекст базы данных для работы с таблицами.

Пример настройки базы данных:

```
string connection =  
"Server=(localdb)\\mssqllocaldb;Database=applicationdb;Trusted_Connection=True;";  
builder.Services.AddDbContext<ApplicationContext>(options => options.UseSqlServer(connection));
```

В этом коде создается подключение к базе данных applicationdb с использованием SQL Server. Контекст базы данных ApplicationContext будет работать с клиентами и продукцией, предоставляя необходимые методы для выполнения CRUD операций.

Пример модели клиента:

```
public class Client  
{  
    public int Id { get; set; }  
    public string Name { get; set; }  
    public int Age { get; set; }  
}
```

2. Разработка системы авторизации

Система авторизации реализована с использованием JWT токенов. При успешной авторизации пользователю генерируется токен, который затем используется для доступа к защищенным ресурсам.

Пример создания JWT токена:

```
string GenerateJwtToken(int clientId, string name)  
{
```

```

var claims = new[]
{
    new Claim(ClaimTypes.Name, name),
    new Claim("ClientId", clientId.ToString())
};

var credentials = new SigningCredentials(new SymmetricSecurityKey(key),
SecurityAlgorithms.HmacSha256Signature);

var token = new JwtSecurityToken(
    issuer: "mydomain.com",
    audience: "mydomain.com",
    claims: claims,
    expires: DateTime.Now.AddMinutes(1),
    signingCredentials: credentials
);

return new JwtSecurityTokenHandler().WriteToken(token);
}

```

Этот метод генерирует JWT токен с именем пользователя и его идентификатором, используя секретный ключ для подписи токена. Токен истекает через 1 минуту.

Пример авторизации с проверкой данных:

```

app.MapPost("/api/auth/login", async (Client loginData, ApplicationContext db) =>
{
    var client = await db.Clients.FirstOrDefaultAsync(c => c.Name == loginData.Name && c.Age.ToString() ==
loginData.Age.ToString());

    if (client == null)
    {
        return Results.Json(new { message = "Invalid username or password" }, statusCode: 401);
    }

    var token = GenerateJwtToken(client.Id, client.Name);
}

```

```
return Results.Json(new { message = "Login successful", token });  
});
```

Здесь при отправке запроса на `/api/auth/login` проверяются имя и возраст пользователя. Если данные корректны, генерируется JWT токен, который возвращается клиенту.

3. Реализация API для работы с клиентами и продукцией

Реализованы эндпоинты для работы с клиентами и продукцией. Для каждого маршрута предусмотрена проверка авторизации через JWT токен.

Пример эндпоинта для получения списка клиентов:

```
app.MapGet("/api/clients", async (ApplicationContext db) =>  
{  
    var clients = await db.Clients.ToListAsync();  
    return Results.Json(clients);  
});
```

Пример добавления нового клиента:

```
app.MapPost("/api/clients", async (Client client, ApplicationContext db) =>  
{  
    await db.Clients.AddAsync(client);  
    await db.SaveChangesAsync();  
    return Results.Json(client);  
});
```

Эндпоинт добавления нового клиента принимает данные клиента и сохраняет их в базе данных.

Пример эндпоинта для работы с продукцией:

```
app.MapGet("/api/products", async (ApplicationContext db) =>  
{  
    var products = await db.Products.ToListAsync();  
    return Results.Json(products);  
});
```

4. Интеграция безопасности через JWT токены

Для защиты эндпоинтов добавлена система аутентификации и авторизации через JWT токены. Все защищенные маршруты требуют наличия валидного токена в заголовке запроса.

Пример использования авторизации для защищенных маршрутов:

```
app.UseAuthentication();
app.UseAuthorization();

Добавление атрибута авторизации для конкретных маршрутов:
app.MapGet("/api/clients", [Authorize] async (ApplicationContext db) =>
{
    var clients = await db.Clients.ToListAsync();
    return Results.Json(clients);
});
```

5. Оптимизация и тестирование

Для оптимизации работы приложения и обеспечения безопасности используется CORS для доступа с различных доменов и обработка ошибок для предотвращения сбоев.

Пример настройки CORS:

```
builder.Services.AddCors(options =>
{
    options.AddPolicy(MyAllowSpecificOrigins,
        policy =>
        {
            policy.WithOrigins("http://localhost:5173")
                .AllowAnyHeader()
                .AllowAnyMethod();
        });
});
```

Эта настройка позволяет разрешить доступ к API только с домена `http://localhost:5173`, что повышает безопасность приложения.

Заключение части Back-end

Практическая часть курсовой работы включает создание полноценного веб-приложения с авторизацией через JWT токены, взаимодействием с базой

данных и API для работы с клиентами и продукцией. Реализованная система безопасности гарантирует защищенный доступ к данным, а использование современных технологий, таких как Entity Framework и JWT, делает приложение надежным и безопасным.

6. Фронтенд: Реализация страницы входа и управления сессией

В этой части работы описана реализация страницы входа, регистрации и управления сессией с использованием React и некоторых библиотек для UI и HTTP-запросов. Используется библиотека react-bootstrap для создания интерфейса и axios для взаимодействия с API.

6.1 Структура страницы входа

Страница входа содержит форму с полями для ввода имени пользователя и пароля. Кнопки позволяют пользователю войти в систему, зарегистрироваться или выйти из нее. Стилизация страницы выполнена с использованием react-bootstrap, а также установлена фоновая картинка.

6.2 Управление состоянием пользователя

Используются хуки React (useState и useEffect) для управления состоянием, включая имя пользователя, пароль, ошибку и статус авторизации. Когда пользователь успешно входит, сохраняется токен в cookies.

6.3 Логика входа

При отправке формы происходит отправка POST-запроса с именем и паролем пользователя на сервер. В случае успешной авторизации сервер возвращает токен, который сохраняется в cookies с помощью js-cookie. Пользователь перенаправляется на страницу после успешного входа.

6.4 Логика выхода

При выходе из системы токен удаляется из cookies, а состояние пользователя обновляется. Пользователь перенаправляется на страницу входа.

6.5 Перенаправление и управление сессией

При загрузке страницы проверяется наличие токена в cookies. Если токен присутствует, пользователь остается авторизованным, в противном случае происходит перенаправление на страницу входа.

6.6 Взаимодействие с сервером

Для отправки запросов на сервер используется библиотека axios. Все запросы защищены JWT токеном, который передается в заголовке авторизации.

Пример кода:

```
const handleLogin = async () => {  
  try {  
    const response = await axios.post("https://localhost:7039/api/auth/login", {  
      name: username, age: password });  
    if (response.status === 200) {  
      Cookies.set("accessToken", response.data.token, { expires: 1, secure: true,  
        sameSite: "Strict" });  
      window.location.href = "/home";  
    }  
  } catch (err) {  
    setError("Incorrect username or password");  
  }  
};  
  
useEffect(() => {  
  const token = Cookies.get("accessToken");  
  if (!token) handleLogout();  
  else setLoggedIn(true);  
}, []);
```

Заключение front-end

Этот раздел описывает, как на фронтенде реализована страница входа и управления сессией пользователя. Использование JWT токенов и cookies позволяет обеспечить безопасное взаимодействие с сервером.

ЗАКЛЮЧЕНИЕ

В результате данной курсовой работы был выполнен комплексный подход к разработке веб-приложения с использованием технологий фронтенда и бэкенда, ориентированных на взаимодействие с базой данных и систему аутентификации. Основным инструментом для создания серверной части стал .NET, а для фронтенда использовались React и связанные с ним библиотеки.

На серверной стороне была реализована система аутентификации с использованием JWT (JSON Web Token) для обеспечения безопасности пользовательских данных. Разработанные API-методы позволили обрабатывать запросы на регистрацию, авторизацию, а также управлять данными о пользователях и продуктах через CRUD-операции. Важным этапом работы было создание безопасного механизма аутентификации и авторизации, который включает в себя проверку прав доступа и хранение токенов в cookies.

На фронтенде была разработана страница входа, которая позволяет пользователю ввести свои данные и получить токен для доступа к защищённым ресурсам. Взаимодействие с сервером происходило через API с помощью библиотеки axios, которая обеспечивала отправку запросов и обработку ответов. Важным аспектом разработки был контроль состояния авторизации, что позволило реализовать логику для работы с сессией, а также автоматическую проверку наличия токена при загрузке страницы.

Особое внимание было уделено созданию удобного и интуитивно понятного пользовательского интерфейса, использующего компоненты из библиотеки react-bootstrap, что обеспечивало хороший внешний вид и отзывчивость страницы. Для управления состоянием приложения использовались хуки React, что позволило эффективно обновлять интерфейс при изменении данных.

В результате был создан рабочий прототип веб-приложения, где реализована полноценная аутентификация с хранением токенов и возможность работы с базой данных. Процесс разработки позволил получить ценные

практические знания в области веб-разработки, а также в работе с серверной частью и безопасности приложений. Полученные навыки станут полезными для дальнейшей разработки более сложных и масштабных систем в будущем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Документация React. Официальная документация по React.
<https://ru.reactjs.org/>.

MDN Web Docs (Русский перевод). Документация по Fetch API.
https://developer.mozilla.org/ru/docs/Web/API/Fetch_API.

JavaScript.info. Полный курс по JavaScript на русском языке.
<https://javascript.info/>.

HTML Academy. Онлайн-курсы по веб-разработке, включая React.
<https://htmlacademy.ru/>.

RuStack Overflow. Вопросы и ответы по разработке на React и других технологиях. <https://ru.stackoverflow.com/>.

W3Schools (Русская версия). Веб-ресурс с обучением HTML, CSS, JavaScript, React и других технологий. <https://www.w3schools.com/>.

CSS Tricks (Переводы). Статьи и советы по работе с CSS и Flexbox.
<https://css-tricks.com/>.

Хабр. Статьи и публикации по различным темам разработки.
<https://habr.com/ru/>.

Code.ru. Платформа для изучения программирования и веб-разработки. <https://code.ru/>.

Stackoverflow Платформа для изучения программирования и веб-разработки <https://stackoverflow.com>

metanit Платформа для изучения программирования и веб-разработки
<https://metanit.com>