Создание сайта на стеке MERN

Выполнил ученик 9А класса Кудашев Сергей

Руководитель Белянкина Анна Михайловна

Оглавление

[Описание проекта 3](#_Toc193226602)

[Цели проекта 3](#_Toc193226603)

[Технологический стек 4](#_Toc193226604)

[Требования 4](#_Toc193226605)

[Клонирование репозитория 4](#_Toc193226606)

[Установка зависимостей 4](#_Toc193226607)

# Описание проекта

**MERN Blog** — это веб-приложение, разработанное с использованием стека MERN (MongoDB, Express.js, React, Node.js). Оно предоставляет пользователям возможность зарегистрироваться, авторизоваться и публиковать свои статьи в блоге.

Блоговая платформа позволяет каждому пользователю создавать уникальный контент, редактировать и удалять свои публикации, а также просматривать посты других пользователей. Приложение имеет удобный интерфейс, разработанный с использованием современных веб-технологий, и поддерживает безопасность на уровне аутентификации и защиты данных.

# Цели проекта

Основная цель данного проекта — разработка удобного, безопасного и интуитивно понятного веб-приложения, позволяющего пользователям вести собственные блоги. Среди основных задач:

* Обеспечение удобного пользовательского интерфейса для работы с контентом.
* Создание механизма регистрации и авторизации пользователей.
* Разработка безопасной системы аутентификации с использованием токенов доступа (JWT).
* Возможность публикации, редактирования и удаления постов.
* Обеспечение взаимодействия клиентской и серверной частей приложения через API.
* Гибкость и масштабируемость проекта с возможностью расширения функционала.

# Технологический стек

Для реализации данного проекта используется современный технологический стек MERN, который включает в себя:

* **Frontend**: React.js — современный JavaScript-фреймворк для создания пользовательских интерфейсов.
* **Backend**: Node.js и Express.js — платформа и фреймворк для обработки запросов, работы с бизнес-логикой и обеспечением API.
* **База данных**: MongoDB — документно-ориентированная NoSQL база данных, позволяющая гибко управлять данными.
* **Дополнительные технологии**:
  + **JWT (JSON Web Token)** — для аутентификации и защиты данных пользователей.
  + **bcrypt** — для безопасного хеширования паролей.
  + **dotenv** — для работы с переменными окружения и конфигурацией приложения.
  + **Mongoose** — ORM для MongoDB, упрощающая работу с базой данных.

# Требования

Для успешного запуска и работы проекта необходимо наличие следующих инструментов:

* **Node.js** — среда выполнения JavaScript (рекомендуемая версия 16+).
* **MongoDB** — база данных, развернутая локально или в облаке (например, MongoDB Atlas).
* **NPM или Yarn** — пакетный менеджер для установки зависимостей.
* **Git** — система контроля версий для работы с репозиторием проекта.

# Клонирование репозитория

Перед началом работы необходимо склонировать репозиторий проекта с помощью системы контроля версий (Git). Это позволит получить актуальную версию кода и развернуть её локально.

Для клонирования репозитория выполните следующую команду:

git clone https://github.com/sergeik1970/blog-with-register.git

После клонирования перейдите в корневую папку проекта:

cd blog-with-register

# Установка зависимостей

После клонирования репозитория необходимо установить все необходимые зависимости для клиентской и серверной частей проекта.

Установите зависимости для серверной части:

cd server

npm install

Затем установите зависимости для клиентской части:

cd client

npm install

Эти команды автоматически загрузят все необходимые библиотеки и модули, указанные в файлах package.json.

# Настройка переменных окружения

Для работы приложения требуются конфигурационные параметры, такие как строка подключения к базе данных и секретные ключи для аутентификации. Эти данные хранятся в файле переменных окружения, который необходимо создать перед запуском.

Создайте файл .env в папке server/ и добавьте в него следующие настройки:

PORT=5000

MONGO\_URI=my\_mongodb\_connection\_string

JWT\_SECRET=my\_jwt\_secret

Этот файл не должен попадать в репозиторий, поэтому рекомендуется добавить его в .gitignore.

# Запуск приложения

После успешной установки всех зависимостей и настройки конфигурации можно запустить сервер и клиентскую часть проекта. Сервер будет обрабатывать запросы к API, а клиентская часть обеспечит взаимодействие пользователя с системой.

После успешного запуска сервер будет слушать входящие запросы на указанном порту (по умолчанию 5000), а клиент будет доступен по адресу <http://localhost:3000/>

**Деплой на Render**

Для удобного развертывания проекта используется сервис **Render**. Бэкенд и фронтенд проекта загружены на Render, что позволяет пользователям взаимодействовать с приложением без необходимости локального развертывания.

* **Frontend** выложен на Render и доступен по адресу: [URL\_Frontend]
* **Backend** также размещен на Render и доступен по адресу: [URL\_Backend]

**Общая структура**

Проект построен на архитектуре клиент-сервер, где фронтенд и бэкенд взаимодействуют через API.

* **Клиентская часть (Frontend)** — отвечает за отображение пользовательского интерфейса, обработку событий и отправку запросов к серверу.
* **Серверная часть (Backend)** — принимает запросы от клиента, обрабатывает данные, выполняет операции с базой данных и отправляет результаты обратно клиенту.
* **База данных (MongoDB)** — хранит данные о пользователях, публикациях и других элементах системы.

**Организация кода**

Код проекта разделен на логические модули, что упрощает его поддержку и дальнейшее развитие.

* **Клиентская часть**
  + Компоненты пользовательского интерфейса.
  + Маршрутизация страниц.
  + Хранение состояния приложения (если используется Redux).
* **Серверная часть**
  + Маршруты API.
  + Контроллеры для обработки запросов.
  + Модели данных для MongoDB.
  + Middleware (промежуточные обработчики).
* **Конфигурация проекта**
  + Настройки подключения к базе данных.
  + Переменные окружения.
  + Защита данных пользователей.