**Министерство образования и науки Кыргызской Республики**

**Кыргызский Государственный Технический Университет**

**им. И. Раззакова**

**Институт электроники и телекоммуникаций**

**кафедра «*Информационные системы и технологии***

***им. акад. А. Жайнакова»***

**направления: 710200 Информационные системы и технологии**

**КУРСОВОЙ ПРОЕКТ**

**по дисциплине**

**«Проектирование и разработка веб-приложений BACKEND»**

**на тему «Веб-приложение для ведения дневника»**

**Разработал: студент гр. ИСТ(б)-1-23 \_\_\_\_\_\_\_\_\_\_\_\_ У.М.Асанкадырова**

**(подпись)**

**Проверила: к.ф.-м.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Н.Аманкулова**

**(подпись)**

**Бишкек-2025**

**ЗАДАНИЯ**

на выполнение курсового проекта по дисциплине  
«Проектирование и разработка веб-приложений (Backend)»

**Студент:** Асанкадырова Махабат Урматовна   
**Группа:** ИСТ(б)-1-23

**Тема курсового проекта:** «**Веб-приложение ведение дневника»**

**Исходные данные для проектирования:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* Требования к функциональности;
* Технические условия;
* Ограничения по технологиям и инструментам;
* Дополнительные условия (при наличии).

**Цель курсового проекта:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  
(формулируется чётко, например: разработка веб-приложения для онлайн-бронирования услуг.)

**Задачи проекта:**  
(перечислить ключевые задачи, например:)

* Проектирование архитектуры веб-приложения;
* Разработка клиентской части на JavaScript/React;
* Разработка серверной части на Node.js/Express;
* Реализация взаимодействия с базой данных;
* Организация аутентификации пользователей (если требуется).

**Требования к разрабатываемому приложению:**  
(указать минимальный функционал, например:)

* Регистрация и вход пользователя;
* Просмотр, создание, редактирование, удаление данных;
* Отправка запросов к серверу через API;
* Корректная обработка ошибок.

**Используемые технологии:**

* Frontend: HTML, CSS, JavaScript (или React, Vue и др.);
* Backend: Node.js, Express.js;
* База данных: MongoDB, PostgreSQL и др.

**График выполнения проекта:**

| Этап | Содержание работ | Срок выполнения |
| --- | --- | --- |
| 1 | Аналитический этап (исследование, постановка задачи) | \_04/ 02/ 2025 |
| 2 | Проектировочный этап (архитектура, выбор технологий) | 15/ 03 / 2025 |
| 3 | Разработка приложения | 20/ 04 / 2025 |
| 4 | Тестирование и отладка | \_15 / 05 / 2025\_\_ |
| 5 | Подготовка пояснительной записки | 25/ 05 / 2025 |
| 6 | Защита проекта | 28/ 05 / 2025 |

**Руководитель курсового проекта:**  
Фамилия, инициалы: Аманкулова Н.А.   
Должность: Заведующая кафедрой «Информационные системы и технологии им. академика А.Жайнкова»

**Подписи:**

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /ФИО/ 27 «май**»  2025**г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /ФИО/ 27 «май**» 2025**г.

**Содержание**

[Введение 4](#_Toc199838621)

[ГЛАВА 1. ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. 5](#_Toc199838622)

**[1.1 Анализ предметной области](#_Toc199838623)** [5](#_Toc199838623)

[**1.2 Требования к функциональности и дизайну веб-приложения**. 6](#_Toc199838624)

[**1.3 Проектирование структуры и интерфейса веб-сайта** 7](#_Toc199838625)

[**1.4. Выбор технологий и инструментов разработки** 9](#_Toc199838626)

[ГЛАВА 2. РАЗРАБОТКА И ТЕСТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ СОСТАВЛЕНИЯ ВЕДЕНИЯ ДНЕВНИКА 11](#_Toc199838627)

[**2.1. Реализация фронтенд-части проекта** 11](#_Toc199838628)

[**2.1.1. Интерфейс пользователя** 12](#_Toc199838629)

[**2.2. Структура backend-проекта** 20](#_Toc199838630)

[**2.2.1. Модели базы данных** 24](#_Toc199838631)

[**2.2.2. Реализация API** 26](#_Toc199838632)

[**2.2.3. Авторизация и безопасность** 29](#_Toc199838633)

[**2.2.4. Тестирование API** 30](#_Toc199838634)

[Заключение 32](#_Toc199838635)

[Список используемой литературы 33](#_Toc199838636)

# **Введение**

В условиях стремительного развития информационных технологий и цифровизации различных сфер жизни всё большую популярность приобретают веб-приложения, которые позволяют пользователям вести личные записи, планировать дела, хранить информацию и взаимодействовать с данными из любой точки мира. Одним из таких приложений является электронный дневник. Веб-приложение для ведения дневника представляет собой удобный инструмент для организации личной информации и ведения записей в онлайн-режиме.

Целью данного курсового проекта является разработка и реализация веб-приложения «Дневник», позволяющего пользователю создавать, просматривать, редактировать и удалять записи, добавлять теги, а также работать с интерфейсом, предоставляющим удобную навигацию и безопасность данных.

Проект реализован с использованием современных веб-технологий: Node.js, Express.js, MongoDB, React.js, HTML, CSS. В качестве базы данных используется MongoDB. Также применяются технологии JWT-аутентификации и контекст для управления пользователями на клиентской стороне.

**Задачи проекта:**

* Проанализировать требования к функциональности приложения;
* Спроектировать архитектуру приложения с разделением на frontend и backend;
* Реализовать пользовательский интерфейс с использованием React;
* Разработать серверную часть с REST API на Node.js и Express;
* Настроить взаимодействие с базой данных MongoDB через mongoosh;
* Реализовать авторизацию пользователей (JWT);
* Обеспечить безопасность и валидацию данных;
* Провести тестирование приложения;
* Оформить пояснительную записку по проекту.

# **ГЛАВА 1. ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1 Анализ предметной области**

Дневник представляет собой способ хранения личных записей, который позволяет пользователю фиксировать важные события, мысли, задачи и даты. С развитием цифровых технологий традиционные бумажные дневники постепенно утрачивают свою популярность, уступая место электронным аналогам. Цифровые дневники обладают рядом значительных преимуществ. Во-первых, они обеспечивают доступ к записям с любого устройства — компьютера, смартфона или планшета, что делает их использование особенно удобным в условиях современной мобильности. Во-вторых, такие приложения позволяют надёжно хранить данные за счёт автоматического резервного копирования, что исключает риск потери информации. Кроме того, электронные дневники предоставляют возможность быстрого поиска и фильтрации записей по ключевым словам, дате или тегам, а также удобную навигацию по интерфейсу. Использование тегов позволяет эффективно структурировать и категоризировать записи. Важным преимуществом является также поддержка нескольких пользователей, что делает возможным ведение дневника не только индивидуально, но и в рамках семейных или групповых целей. Всё это обуславливает актуальность создания современного веб-приложения «Дневник», которое должно сочетать в себе интуитивно понятный интерфейс, простоту использования, гибкость в работе с записями и безопасность хранения данных. Таким образом, основная задача разрабатываемого приложения — предоставить пользователю удобный, надёжный и функциональный инструмент для ежедневного ведения личных записей в цифровом формате [6].

## **1.2 Требования к функциональности и дизайну веб-приложения**.

Веб-приложение «MyDiary» предназначено для создания, хранения и управления личными дневниковыми записями пользователей. Оно должно обеспечивать удобный интерфейс, надёжную систему хранения данных и высокий уровень безопасности. Основными функциональными возможностями приложения являются: регистрация и вход пользователя, возможность создания новой учетной записи, авторизация через логин и пароль, хранение пользовательских данных в базе данных, а также использование JWT-токенов или сессий для защиты пользовательских сессий.

Работа с дневниковыми записями включает создание новых записей с заголовком, основным текстом, тегами и датой, просмотр всех записей текущего пользователя, отображение записей в виде карточек с краткой информацией (заголовок, дата, часть текста), возможность полного просмотра отдельной записи, редактирование и удаление существующих записей.

Дополнительно реализуется функциональность поиска и фильтрации: пользователь может осуществлять поиск по заголовку или содержимому записи, а также фильтрацию по тегам. Для работы с тегами предусмотрена возможность добавления одного или нескольких тегов к каждой записи, отображение списка всех доступных тегов и использование кликабельных тегов для быстрого доступа к связанным записям.

Важной составляющей приложения является безопасность. Предусмотрена валидация данных на стороне клиента и сервера, защита от несанкционированного доступа и обязательное шифрование паролей. В многопользовательском режиме каждый пользователь видит только свои записи, что обеспечивается индивидуальной авторизацией и персонализацией данных. Также реализована возможность выхода из системы с последующим удалением текущей сессии [4].

## **1.3 Проектирование структуры и интерфейса веб-сайта**

Проектирование структуры и интерфейса веб-приложения «MyDiary» является важным этапом разработки, направленным на обеспечение удобства, логичности и визуальной привлекательности системы. Основной задачей данного этапа является создание интуитивно понятного пользовательского интерфейса и логичной навигации между страницами приложения.

Веб-приложение «MyDiary» реализовано по принципу одностраничного приложения (SPA), что позволяет загружать и отображать контент без полной перезагрузки страницы. Это делает работу приложения более быстрой и плавной. Для маршрутизации между различными разделами используется клиентская навигация.

Основные страницы приложения включают: страницу авторизации, где пользователь вводит логин и пароль, а также может перейти к регистрации нового аккаунта; страницу регистрации с полями для имени пользователя, пароля и его подтверждения; главную страницу, на которой отображаются все записи текущего пользователя в виде карточек с заголовком, датой и фрагментом текста. Также здесь присутствует кнопка создания новой записи и возможность фильтрации по тегам.

Для создания новой записи используется отдельная страница с формой ввода заголовка, текста и тегов, а также кнопками «Сохранить» и «Отмена». Редактирование записи происходит на аналогичной странице, но форма уже содержит предзаполненные текущие данные. Кроме того, реализован раздел тегов, в котором отображаются все использованные пользователем теги, и предусмотрена возможность фильтрации записей по выбранному тегу.

Присутствует также страница настроек, которая на текущем этапе может содержать базовую информацию или быть заглушкой, однако в перспективе здесь планируется реализовать функциональность смены пароля и имени пользователя. Выход из аккаунта осуществляется путём очистки токена или сессии и перенаправления пользователя обратно на страницу входа [10].

Для удобной и быстрой навигации в приложении реализовано боковое меню, которое отображается слева и доступно на всех страницах после входа в систему. В меню находятся кнопки для перехода на главную страницу, создания новой записи, просмотра тегов, открытия настроек и выхода из аккаунта.

Интерфейс приложения оформлен в современном минималистичном стиле. Вёрстка выполнена с использованием стандартных HTML-элементов, стилизованных с помощью CSS. В дальнейшем возможно подключение CSS-фреймворков (например, Tailwind CSS или Bootstrap) для упрощения адаптации интерфейса под различные устройства. Визуально каждая запись представлена в виде карточки, содержащей заголовок, дату, текст и кнопки действий (редактировать, удалить). Для подтверждения действий, таких как удаление записи, используются модальные окна. Формы ввода данных снабжены валидацией. Интерфейс адаптивен и корректно отображается на разных устройствах, от мобильных до десктопных.

Цветовая палитра интерфейса выдержана в светлых и пастельных тонах, что создаёт атмосферу лёгкости и уюта. Заголовки выделены полужирным шрифтом, основной текст имеет стандартную насыщенность. Используются современные читаемые шрифты, такие как Roboto или Open Sans [9].

## **1.4. Выбор технологий и инструментов разработки**

При создании веб-приложения «MyDiary» был осуществлён обоснованный выбор современных технологий и инструментов, обеспечивающих удобную разработку, масштабируемость, безопасность и высокую производительность. Выбор технологий основывался на актуальных практиках веб-разработки, а также на личном опыте и целях проекта. Важным критерием также стало соотношение простоты использования и функциональности, что позволило сосредоточиться на реализации ключевых функций приложения, не тратя излишние ресурсы на изучение сложных или малораспространённых решений.

На стороне **frontend** для реализации пользовательского интерфейса был выбран **React** — популярная библиотека JavaScript, позволяющая создавать динамичные одностраничные приложения (SPA). React обеспечивает высокую производительность за счёт виртуального DOM, поддерживает компонентный подход и позволяет эффективно управлять состоянием интерфейса. Для организации маршрутизации в приложении используется библиотека **React Router**, обеспечивающая переход между страницами без перезагрузки. Стилизация интерфейса реализована с помощью **кастомного CSS**, что позволяет точно контролировать внешний вид компонентов и адаптировать их под конкретный дизайн. При необходимости в будущем возможно подключение CSS-фреймворков [6].

На стороне **backend** используется **Node.js** в связке с **Express.js** — лёгким и гибким фреймворком, позволяющим эффективно разрабатывать серверную часть приложения. Express обеспечивает маршрутизацию, обработку HTTP-запросов и реализацию REST API. Это позволяет удобно взаимодействовать с фронтендом, обрабатывать пользовательские действия, управлять сессиями и защищать маршруты [1].

В качестве базы данных используется **MongoDB** — современная документно-ориентированная база данных NoSQL, идеально подходящая для приложений, в которых структура данных может быть гибкой и изменяемой. MongoDB хранит данные в формате JSON-подобных документов (BSON), что удобно при работе с JavaScript. Для взаимодействия с базой данных применяется библиотека **Mongoose**, которая предоставляет средства для моделирования данных, их валидации, создания связей между коллекциями и упрощения CRUD-операций.

MongoDB обеспечивает горизонтальное масштабирование, высокую производительность и быструю обработку запросов. Это особенно важно для приложений, где пользователи могут создавать большое количество записей, тегов и взаимодействовать с ними в реальном времени. Гибкость MongoDB позволяет легко вносить изменения в структуру данных без необходимости пересоздания таблиц или сложных миграций [2].

Для обеспечения безопасности используется **аутентификация с помощью JSON Web Token (JWT)** — надёжный способ идентификации пользователей и защиты маршрутов от неавторизованного доступа. Пароли пользователей хранятся в зашифрованном виде с использованием **bcrypt**, что предотвращает возможность их утечки в случае компрометации базы данных.

Для тестирования API и отладки запросов активно используется инструмент **Postman**, позволяющий вручную отправлять HTTP-запросы к серверу, просматривать ответы, анализировать заголовки и структуру данных. Это помогает выявлять ошибки и быстро реагировать на изменения во время разработки [5].

Проект разделён на логические модули: frontend, backend и база данных, что соответствует современным архитектурным принципам и упрощает сопровождение кода. Также соблюдаются стандарты REST, что делает API логичным, предсказуемым и легко расширяемым [2].

# **ГЛАВА 2. РАЗРАБОТКА И ТЕСТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ СОСТАВЛЕНИЯ ВЕДЕНИЯ ДНЕВНИКА**

## **2.1. Реализация фронтенд-части проекта**

Фронтенд-часть веб-приложения *«*MyDiar*y»* разработана с использованием библиотеки **React** и представляет собой одностраничное приложение (SPA — Single Page Application), обеспечивающее быструю и интуитивно понятную навигацию без перезагрузки страницы. Все компоненты интерфейса разработаны с нуля, без использования CSS-фреймворков, с акцентом на чистый дизайн и удобство пользователя [6].

**Структура проекта**

Фронтенд проекта расположен в директории frontend и имеет следующую структуру:

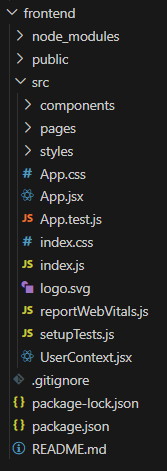


Рис.2.1.1 структура папок фронтенд часть.

## **2.1.1. Интерфейс пользователя**

Интерфейс веб-приложения «MyDiary» разработан с учётом принципов удобства, минимализма и адаптивности. Он состоит из нескольких ключевых элементов, обеспечивающих простую и логичную навигацию, а также быстрый доступ к основным функциям приложения.

**Страница входа**

Первым экраном, который видит пользователь, является **страница входа**. Она содержит форму с полем для ввода имени пользователя и кнопкой «Войти». При вводе имени (например, "Введите ваше имя") приложение проверяет, существует ли такой пользователь. Если имя вводится впервые — создаётся новый пользователь, если повторно — восстанавливаются его записи.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рис.2.1.2. Страница «Добро пожаловать в дневник».

На этом экране пользователь приветствуется по имени, которое он ввёл, и перенаправляется на главную страницу с записями.

**Боковое меню (Sidebar)**

Слева на всех страницах отображается **боковая панель навигации (Sidebar)**. Она содержит следующие интерактивные кнопки:

**Все записи** — переход к списку всех записей текущего пользователя;

**Создать запись** — открывает форму для добавления новой записи;

**Теги** — просмотр записей с фильтрацией по тегам (в разработке);

**Настройки** — управление внешним видом и данными пользователя;

**Выйти** — выход из системы с очисткой данных пользователя.

Изображение выглядит как текст, снимок экрана, число, дизайн

Автоматически созданное описание

Рис.2.1.1.3. Главная страница.

Отображаются карточки записей, каждая из которых содержит заголовок, текст, дату и список тегов. Все записи принадлежат только текущему пользователю.

**Страница создания записи**

При переходе по маршруту /create открывается форма **создания записи**, которая включает поля:

**Заголовок записи**; **Основной текст**; **Теги** (вводятся через запятую).

После заполнения формы и нажатия кнопки «Создать» запись сохраняется и отображается в общем списке.

Изображение выглядит как текст, снимок экрана, Прямоугольник, число

Автоматически созданное описание

Рис.2.1.1.4.Страница «Создать запись».

Показывает интерфейс добавления новой записи. Обязательные поля — заголовок и текст.

Потом ты создаем запись и она сохраняется в теги

Изображение выглядит как текст, снимок экрана, Шрифт, коричневый

Автоматически созданное описание

Рис.2.1.5. Сохраненная запись в теги

Демонстрирует, как выглядит карточка записи после сохранения. Теги отображаются внизу карточки и могут быть использованы для будущей фильтрации.

**Настройки**

Раздел **настроек** позволяет пользователю:

* изменить **тему интерфейса** — доступны варианты: light (светлая), dark (тёмная), nature (природная);
* **изменить имя пользователя** (например, если была допущена ошибка при входе);
* **удалить все записи** текущего пользователя, при этом данные необратимо удаляются из хранилища.

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Рис.2.1.6 Настройки переключение темы на светлый.

Показывает, как интерфейс выглядит при активной светлой теме.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рис.2.1.7.Настройки переключение на темный.

Демонстрирует внешний вид приложения в тёмной цветовой палитре.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рис.2.1.8. Настройки переключении темы на природный

Интерфейс оформлен в зелёных и естественных тонах, создающих ощущение природы.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рис.2.1.9. Настройки изменить имя пользователя

Пользователь может ввести новое имя и продолжить работу с другим идентификатором

**Адаптивность и стилизация**

Всё приложение имеет адаптивный дизайн, корректно отображается на различных разрешениях экранов — от ноутбуков до мобильных устройств. Для стилизации использован **чистый CSS**, без библиотек. Все стили структурированы по компонентам, что обеспечивает:

* лёгкость поддержки;
* повторное использование классов;
* изоляцию внешнего вида каждого компонента [6].

## **2.1.2. Навигация, маршруты и управление состоянием**

Навигация и маршрутизация играют важную роль в организации пользовательского интерфейса и логики переходов в веб-приложении «MyDiary». Для реализации маршрутов и управления страницами используется библиотека **React Router DOM** (версия 6), что позволяет создавать одностраничное приложение (SPA) без перезагрузки страниц и обеспечивает плавные переходы между компонентами.

**Основные маршруты приложения:**

* / — главная страница, на которой отображаются все записи пользователя в виде карточек;
* /create — страница создания новой записи с формой ввода заголовка, текста и тегов;
* /login — страница входа в систему с формой авторизации;
* /tags, /settings, /logout — вспомогательные маршруты, представленные временными заглушками или простыми страницами с соответствующим сообщением.

Каждый маршрут подключается с помощью компонента <Route /> внутри компонента <Routes> и связан с определённым React-компонентом. Это позволяет отображать нужную страницу без полной перезагрузки приложения. Пример настройки маршрутов:

<Routes>

<Route path="/" element={<HomePage />} />

<Route path="/create" element={<CreateEntryPage />} />

<Route path="/login" element={<LoginPage />} />

<Route path="/tags" element={<TagsPage />} />

<Route path="/settings" element={<SettingsPage />} />

<Route path="/logout" element={<LogoutPage />} />

</Routes>

Кроме маршрутизации, в приложении реализована продвинутая система глобального состояния с использованием **React Context API**. Это позволяет хранить информацию о текущем пользователе и обеспечивать доступ к этим данным из любого компонента без необходимости «пробрасывать» props через множество уровней вложенности. Такой подход упрощает архитектуру приложения и повышает его масштабируемость.

Контекст создаётся в отдельном модуле и оборачивает всё приложение, что гарантирует доступность информации о пользователе в любом разделе, включая страницы с записями, настройками и формами создания. Это также облегчает управление состоянием — например, изменение имени пользователя или выход из системы автоматически отражаются во всех компонентах.

Для обеспечения устойчивости состояния при обновлении страницы используется **localStorage** — стандартное браузерное хранилище. После успешного входа данные о пользователе сериализуются и сохраняются в localStorage, а при загрузке приложения контекст автоматически извлекает и восстанавливает эти данные. Такой механизм повышает удобство использования: пользователь не теряет сессии, а повторный вход не требуется при каждом обновлении страницы или закрытии браузера.

Кроме того, реализация с контекстом и localStorage позволяет легко расширять функционал — например, добавлять поддержку нескольких пользователей, интегрировать аутентификацию или сохранять другие параметры пользователя, такие как настройки темы.

Таким образом, навигация, маршрутизация и управление состоянием в приложении «MyDiary» построены на современных и надёжных инструментах React [8].

## **2.2. Структура backend-проекта**

Backend-часть проекта отвечает за обработку запросов, хранение и управление данными пользователей и их дневниковых записей. Она реализована с использованием **Node.js** и фреймворка **Express.js**, а база данных — **MongoDB** с использованием ODM-библиотеки **Mongoose**. Это обеспечивает гибкую работу с документами, удобное создание REST API и выполнение всех CRUD-операций (создание, чтение, обновление, удаление) [7].

**Цели backend-части**

Основными задачами реализации серверной части являются:

* предоставление REST API для взаимодействия с frontend;
* реализация CRUD-функционала для дневниковых записей;
* защита маршрутов с помощью JWT-токенов;
* валидация и обработка ошибок во входящих данных;
* логическая привязка записей к конкретному пользователю.

**Используемые технологии и библиотеки**

|  |  |
| --- | --- |
| Технология / библиотека | Назначение |
| Node.js | Среда выполнения JavaScript на сервере |
| Express.js | Лёгкий и быстрый фреймворк для создания REST API |
| MongoDB | Документо-ориентированная база данных для хранения записей и пользователей |
| Mongoose | ODM-библиотека для работы с MongoDB в стиле моделей и схем |
| JWT (jsonwebtoken) | Генерация и проверка токенов авторизации |
| express-validator | Валидация входящих данных |
| Postman | Инструмент для тестирования API-запросов |

Использование MongoDB позволяет хранить данные в формате JSON-документов, что хорошо сочетается с архитектурой REST-приложения и делает структуру гибкой и масштабируемой. Благодаря Mongoose обеспечивается валидация схем, связи между коллекциями и простое взаимодействие с базой данных через JavaScript.

Серверная часть проекта организована по модульной структуре, что обеспечивает чистоту, читаемость и масштабируемость кода. Каждый логический компонент проекта вынесен в отдельную папку.

Папка config/

Содержит файл конфигурации подключения к базе данных. В частности, db.js отвечает за установку соединения с MongoDB. Адрес базы данных передаётся через переменные окружения для повышения безопасности и гибкости настройки. Централизованное хранение конфигурации упрощает перенос проекта между различными средами (разработка, тестирование, продакшн).

Папка controllers/

В этой папке размещаются функции, реализующие основную бизнес-логику приложения. Контроллеры принимают входящие запросы, выполняют обработку данных, обращаются к базе через модели и формируют ответы клиенту. Например, authController.js реализует регистрацию и вход пользователей, а entryController.js — работу с дневниковыми записями (создание, чтение, обновление, удаление).

Папка middleware/

Содержит промежуточные обработчики запросов. Один из основных — authMiddleware.js, который проверяет наличие и валидность JWT-токена, обеспечивая защиту закрытых маршрутов. Middleware повышают переиспользуемость кода и изолируют общие проверки от основной логики.

Папка models/

В этой папке определены Mongoose-модели — схемы коллекций MongoDB. Модели описывают структуру документов и включают типы полей, требования к заполнению, связи между сущностями. Например, User.js описывает пользователя с полями name, email, password, а Entry.js — дневниковую запись с полями title, text, tags, createdAt и связью с пользователем. Использование Mongoose упрощает работу с базой и минимизирует необходимость в ручных запросах.

Папка routes/

Определяет REST API для взаимодействия клиента с сервером. Каждый файл маршрутов привязан к определённой сущности. Например, auth.js реализует маршруты для регистрации (/register) и входа (/login), а entries.js — для работы с записями (/entries, /entries/:id). Маршруты направляют запросы к соответствующим функциям в контроллерах.

Файл .env

Файл переменных окружения, содержащий чувствительные данные (адрес базы данных, секретный ключ для JWT, номер порта и др.). Этот файл не включается в систему контроля версий и обеспечивает безопасную настройку приложения без раскрытия конфиденциальной информации.

Файл server.js

Главный файл проекта, с которого начинается запуск сервера. Здесь подключаются все библиотеки, middleware, маршруты и устанавливается соединение с MongoDB. После всех инициализаций приложение запускается на заданном порту. Этот файл объединяет все части проекта в единое работающее приложение [5].

Файл package.json

Содержит основную информацию о проекте: название, версию, зависимости и скрипты. Здесь указаны используемые библиотеки, такие как express, mongoose, dotenv, bcryptjs, jsonwebtoken и другие. Также определены команды для запуска сервера, например "start": "node server.js" или "dev": "nodemon server.js".

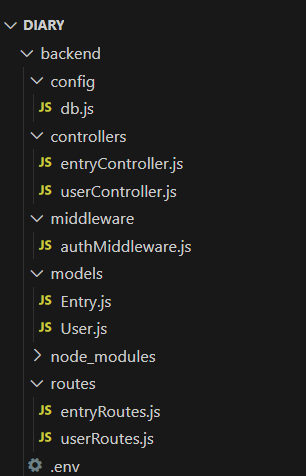
****

Рис.2.2.1 Структура backend-проекта

Backend-проект разделён на логические модули, каждый из которых выполняет отдельную функцию в архитектуре приложения. Такая структура способствует упрощению поддержки, масштабирования и повторного использования кода.

## **2.2.1. Модели базы данных**

Созданы две основные модели:

**Пользователь (User):**

id — уникальный идентификатор

username — имя пользователя (уникальное)

password — хешированный пароль

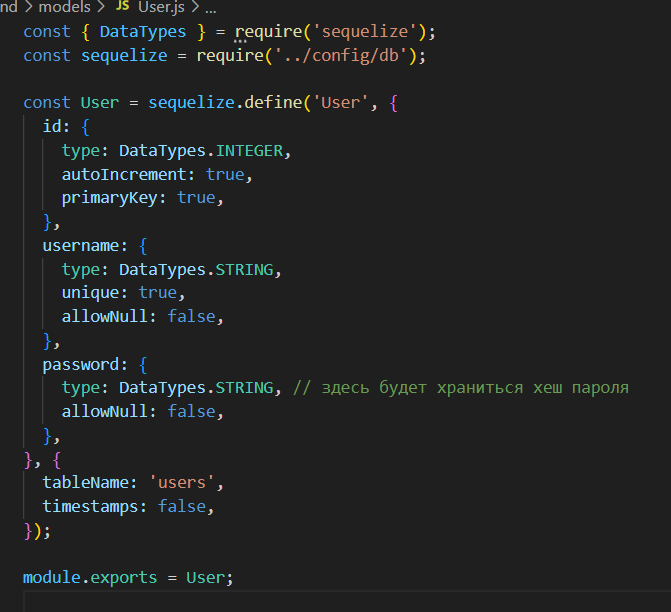


Рис.2.2.2 Модель User описывает пользователя с уникальным username и хешированным password

**Запись (Entry):**

* id — уникальный идентификатор записи
* title — заголовок
* content — содержимое
* tags — теги (в виде строки или массива)
* date — дата создания
* userId — внешний ключ, связывающий запись с пользователем



Рис.2.2.3. Модель Entry описывает запись дневника с заголовком, содержимым, тегами (строка с тегами через запятую), датой и связью с пользователем через userId.

## 

## **2.2.2. Реализация API**

Все маршруты реализованы в REST-стиле:

**Пользователи:**

|  |  |  |
| --- | --- | --- |
| **Метод** | **Маршрут** | **Описание** |
| POST | /api/auth/register | Регистрация пользователя |
| POST | /api/auth/login | Авторизация, выдача токена |
| PUT | /api/entries/:id | Обновление записи |
| DELETE | /api/entries/:id | Удаление записи |

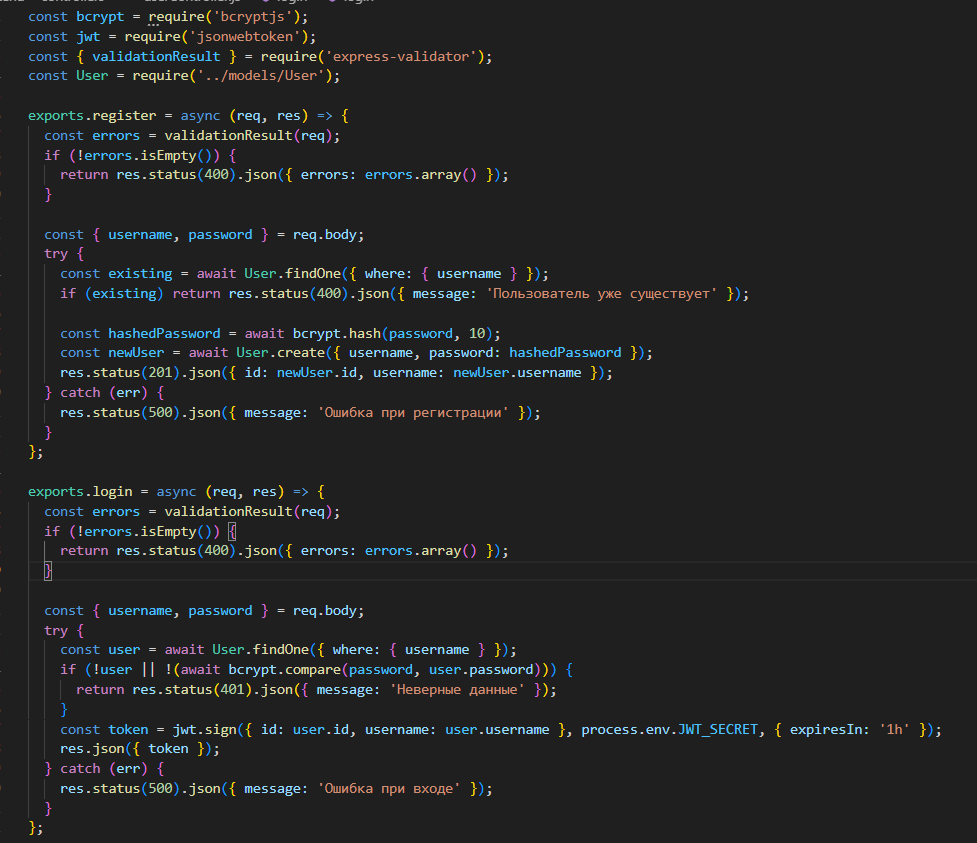


Рис.2.2.4 register() Создаёт нового пользователя с безопасным хешированием пароля. login() Проверяет логин и пароль и выдаёт токен при успешной авторизации.

Bcryptjs Защищает пароли от кражи (даже при взломе базы). Jsonwebtoken Обеспечивает сессии и авторизацию без сохранения сессии на сервере. express-validator Проверяет правильность введённых данных перед обработкой [1].

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рис.2.2.5  **Entry** — модель записи (из базы данных).**req.user.id** — ID

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рис.2.2.5  **Entry** — модель записи (из базы данных).**req.user.id** — ID Текущего пользователя, полученный из токена авторизации (через middleware). **validationResult** — проверка правильности данных (например, не пустой заголовок). req.user.id ID текущего авторизованного пользователя (из JWT, через middleware). validationResult(req) Проверка данных перед выполнением действия. Entry.findAll, findOne, create, destroy Методы Mongoosh для работы с записями. res.status().json() Отправка ответа клиенту (успех или ошибка)

## **2.2.3. Авторизация и безопасность**

Для защиты пользовательских данных и обеспечения безопасности взаимодействия с сервером реализована авторизация на основе JWT (JSON Web Token).

При входе в систему (авторизации) пользователь отправляет логин и пароль. Если данные верны, сервер генерирует уникальный токен и возвращает его клиенту. Этот токен имеет ограниченный срок действия (обычно 1 час) и служит подтверждением личности пользователя [9].

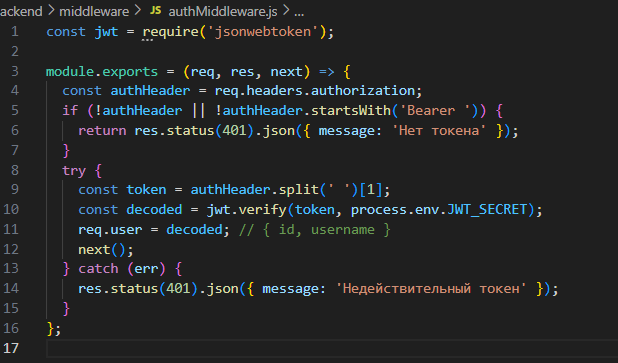


Рис.2.2.6 При входе пользователь получает токен (время жизни ~1 час).

Все запросы к маршрутам записей требуют наличие Authorization: Bearer <token>. Middleware authMiddleware.js проверяет токен и передаёт ID пользователя в req.user.

## **2.2.4. Тестирование API**

Тестирование веб-приложения для ведения личного дневника выполнялось на локальном сервере, где фронтенд работал по адресу http://localhost:3000/, а бэкенд (API) — по адресу http://localhost:5002/api/. В процессе тестирования проверялись ключевые функции приложения, включая регистрацию и вход пользователя с использованием JWT-аутентификации, а также операции с записями дневника: создание, редактирование и удаление.

Особое внимание уделялось корректному отображению записей конкретного пользователя и работе функционала тегов, включая фильтрацию по ним. Для проверки правильности работы API использовалась программа Postman, а пользовательский интерфейс тестировался в браузере.

Кроме того, проверялась защита маршрутов: все запросы к защищённым ресурсам требовали наличия действительного токена авторизации, а попытки доступа к чужим данным корректно блокировались. На главной странице проверялись отображение статистики — количество записей, список тегов и данные пользователя.

Результаты тестирования подтверждают, что приложение работает стабильно и соответствует требованиям по безопасности и функционалу. Все основные сценарии прошли успешно без критических ошибок [3].

Таблица 1. Результаты тестирования программы

|  |  |  |  |
| --- | --- | --- | --- |
| Выполняемое действие | Ожидаемый результат | Фактический результат | Статус |
| Регистрацияпользователя Mahabat(email  :mahabat@gmail.com, пароль: 123) | Успешная регистрация, выдача JWT, переход на главную страницу | Регистрация прошла успешно, пользователь авторизован | Успешно |
| Повторная регистрация того же email | Ошибка: пользователь уже существует | Регистрация заблокирована, отображено сообщение об ошибке | Успешно |
| Вход пользователя Mahabat | Запись сохраняется в базе, доступна на главной | Запись успешно создана и отображена | Успешно |
| Редактирование ранее созданной записи | Обновлённые данные сохраняются и отображаются | Изменения успешно применены, запись обновлена | Успешно |
| Удаление записи | Запись удаляется, в списке больше не отображается Удаление записи | Запись удалена без ошибок | Успешно |
| Доступ к маршрутам без токена | Ошибка 401: авторизация обязательна | Доступ запрещён, отображено сообщение об ошибке | Успешно |
| Попытка пользователя Vlad получить записи Mahabat | Ошибка доступа (403), защита сработала | Запрещённый доступ, данные защищены | Успешно |
| Проверка статистики на главной: количество | Отображается количество записей, | Все данные загружены корректно | Успешно |

Все проведённые тесты подтвердили корректную работу основных функций веб-приложения. Обработка ошибок происходит согласно ожиданиям, а система авторизации надёжно защищает данные пользователей. Благодаря использованию JWT обеспечивается безопасный и удобный механизм аутентификации.  
Приложение устойчиво к попыткам несанкционированного доступа, что гарантирует конфиденциальность личных данных и защиту от распространённых уязвимостей, включая XSS и CSRF.

Кроме того, протестированы сценарии работы с базой данных, что подтверждает целостность и корректность сохранения, обновления и удаления записей дневника. Интерфейс приложения отзывчив и корректно отображается на различных устройствах и браузерах, что обеспечивает широкий охват аудитории.

В дальнейшем планируется расширение функционала, включая более сложные сценарии фильтрации и аналитики, интеграцию с внешними сервисами для резервного копирования данных и улучшение интерфейса для повышения удобства пользователя. Также рассматривается внедрение возможностей офлайн-работы и синхронизации данных при восстановлении подключения.

Тестирование показало, что текущая реализация является стабильной и готовой к развертыванию в рабочей среде. Регулярное обновление и поддержка кода, а также внедрение новых модулей будут способствовать дальнейшему росту качества и функциональности приложения.

# **Заключение**

В ходе выполнения курсового проекта была разработана полноценная веб-система для ведения личного дневника. Приложение предоставляет пользователю удобный интерфейс для создания, просмотра личных записей, а также реализует регистрацию, авторизацию и безопасное хранение данных.

На серверной части использовались технологии Node.js, Express и база данных (MySQL или MongoDB — в зависимости от варианта реализации). Реализованы REST API, проверка данных, авторизация с использованием JWT, защита маршрутов и базовые механизмы безопасности.

Клиентская часть выполнена с использованием HTML, CSS и JavaScript (или React), обеспечивая интерактивный и удобный интерфейс для взаимодействия с системой. Интерфейс включает боковое меню навигации, карточки записей и модальные окна для ввода данных.

Создание такого приложения позволило закрепить знания по архитектуре веб-приложений, работе с базами данных, маршрутизации, взаимодействию frontend и backend, а также обеспечению безопасности и авторизации пользователей.

Проект может быть легко расширен в будущем: добавить поиск по записям, фильтрацию по тегам, прикрепление изображений, напоминания и многое другое. Таким образом, веб-дневник представляет собой не только учебную, но и практическую разработку, которую можно использовать и развивать в реальных условиях.

# **Список используемой литературы**

1. Документация Express.js — <https://expressjs.com/ru/>
2. Документация MongoDB Manual — <https://docs.mongodb.com/manual/> (дата обращения: 28.05.2025)
3. Документация Mongoose — <https://mongoosejs.com/docs/>
4. Документация по JWT — <https://jwt.io/>
5. Документация Node.js — <https://nodejs.org/ru/docs/>
6. Документация React — <https://reactjs.org/>
7. Документация Sequelize ORM — <https://sequelize.org/>
8. Курсы и статьи на сайте <https://habr.com/> по разработке веб-приложений
9. Node.js Documentation — <https://nodejs.org/en/docs/> (дата обращения: 28.05.2025)
10. OpenAI ChatGPT — консультации и генерация пояснительных текстов