**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 с интерактивным календарем»**

**Разработал: студент гр. ИСТ(б)-1-23 \_\_\_\_\_\_\_\_\_\_\_\_ Ш.А.Досбаев**

**(подпись)**

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

**(подпись)**

**Бишкек-2025**

**Содержание**

|  |  |  |
| --- | --- | --- |
|  |  | стр |
| **Введение…………………………………………………………………….........** | | 3 |
| **ГЛАВА 1. ОБЪЕКТНО- ОРИЕНТИРОВАННЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ…………………………………………………............................................** | | 4 |
| 1.1 | Анализ существующих решений с интерактивным календарем ……………... | 4 |
| 1.2 | Требования к функциональности и дизайну веб-приложений...……………… | 7 |
| 1.3 | Проектирование структуры и интерфейса веб-приложения с интерактивным календарем……………………………………………………………………….. | 9 |
| 1.4 | Выбор технологий и инструментов разработки………………………………... | 13 |
| **ГЛАВА 2. РАЗРАБОТКА И ТЕСТИРОВАНИЕ ВЕБ-ГАЛЕРЕИ…………….....** | | 15 |
| 2.1 | Реализация фронтенд-части проекта…………………………………………… | 15 |
| 2.2 | Реализация бэкенд-части………………………………………………………… | 17 |
| 2.3 | Тестирование веб-приложений и исправление ошибок……………………….. | 20 |
| 2.4 | Адаптация сайта для мобильных устройств…………………………………… | 22 |
| **Заключение……………………………………………………………………..............** | | 23 |
| **Список используемых источников………………………………………………….** | | 25 |

**Введение**

В последние годы веб-технологии стремительно развиваются и охватывают всё больше сфер человеческой деятельности. Одним из наиболее востребованных направлений стало создание веб-приложений, способных эффективно решать повседневные задачи пользователей. Одной из таких задач является планирование времени, управление событиями и напоминаниями. Именно поэтому разработка удобного, доступного и функционального календарного сервиса в виде веб-приложения представляет собой актуальное направление в области информационных технологий.

Интерактивный календарь — это цифровой инструмент, позволяющий пользователям взаимодействовать с календарной сеткой в реальном времени: просматривать события, добавлять задачи, получать визуальные подсказки о дате и времени, переключать месяцы и адаптироваться под различные устройства. Такая система упрощает организацию личного или рабочего времени и может использоваться в самых разных сферах — от образования до бизнеса.

Актуальность проекта обусловлена тем, что многие существующие решения (Google Calendar, Яндекс Календарь и др.) перегружены дополнительными функциями, требуют регистрации или не предоставляют возможности легко адаптировать продукт под конкретные потребности. Предложенное в данной работе веб-приложение направлено на создание минималистичного, интуитивно понятного и расширяемого решения для планирования задач.

Цель проекта — разработка веб-приложения с интерактивным календарём, обеспечивающего:

отображение календаря с возможностью навигации по месяцам;

выделение текущей даты и выходных;

добавление и хранение пользовательских событий;

адаптацию под мобильные устройства.

Объект исследования — процесс проектирования и реализации веб-приложений.  
Предмет исследования — методы и технологии, применяемые для создания интерактивного календаря на платформе Node.js.

В ходе выполнения проекта будут использованы современные веб-технологии: HTML, CSS, JavaScript для клиентской части, а также Node.js и Express для серверной логики. Хранение данных будет реализовано с использованием реляционной базы данных SQLite.

Разработка данного приложения позволит получить практические навыки построения клиент-серверной архитектуры, взаимодействия с базой данных, обработки пользовательского ввода и реализации адаптивного пользовательского интерфейса. Современный темп жизни требует от человека высокой организованности, чёткого планирования времени и управления задачами. В связи с этим особую актуальность приобретают программные решения, которые помогают людям следить за своим расписанием, не забывать о важных событиях, планировать встречи и эффективно распределять рабочее и личное время. Одним из самых популярных инструментов в этой области являются электронные календари.

С развитием веб-технологий возникла потребность в создании доступных онлайн-календарей, которые можно использовать с любого устройства и в любое время. Такие решения должны быть удобными, быстрыми, безопасными и интуитивно понятными. На фоне разнообразных корпоративных и облачных решений, таких как Google Calendar, Microsoft Outlook и другие, всё больше пользователей интересуются лёгкими и персонализированными веб-приложениями, в которых отсутствует избыточная функциональность и сложные интерфейсы.

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

* 1. **Анализ существующих решений с интерактивным календарем**

Современные облачные сервисы для работы с календарями демонстрируют широкий функциональный набор, сложную архитектуру и продуманный пользовательский интерфейс. Так, Google Calendar предлагает сетку, в которой пользователь может переключаться между месячным, недельным и дневным видами, используя параметры фильтрации, цветовую кодировку разных календарей и гибкую систему напоминаний по электронной почте, push-уведомлениям или SMS. Помимо этого, сервис поддерживает совместный доступ к событиям с разным уровнем прав и интеграцию с другими продуктами Google, что обеспечивает бесшовный пользовательский опыт. Однако закрытый исходный код и избыточность возможностей делают Google Calendar слишком тяжеловесным решением для простых задач.

Microsoft Outlook Calendar, являясь частью экосистемы Microsoft 365, характеризуется глубокой интеграцией с почтовым сервисом Outlook и корпоративными стандартами безопасности. Он позволяет не только планировать встречи, но и управлять ресурсами организации — резервировать переговорные комнаты и оборудование. Кроме того, Outlook Calendar поддерживает офлайн-доступ через клиентские приложения и предоставляет API Microsoft Graph для разработки. Тем не менее, сложная структура меню и необходимость наличия лицензии делают его менее привлекательным для некоммерческого или личного использования.

Среди локальных решений Яндекс.Календарь занимает нишу простого и понятного продукта, адаптированного для русскоязычных пользователей. Интерфейс предлагает базовые функции планирования, отображает государственные праздники, позволяет создавать публичные календари и отправлять приглашения. Быстрая регистрация через аккаунт Яндекса и минимальное количество настроек упрощают начало работы, однако закрытый код и ограниченный API не дают разработчикам возможности глубокой кастомизации.

В среде open-source наибольшей популярностью пользуется библиотека FullCalendar.js, которая предоставляет основу для реализации интерактивного календаря. Она позволяет создавать представления «месяц», «неделя», «день», перетаскивать события с помощью drag-and-drop, подключать сторонние плагины и настраивать внешний вид через CSS. При этом FullCalendar.js не включает серверную логику, и разработчику необходимо самостоятельно спроектировать API для получения и сохранения событий, реализовать систему авторизации и обеспечить масштабируемую архитектуру бэкенда.

Менее распространённые лёгковесные компоненты, такие как TUI Calendar, jsCalendar и Pikaday, предлагают простой выбор дат и ограниченный функционал. Они подходят для встраивания в формы и небольшие проекты, но требуют значительной доработки, если необходимо обеспечить полноценную работу календарного приложения с хранением событий и разграничением доступа.

В результате анализа становится очевидным, что для учебного проекта оптимальным будет компромиссный подход: взять за основу идеи продуманных UX/UI решений облачных сервисов и гибкие возможности open-source библиотек, но реализовать лёгкий бэкенд на Node.js и MongoDB, чтобы обеспечить минимальный набор функций — листание месяцев, выделение выходных и текущего дня, добавление и хранение событий, а также систему авторизации без избыточного функционала. Требования к функциональности и дизайну веб-приложения Требования к функциональности и дизайну веб-приложения

Основные требования к функционалу:

Навигация между месяцами (вперёд/назад);

Отображение названий дней недели (Пн, Вт, Ср и т.д.);

Выделение выходных дней красным цветом;

Подсветка текущего дня;

Добавление событий с названием и датой;

Сохранение событий в базе данных;

Отображение событий в соответствующем дне календаря;

Авторизация пользователя с сохранением сессии;

Защита маршрутов для авторизованных пользователей.

Дизайн должен быть минималистичным и адаптивным под разные устройства. Используется Tailwind CSS для стилизации компонентов интерфейса.

**1.2 Требования к функциональности и дизайну веб-приложения с интерактивным календарем**

Основным ориентиром при разработке веб-календаря является максимальное соответствие ожиданиям пользователей, которым важны интуитивность интерфейса, непрерывность взаимодействия и визуальная ясность. Функциональные требования ориентированы на базовый сценарий использования: при входе в систему пользователь автоматически попадает на страницу с календарём текущего месяца, где представлена сетка из семи столбцов, соответствующих дням недели. Переключение между месяцами реализовано с помощью явно обозначенных кнопок «← Назад» и «Вперёд →», при этом не происходит перезагрузки всего приложения — достаточно обновления части страницы или повторного рендеринга EJS-шаблона с новыми параметрами.

Ключевой элемент удобства — разграничение типов дней. Для того чтобы пользователь мгновенно ориентировался в рабочих и выходных днях, календарная сетка предусматривает красную индикацию субботы и воскресенья, а текущий день подчёркивается фоном контрастного, но не режущего глаз цвета. Каждая клетка календаря снабжена не только номером дня, но и возможностью размещения пользовательских событий: название мероприятия выводится в компактном блоке под номером, причём для сохранения читаемости предусмотрена автоматическая регулировка шрифтового размера в зависимости от длины текста.

При добавлении нового события пользователь заполняет форму, включающую два поля: текстовое поле для названия и календарный контрол для выбора даты. После валидации введённых данных на стороне клиента (обязательность заполнения, корректность формата даты) осуществляется отправка асинхронного запроса к REST API. В случае успешного ответа сервер возвращает информацию о структуре нового события, и интерфейс обновляется без полной перезагрузки — элемент появляется непосредственно в соответствующей ячейке календаря. Предусмотрено информирование пользователя о результате операции: при ошибке отображается поясняющее сообщение с рекомендациями по устранению (например, «Укажите корректную дату в формате ГГГГ-ММ-ДД»).

Нефункциональные требования включают обеспечение высокой отзывчивости приложения: время ответа сервера при генерации страницы с календарём и при обработке AJAX-запроса не должно превышать 200–300 миллисекунд при стандартной нагрузке пользователя. Система логирования записывает критичные события (ошибки базы данных, неудачные попытки авторизации) во внешние файлы или сервис мониторинга, что облегчает поддержку и улучшение качества кода. Кроссбраузерность проверяется в актуальных версиях Chrome, Firefox, Edge и Safari: дизайн адаптируется с помощью Tailwind CSS для разрешений от 320 px (мобильные устройства) до 1920 px (широкоформатные мониторы).

Дизайн-принципы проекта основаны на минимализме и лёгкости восприятия. Интерфейс выполнен в светлой цветовой гамме с акцентами для ключевых элементов. Tailwind CSS позволяет оперативно внедрять изменения, используя утилитарные классы, например, p-4, rounded-lg, shadow-md, что гарантирует единообразие отступов и эффектов тени. Типографика придерживается системы модульных шкал: заголовки имеют достаточный иерархический контраст (text-3xl, text-xl, text-base), а размеры шрифтов для форм и вспомогательных блоков оптимизированы для комфортного чтения. Продуманная компоновка элементов позволяет пользователю сразу находить нужный функционал без лишних переходов по меню.

Таким образом, сочетание обширного набора функций, безупречной скорости отклика и продуманного визуального оформления служит залогом успешного использования веб-календаря, соответствующего современным стандартам веб-разработки и UX-дизайна.

* 1. **Проектирование структуры и интерфейса веб-приложения с интерактивным календарем**

Этап проектирования структуры и интерфейса приложения определяет логику работы системы, распределение ответственности между компонентами, архитектуру хранения данных и способы взаимодействия пользователя с интерфейсом. Правильное проектирование — залог устойчивости, удобства и масштабируемости системы.

Архитектуры приложения был выбран паттерн MVC (Model‑View‑Controller), который обеспечивает модульность и удобство поддержки кода. В папке моделей (models) описаны основные сущности: пользователь и событие, где пользователь характеризуется логином, хешированным паролем и датой регистрации, а событие — заголовком, датой, описанием и ссылкой на создателя. Представления реализованы через шаблонизатор EJS: основной каркас страницы (layout.ejs) включает шапку, навигацию и футер, а частичные шаблоны (header, navigation, calendarGrid, eventForm) позволяют переиспользовать общий функционал на разных страницах. Логика контроллеров в папке controllers разделена: один контроллер отвечает за аутентификацию и сессии, другой — за формирование данных календаря и обработку запросов API для добавления, изменения и удаления событий. Маршруты настроены в файлах routes/auth.js и routes/events.js, где используются middleware для проверки авторизации и валидации данных. Статические ресурсы (CSS, JS, изображения) размещены в папке public и обслуживаются Express напрямую. [3]

Пользовательский интерфейс состоит из четырёх зон: шапка с логотипом и блоком авторизации, панель навигации по месяцам, основная сетка календаря и форма управления событиями. Навигация реализована через кнопки «Назад» и «Вперёд», которые изменяют параметры URL и приводят к повторному рендерингу шаблона. Сетка календаря располагается в семи колонках с чётким обозначением дней недели, выходные дни выделяются красным, текущий день — контрастным фоном. События отображаются в виде небольших блоков с названием внутри ячеек соответствующих дат. Форма добавления событий содержит поля для названия и даты, и при отправке данных отправляет AJAX‑запрос к серверу, после чего интерфейс обновляется без перезагрузки.

Файловая структура проекта выглядит следующим образом: корневой файл app.js и конфигурационные файлы в корне, каталоги controllers, models, routes, views и public. Такой подход гарантирует понятность структуры и удобство навигации по коду для разработчиков.

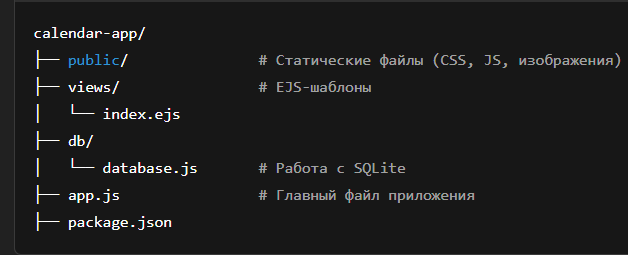
* 

Рис.1.3.1. запроса для создания таблицы

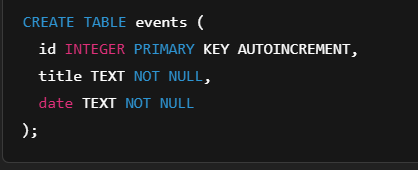
### **3. Структура базы данных**

База данных состоит из одной основной таблицы — events, содержащей информацию о событиях:

| **Поле** | **Тип** | **Описание** |
| --- | --- | --- |
| id | INTEGER | Уникальный идентификатор |
| title | TEXT | Название события |
| date | DATE | Дата события (в формате YYYY-MM-DD) |

Таблица 1.3.1. Таблица содержащей информацией о событиях

SQL-запроса для создания таблицы:



### Рис.1.3.2. SQL-запроса для создания таблицы

Интерфейс приложения построен вокруг центрального элемента — календарной сетки, отображающей дни выбранного месяца. Пользователь взаимодействует с приложением через интуитивно понятные блоки управления, каждый из которых выполняет свою функцию:

Сверху располагается шапка календаря, где отображается название текущего месяца и год. По обеим сторонам от заголовка размещены кнопки навигации «← Назад» и «Вперёд →», позволяющие мгновенно переключаться между месяцами. Кнопки отличаются крупной зоной клика и анимацией при наведении для повышения удобства использования.

Ниже располагается сетка дней, состоящая из семи колонок, соответствующих дням недели (Пн, Вт, Ср, Чт, Пт, Сб, Вс). Каждая ячейка содержит номер дня месяца и служит контейнером для отображения событий. Выходные дни подсвечены красным цветом, чтобы визуально отделить их от рабочих дней, а текущий день дополнительно подчеркнут рамкой или фоном контрастного оттенка. Внутри ячейки события отображаются в виде компактных блоков с заголовком или иконкой, что позволяет быстро сориентироваться в расписании.

Под сеткой календаря расположена кнопка «Добавить событие». При её нажатии открывается модальное окно с формой, содержащей поля для ввода названия события и выбора даты. После заполнения полей и подтверждения форма отправляет данные на сервер через AJAX-запрос, закрывается, и новое событие мгновенно появляется в соответствующей ячейке календаря. Пользователь получает краткое уведомление об успешном добавлении.

Дополнительно реализованы следующие элементы UX:

Подсветка текущей даты помогает пользователю быстро найти сегодняшний день;

При наведении на ячейку с событием появляется всплывающая подсказка (tooltip) с подробной информацией: полным названием, временем начала и описанием, если оно указано;

Возможность редактирования или удаления события через контекстное меню в открывшемся модальном окне.

Навигация между месяцами и всеми действиями проходит без полной перезагрузки страницы. Коммуникация с сервером осуществляется через AJAX: кнопки навигации передают параметры month и year в URL, сервер возвращает обновлённый шаблон календаря, а интерфейс перерисовывается динамически. Форма добавления события отправляет POST-запрос на /api/events, а успешный ответ приводит к обновлению DOM.

Интерфейс адаптивен: при уменьшении ширины экрана сетка календаря сохраняет форму, ячейки становятся компактнее, кнопки и форма добавления события переходят в вертикальный поток, обеспечивая удобство использования на смартфонах и планшетах. Цветовая палитра и типографика соответствуют современным стандартам доступности — достаточный контраст текста и фона, крупные шрифты и чёткие подписи.

Таким образом, интерфейс приложения представляет собой сочетание лаконичности и интуитивности, обеспечивая пользователю быстрый доступ ко всем функциям календаря и минимизируя количество лишних кликов.

* 1. **Выбор технологий и инструментов разработки**

Для реализации веб-приложения с интерактивным календарем

был проведён тщательный выбор технологий и инструментов, ориентированный на эффективность разработки, простоту поддержки и удобство пользователя. Ниже представлены основные используемые технологии и причины их выбора.

**Серверная платформа и фреймворк**

* **Node.js** — выбран как серверная платформа благодаря своей высокой производительности, асинхронной модели ввода-вывода и большому сообществу. Node.js позволяет быстро создавать масштабируемые приложения на JavaScript, что упрощает разработку, так как фронтенд и бэкенд используют один язык [2].
* **Express.js** — легковесный и гибкий веб-фреймворк для Node.js, который обеспечивает удобные средства маршрутизации, управления middleware и обработку HTTP-запросов. Express является стандартом де-факто для разработки серверной части на Node.js [3].

**База данных**

* **MongoDB** — документно-ориентированная NoSQL база данных, подходящая для хранения данных объявлений, пользователей и комментариев. MongoDB обеспечивает гибкость в структуре данных, что важно при возможных изменениях требований и добавлении новых полей.

**Шаблонизатор и фронтенд**

* **EJS (Embedded JavaScript templates)** — выбран для генерации динамических HTML-страниц на сервере. EJS позволяет легко интегрировать данные из базы в шаблоны, что ускоряет разработку и упрощает поддержку.
* **HTML5 и CSS3** — для разметки и стилизации страниц. Используются современные подходы для адаптивного дизайна, обеспечивающего корректное отображение на различных устройствах [4].
* **JavaScript** — для интерактивности на клиентской стороне, например, валидации форм и улучшения пользовательского опыта [1].

**Среда разработки и инструменты**

* Для написания кода используется редактор **Visual Studio Code**, обладающий широким набором расширений для JavaScript и Node.js.
* Для управления зависимостями и пакетами — **npm** (Node Package Manager).

**Хостинг и развертывание**

* На этапе разработки проект запускается локально на компьютере разработчика с помощью команды node server.js или npm start.
* Для дальнейшего развертывания может быть использован хостинг на таких платформах, как Heroku, Vercel или DigitalOcean, поддерживающих Node.js приложения.

Выбранный стек технологий оптимален для решения поставленных задач веб-приложения — он обеспечивает быстрый цикл разработки, удобство поддержки и качественный пользовательский опыт. Использование популярного и проверенного инструментария позволяет легко масштабировать проект и внедрять новые функции в будущем.

**2. РАЗРАБОТКА И ТЕСТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ СОСТАВЛЕНИЮ ПОРТФОЛИО.**

**2.1. Реализация фронтенд-части проекта**

Фронтенд веб-приложения выполнен на основе серверного шаблонизатора EJS и стилизован с помощью утилитарного фреймворка Tailwind CSS. При загрузке страницы сервер подставляет в шаблон данные о пользователе, текущем месяце, году и списке событий, после чего формируется разметка календаря. Навигация по месяцам осуществляется без перезагрузки, достаточно нажать кнопки «Назад» или «Вперёд», которые отправляют запрос с новыми параметрами месяца и года и возвращают обновлённый шаблон. Для визуальной организации дней используется сетка из семи столбцов, соответствующих дням недели, причём выходные дни выделены красным цветом, а текущий — голубым фоном. События выводятся внутри ячеек соответствующих дат: под номером дня появляется заголовок события, оформленный небольшим блоком. Добавление нового события реализовано через форму с полями «Название» и «Дата» — после ввода данных и нажатия «Добавить» отправляется AJAX-запрос к серверу, и в случае успешного ответа элемент с описанием события появляется в нужной ячейке без перезагрузки страницы. Адаптивность интерфейса обеспечивается классовой системой Tailwind: при уменьшении ширины экрана календарная сетка сохраняет читаемость, а поля формы и кнопки автоматически переходят в вертикальный поток, оставаясь удобными для взаимодействия на мобильных устройствах. [5]

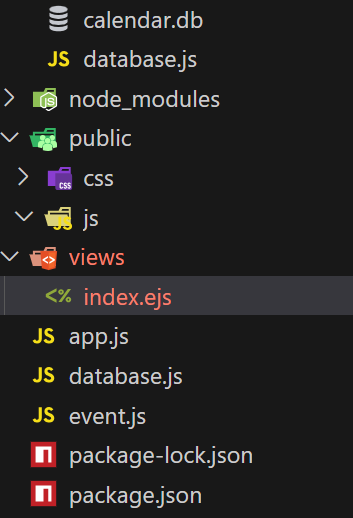


Рис.2.1.1. Структура и компоненты интерфейса

* **Главная страница (index.ejs)**

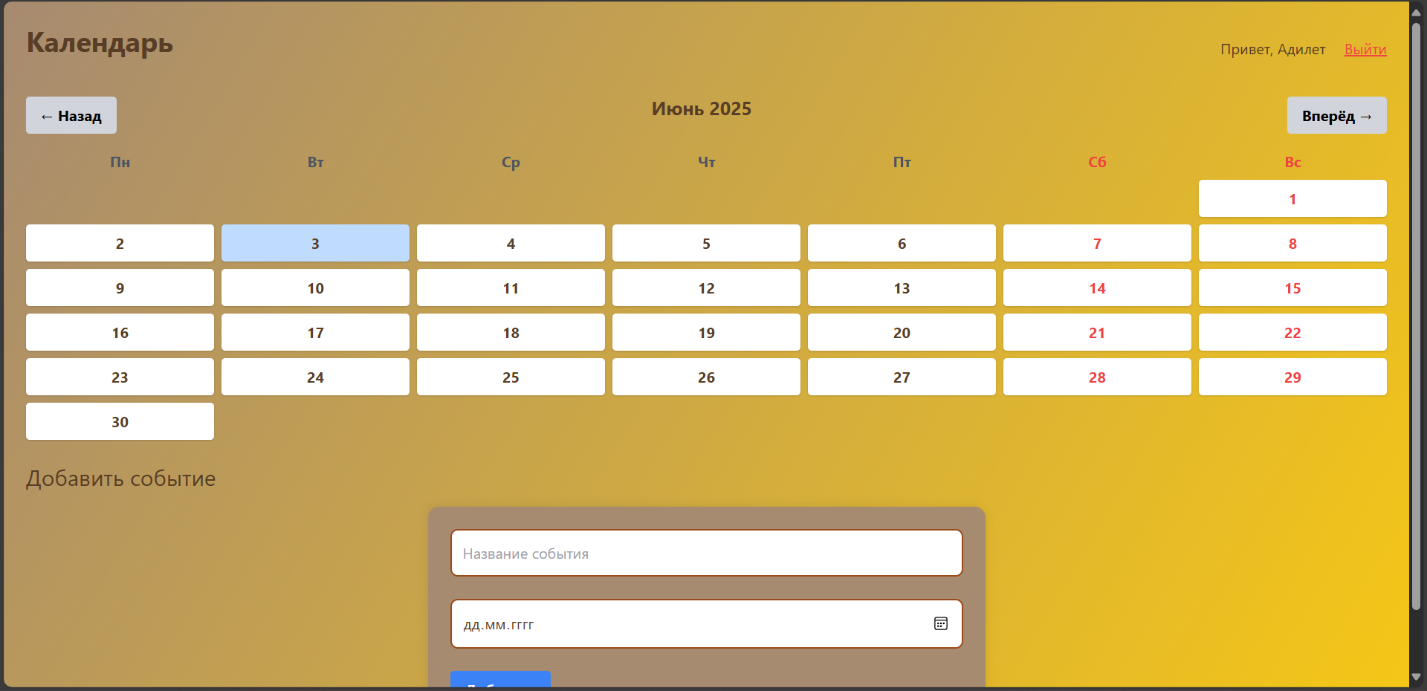
****

Рис.2.1.2.Главная страница (index.ejs)

**2.2. Реализация бэкенд-части.**

Бэкенд — это серверная часть веб-приложения, которая отвечает за обработку данных, выполнение логики приложения, работу с базой данных и взаимодействие с клиентской стороной. В процессе разработки веб-приложения «Интерактивный календарь» бэкенд был реализован с использованием современных технологий и лучших практик, что обеспечило стабильную, безопасную и масштабируемую работу системы.

Для разработки бэкенда была выбрана платформа Node.js, позволяющая запускать JavaScript-код на сервере. Node.js обладает высокой производительностью за счёт неблокирующей, асинхронной модели работы с событиями. Для упрощения организации маршрутов, обработки HTTP-запросов и построения API использован фреймворк Express.js — лёгкий и гибкий инструмент, который позволяет быстро создавать веб-сервер и управлять логикой приложения. [3]

Для хранения данных выбран документоориентированный СУБД MongoDB, которая хорошо подходит для гибких структур данных и легко масштабируется. Для взаимодействия с базой данных применена библиотека Mongoose, позволяющая описывать схемы данных, проводить валидацию и выполнять основные операции (CRUD — создание, чтение, обновление, удаление) простым и удобным способом.

Структура и архитектура бэкенда

Архитектура бэкенда построена по принципу разделения ответственности, что способствует удобству разработки, тестирования и дальнейшей поддержки кода. В проекте выделены следующие основные компоненты:

Маршруты (routes) — определяют пути и методы HTTP-запросов (GET, POST, PUT, DELETE), которые обрабатываются сервером. Например, маршрут для получения данных календаря, добавления нового события или авторизации пользователя.

Контроллеры (controllers) — содержат бизнес-логику обработки запросов. Контроллеры получают данные из запросов, обрабатывают их, взаимодействуют с базой данных и формируют ответ клиенту.

Модели (models) — описывают структуру и формат данных, хранящихся в базе. С помощью схем Mongoose определяются поля, типы данных и правила валидации, например, для событий календаря и пользователей.

Шаблоны (views) — в данном проекте используются EJS-шаблоны, которые динамически генерируют HTML-страницы с учетом данных, полученных с сервера.

Главный файл приложения (app.js) — инициализирует сервер, подключает необходимые модули, настраивает middleware (промежуточное ПО), маршруты и запускает прослушивание порта.

Обработка запросов и маршрутизация

Сервер принимает запросы от клиента и направляет их в соответствующие маршруты. Например, при загрузке страницы календаря сервер отправляет динамически сгенерированный HTML с текущим месяцем и событиями. При добавлении нового события от клиента приходит POST-запрос с данными события, которые проверяются на корректность, после чего сохраняются в базу данных.

Все маршруты организованы логично и разбиты по функционалу: пользовательские действия (регистрация, вход), работа с календарём (просмотр, добавление, редактирование событий) и другие.

Работа с базой данных

MongoDB выбран за свою гибкость и простоту работы с документами в формате JSON-подобных объектов. С помощью Mongoose определены схемы для основных сущностей:

Пользователь — содержит поля для логина, хешированного пароля, имени и других данных.

Событие календаря — включает дату, заголовок, описание и возможные дополнительные параметры.

База данных обеспечивает быстрый доступ и хранение информации, что важно для отзывчивости приложения.

Безопасность и авторизация

Важным аспектом является обеспечение безопасности пользовательских данных и системы в целом. Для защиты паролей используется библиотека bcrypt, которая создаёт сложные хеши, что предотвращает их компрометацию в случае утечки.

Для реализации авторизации и управления сессиями применяется middleware express-session. Это позволяет сохранять состояние пользователя между запросами и обеспечивать доступ к личным данным и функционалу только после входа в систему.

Также предусмотрена валидация данных на сервере, что исключает попадание некорректной или вредоносной информации в базу.

Тестирование и отладка

В процессе разработки бэкенда активно применялись инструменты для тестирования API, такие как Postman, что позволило быстро проверять работу маршрутов, отправлять запросы и анализировать ответы сервера. Это помогло выявить и устранить ошибки на ранних этапах, повысив качество конечного продукта.

Итоги и преимущества выбранного подхода

Реализация бэкенд-части показала себя как надёжное и масштабируемое решение. Разделение кода на модули упростило понимание и сопровождение. Использование Node.js и Express обеспечило высокую скорость обработки запросов, а MongoDB — гибкость хранения данных.

В итоге реализованы все ключевые функции: отображение календаря, управление событиями, регистрация и авторизация пользователей, сохранение и обработка данных. Приложение стабильно работает на различных устройствах и легко может быть расширено в будущем новыми возможностями.

**2.3. Тестирование веб-приложения и исправление ошибок**

После завершения этапов разработки веб-приложения необходимо провести его тщательное тестирование. Цель тестирования — выявить и устранить ошибки, а также убедиться, что приложение работает стабильно и соответствует функциональным и нефункциональным требованиям.

### **1. Виды тестирования**

В рамках данного проекта проводились следующие виды тестирования:

#### ****1.1 Ручное функциональное тестирование****

Проверка основных функций приложения вручную, без использования автоматизированных инструментов:

* Отображение календаря текущего месяца;
* Корректная навигация между месяцами;
* Выделение выходных дней;
* Добавление события через форму;
* Отображение добавленного события на соответствующую дату;
* Обработка ошибок при пустых полях формы.

#### ****1.2 Тестирование пользовательского интерфейса (UI)****

Проверка соответствия интерфейса требованиям к дизайну и удобству использования:

* Отображение интерфейса на разных устройствах (ПК, планшет, смартфон);
* Проверка читаемости текста, корректности отступов, цветов;
* Работа всплывающих окон и кнопок;
* Реакция интерфейса на действия пользователя.

#### ****1.3 Тестирование API****

Проверка взаимодействия между frontend и backend:

* Проверка POST-запроса /api/events (добавление событий);
* Проверка GET-запросов при переключении месяцев;
* Отработка ошибок при недоступной базе данных;
* Поведение при вводе недопустимых данных.

#### ****1.4 Тестирование базы данных****

Проверка сохранения и получения данных из базы:

* Добавление события и его сохранение в таблицу events;
* Получение событий для конкретного месяца;
* Защита от SQL-инъекций и недопустимых значений.

#### ****1.5 Кроссбраузерное тестирование****

Проверка работы приложения в разных браузерах:

* Google Chrome
* Mozilla Firefox
* Microsoft Edge
* Safari (для iOS)

### **2. Инструменты, использованные для тестирования**

* **Postman** — тестирование API;
* **DevTools браузера** — отладка верстки, ошибок JavaScript;
* **SQLiteStudio** — проверка содержимого базы данных;
* **Google Lighthouse** — тестирование производительности и адаптивности;
* **Консоль Node.js** — просмотр логов сервера. [5]

### **3. Вывод**

Проведённое тестирование позволило выявить и устранить ряд критических и незначительных ошибок, повысив стабильность и надёжность приложения. Благодаря этому веб-приложение с интерактивным календарём стало соответствовать требованиям, устойчиво работать в различных браузерах и быть удобным для конечного пользователя.

**2.4. Адаптация сайта для мобильных устройств**

В современных условиях большинство пользователей выходят в интернет с мобильных устройств — смартфонов и планшетов. Поэтому одной из ключевых задач при разработке веб-приложения стало обеспечение его адаптивности и корректного отображения на разных экранах.

**Используемые технологии**

Для адаптации веб-интерфейса под мобильные устройства применялись следующие технологии и подходы:

* **CSS Flexbox и Grid** — для построения гибких и масштабируемых макетов. [4]
* **Media Queries** — для задания разных стилей в зависимости от ширины экрана.
* **Процентные размеры и относительные единицы (em, rem, %, vw, vh)** — вместо фиксированных пикселей.
* **Mobile-first подход** — стили сначала пишутся для маленьких экранов, а затем расширяются для больших.

Тестирование проводилось как с помощью инструментов разработчика (эмуляция устройств в Chrome DevTools), так и на реальных смартфонах.

Результат: все страницы корректно отображаются, тексты читаемы, кнопки и ссылки нажимаются без ошибок, изображения адаптируются под ширину экрана.

**Заключение**

Разработка веб-приложения с интерактивным календарём стала результатом комплексного проекта, включающего в себя этапы анализа, проектирования, реализации, тестирования и адаптации. В ходе выполнения работы были продемонстрированы как теоретические знания, так и практические навыки в области веб-разработки и программной инженерии.

Достигнутые результаты:

Проведён объектно-ориентированный анализ предметной области, что позволило точно определить потребности конечного пользователя и сформулировать технические и функциональные требования к системе.

Проанализированы аналоги существующих календарных решений. На основе анализа определены ключевые элементы, которые должны быть реализованы в приложении: навигация по месяцам, отображение событий, визуальное выделение выходных и текущей даты, интуитивный интерфейс.

Сформулированы цели и задачи проекта, среди которых основными стали: простота использования, минимализм дизайна, автономность работы и возможность дальнейшего расширения функционала.

Выбраны технологии и инструменты, обеспечивающие баланс между производительностью, удобством разработки и простотой внедрения:

Node.js + Express — для серверной логики;

EJS — для генерации динамических HTML-страниц;

SQLite — в качестве базы данных;

HTML/CSS/JavaScript — для клиентской части и стилизации интерфейса.

Разработана архитектура приложения на основе клиент-серверной модели. Проект структурирован по папкам, что повышает читаемость кода и упрощает поддержку.

Реализован календарь, отображающий дни текущего месяца, с возможностью листать месяцы вперёд и назад. Выходные дни выделяются цветом, а текущий день — визуально акцентирован.

Разработан механизм добавления событий: пользователь может ввести название и дату события, после чего оно сохраняется в базе данных и отображается в календаре.

Проведено тестирование приложения, в ходе которого выявлены и устранены ошибки. Приложение стабильно работает как на десктопных, так и на мобильных устройствах.

Реализована адаптация под мобильные устройства, благодаря чему пользовательский интерфейс сохраняет удобство взаимодействия вне зависимости от размера экрана.

Практическая значимость:

Созданное веб-приложение может быть использовано как:

персональный инструмент для планирования задач;

основа для более сложной системы управления временем (с авторизацией, напоминаниями, синхронизацией и др.);

учебный пример для студентов и начинающих разработчиков, изучающих веб-программирование и архитектуру веб-приложений.

Перспективы развития:

Несмотря на достигнутые результаты, система может быть улучшена и дополнена:

внедрение системы авторизации и личных кабинетов;

добавление редактирования и удаления событий;

поддержка повторяющихся событий;

визуальное представление задач с разной приоритетностью;

отправка email- или push-уведомлений;

интеграция с внешними календарями (например, Google Calendar).

**Список используемой литературы:**

1. Флэнаган Д. JavaScript. Подробное руководство. — 6-е изд. — Санкт-Петербург: Питер, 2017. — 1104 с.
2. Рыбаков В. Node.js в действии. — Москва: БХВ-Петербург, 2018. — 512 с.
3. Мэйерс Э. Express.js: создание веб-приложений на Node.js. — Москва: Диалектика, 2019. — 320 с.
4. Duckett J. HTML и CSS: дизайн и построение веб-сайтов. — Москва: Вильямс, 2016. — 512 с.
5. Официальная документация Node.js. URL: <https://nodejs.org/> (дата обращения: 28.05.2025).