Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра информационных систем и технологий

Дисциплина «Основы алгоритмизации и программирования»

|  |
| --- |
| «К защите допустить» |
| Руководитель курсового проекта канд.техн.наук, доцент  А.В. Хмелёва |
| . .2025 |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему:

ПРОГРАММНОЕ СРЕСДВТО «ЭЛЕКТРОННЫЙ ЕЖЕДНЕВНИК»

БГУИР КП 6-05-0612-01 005 ПЗ

|  |
| --- |
| Выполнил студент группы 410971 ГУРИНОВИЧ Валерия Вячеславовна  (подпись студента) |
| Курсовой проект представлен на проверку 2025  (подпись студента) |

Минск 2025

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

(подпись)

Лапицкая Н.В.

« 4 » февраля 2025г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Гуринович Валерии Вячеславовне Тема работы *Программное средство «Электронный ежедневник»*

1. Срок сдачи законченной работы *01.06.2025г.*
2. Исходные данные к работе *Среда программирования: Visual Studio Code*

Система авторизации пользователей в программе. Предусмотрение контроля корректности ввода данных, а именно: имя, почта, пароль. Предоставление информации о программе. Возможность просмотра календаря с навигацией по месяцам. Возможность добавления записей в дни календаря. Возможность редактирования, удаления и форматирования текста записей ежедневника. Загрузка и отображение записей. Подтверждение важных действий пользователя с помощью диалоговых окон. Автоматическое сохранение в локальном хранилище.

1. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение

1. *Анализ литературных источников*
2. *Постановка задачи*
3. *Разработка программного средства*
4. *Тестирование и проверка работоспособности программного средства*
5. *Руководство по установке и использованию программного средства*

Заключение Список использованных источников

\_Приложения

1. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

Информационная модель. Блок-схемы алгоритмов, реализующих бизнес- логику. Листинг алгоритмов реализующих бизнес-логику. Листинг основных элементов программы.

1. Консультант по курсовой работе  *Хмелева А.В.*
2. Дата выдачи задания *04.02.2025г.*
3. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объема работы): *Раздел 1. Введение к 28.02.2025г. – 10 % готовности работы; Раздел 2 к 15.03.2025г. – 30% готовности работы*

Раздел 3 к 15.04.2025г. – 60% готовности работы Раздел 4 к 10.05.2025г. – 80% готовности работы

Раздел 5.Заключение. Приложения к 20.05.2025г. – 90% готовности работы; оформление пояснительной записки и графического материала к 01.06.2025г. – 100% готовности работы. Защита курсового проекта с 05.06.2025г.

РУКОВОДИТЕЛЬ *Хмелева А.В.*

(подпись)

Задание принял к исполнению *Гуринович В.В. 04.02.2025г.*

(дата и подпись студента)

ВВЕДЕНИЕ

Еще 20–30 лет назад основными инструментами планирования были бумажные ежедневники и блокноты. Люди записывали задачи, перечеркивали выполненное, делали пометки и лихорадочно листали страницы в поисках нужной информации. Такой способ организации дел требовал времени и внимания, а вносить изменения в расписание было не всегда удобно.

Однако в 21 веке технологии значительно упростили этот процесс. Электронные ежедневники пришли на смену традиционным бумажным носителям, предлагая удобные инструменты для планирования и управления временем. Они позволяют быстро добавлять, редактировать и сортировать задачи, устанавливать напоминания и синхронизировать данные между устройствами. [6]

Использование электронных ежедневников способствует снижению потребления бумаги, что положительно сказывается на окружающей среде. Это особенно важно в условиях глобальных экологических проблем. Каждый год миллионы деревьев вырубаются для производства бумаги, и переход на электронные средства планирования может помочь сократить этот объем. Кроме того, многие электронные приложения предлагают функции для отслеживания вашего углеродного следа и предлагают советы по его снижению. [5]

Одной из целей данного курсового проекта является разработка электронного ежедневника, предназначенного для удобного планирования задач, управления расписанием и повышения личной продуктивности. Используя современные технологии, пользователи смогут легко создавать, редактировать и организовывать свои дела, а также следить за выполнением запланированных мероприятий. Этот проект позволит не только структурировать повседневные задачи, но и повысить эффективность управления временем, минимизируя вероятность забытых дел. Создание удобного и функционального электронного помощника – это шаг к оптимизации повседневной рутины и упрощению процесса планирования для людей с любым уровнем занятости.

Разработка электронного ежедневника базируется на ряде фундаментальных принципов, обеспечивающих его эффективность, удобство и надежность. В основе проектирования лежит удобство использования, что означает интуитивно понятный интерфейс, не требующий длительного обучения. Гибкость и масштабируемость позволяют адаптировать приложение под различные сценарии использования и расширять его функционал в будущем. Кроме того, система должна быть интерактивной и наглядной, предлагая удобные инструменты визуализации задач, календарей и общего функционала.

С научной точки зрения проектирование базируется на системном подходе, включающем анализ существующих решений, их преимуществ и недостатков, а также формирование требований к разрабатываемому продукту. Исследование пользовательских потребностей позволяет понять, какие функции наиболее востребованы. Также важной частью процесса является тестирование и валидация, чтобы убедиться, что приложение соответствует требованиям, удобно в использовании и работает без сбоев.

Поиск технического решения основан на принципах оптимизации производительности, что гарантирует быструю работу приложения даже при увеличении количества записей. Особое внимание уделяется защите данных, включая механизмы резервного копирования и предотвращения несанкционированного доступа. Все эти принципы направлены на создание качественного, функционального и удобного инструмента для управления временем, который станет надежным помощником пользователя в повседневной жизни.

Пояснительная записка включает несколько разделов, каждый из которых посвящен решению определенных задач.

В разделе «Анализ существующих аналогов» рассматриваются другие электронные ежедневники, их преимущества и недостатки, что позволяет определить требования к разрабатываемому приложению.

В разделе «Постановке задачи» формулируются цели проекта, целевая аудитория, ключевые функциональные возможности и ограничения, которые необходимо учитывать при разработке.

Раздел «Разработка программного средства» описывает используемые структуры данных, архитектуру системы, алгоритмы решения задач и ключевые модули, обеспечивающие удобство и надежность приложения.

В разделе «Тестировании и проверке работоспособности» проводится оценка функциональности, выявляются ошибки и проверяется соответствие программы требованиям, что гарантирует ее стабильность.

Раздел «Руководство по использованию программы» содержит инструкции по настройке, эксплуатации и основным возможностям ежедневника, обеспечивая его удобство освоения пользователями.

В разделе «Заключение» подводятся итоги проделанной работы, оцениваются достигнутые результаты, их соответствие поставленным задачам, а также рассматриваются возможные направления дальнейшего совершенствования программного продукта.

1 АНАЛИЗ СУЩЕСТВУЮЩИХ АНАЛОГОВ

В условиях стремительного развития цифровых технологий и постоянного увеличения объема информации, с которой сталкивается современный человек, возрастает потребность в эффективной организации личного времени. Управление повседневными задачами, контроль за выполнением дел, планирование встреч и мероприятий — всё это требует системного подхода и надежных инструментов. Если ранее основным способом планирования были бумажные ежедневники, то сегодня всё большую популярность приобретают их электронные аналоги.

Электронные ежедневники стали неотъемлемой частью цифровой экосистемы, предлагая пользователям удобные и функциональные решения для планирования и самоорганизации. Благодаря широкому распространению смартфонов, планшетов и персональных компьютеров, такие приложения становятся доступными в любой момент времени и в любом месте, предоставляя быстрый доступ к информации, напоминаниям и спискам задач.

Кроме того, современные электронные планировщики играют важную роль не только в повседневной жизни, но и в профессиональной деятельности, образовательной среде, а также в сфере управления проектами. Они позволяют повысить продуктивность, сократить количество забытых дел и сформировать более осознанный подход к распределению времени.

На сегодняшний день на рынке представлено множество решений, среди которых можно выделить такие популярные приложения, как «ЛидерТаск», «Notion», «Any.do» и др.

ЛидерТаск — это многофункциональное российское приложение для управления задачами, которое сочетает в себе функции личного органайзера, таск-менеджера и планировщика. Пользователи могут работать с этим инструментом как онлайн, так и офлайн, что делает его удобным для различных условий использования. ЛидерТаск позволяет формировать списки задач, разделять их на подзадачи и группировать по приоритетам, а также делегировать их исполнение коллегам или выполнять самостоятельно. Преимуществами данного приложения являются:

* Возможность легкого переключения между различными видами календарного планирования.
* Синхронная работа на разных устройствах с мгновенным обновлением данных для всех участников.
* Встроенный корпоративный чат для обсуждения деталей проектов.
* Использование цветовых меток для организации задач.
* Расширенные возможности фильтрации.
* Возможность прикрепления к задачам аудио, заметок, комментариев, фотографий и документов.
* Автоматический перенос незавершенных задач на следующий день.
* Гибкие настройки для адаптации под нужды пользователя.

Недостатками данного приложения являются:

* Приложение может быть излишне сложным для пользователей, которым нужен более простой таск-менеджер.
* Некоторые пользователи могут столкнуться с перегрузкой функций, что усложняет быстрое освоение инструмента.
* Ограниченная функциональность в бесплатной версии по сравнению с платными тарифами. [4]

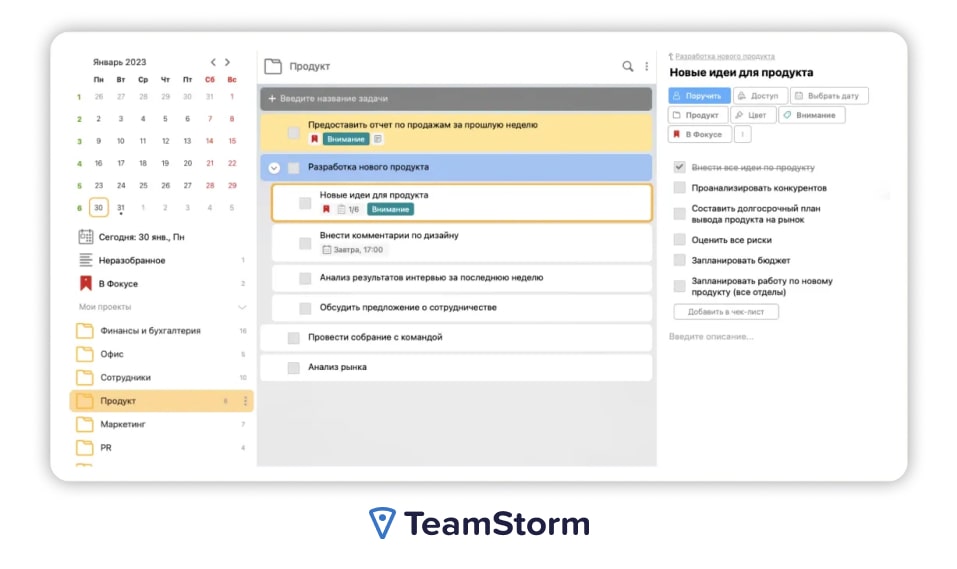


Рисунок 1 – Интерфейс программы «ЛидерТаск»

Следующим аналогом является «Notion» – это популярное приложение для организации информации, предоставляющее пользователям возможность создавать заметки, вести таблицы, формировать базы данных и сохранять ссылки из Интернета. Приложение включает около двадцати шаблонов для быстрого создания галерей, документов, чек-листов, инструкций и управления задачами через систему Kanban. Преимуществами данного приложения являются:

* Многофункциональность, делающая приложение пригодным для широкого спектра задач.
* Удобство в работе с документацией.
* Способность к комплексному сбору и хранению информации о проектах или продуктах.
* Простой и понятный интерфейс без излишних элементов.

Недостатками данного приложения являются:

* Некоторые продвинутые функции доступны только в рамках платных тарифов, ограничивая бесплатных пользователей в доступе к полному потенциалу приложения.
* Хотя интерфейс прост и интуитивно понятен, из-за широкого спектра функций новым пользователям может потребоваться время, чтобы полностью освоить все возможности Notion. [4]

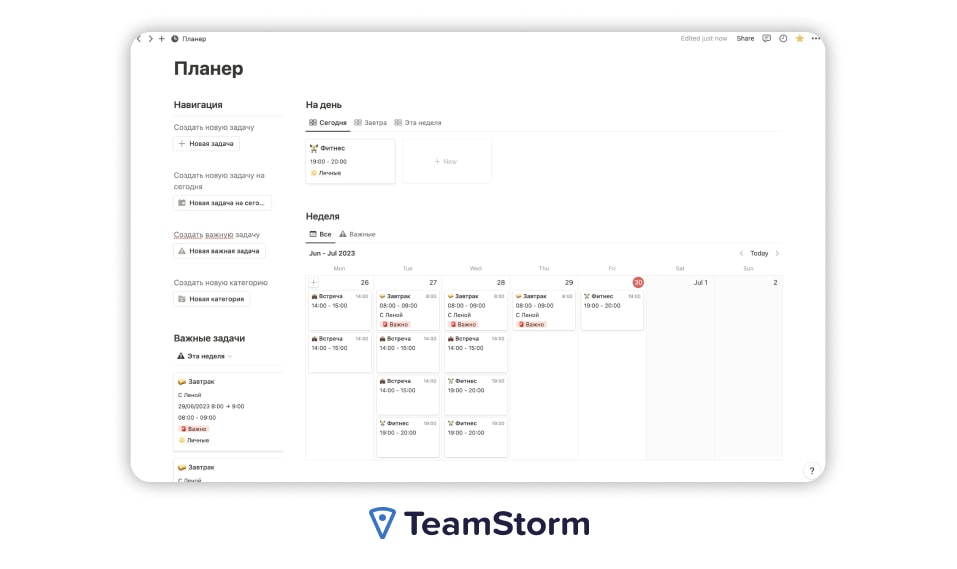


Рисунок 2 – Интерфейс программы «Notion»

Еще одним аналогом является «Any.do» – это приложение для управления задачами, которое отличается стильным дизайном и удобным интерфейсом. Оно включает в себя календарь на неделю, оформленный в виде доски, и списки задач, а также предлагает новый формат отображения, аналогичный Google Календарю. Преимуществами данного приложения являются:

* Современный дизайн и простота использования.
* Синхронизация с календарями Google и Outlook.
* Возможность интеграции с такими сервисами, как Slack, Gmail и WhatsApp.
* Разнообразие шаблонов для организации досок.
* Эстетически привлекательные фоны для досок, доступные в Premium версии.

Недостатками данного приложения являются:

* Недочеты в русской локализации: неправильное использование заглавных букв, ошибки в верстке, оставленные элементы на английском языке.
* Неясная необходимость функции «Мой календарь», которая дублирует функции Google Календаря.
* Ярко-синий цвет интерфейса, который может быть неприятен для глаз и не подлежит изменению.
* Назойливая реклама подписки Premium. [4]

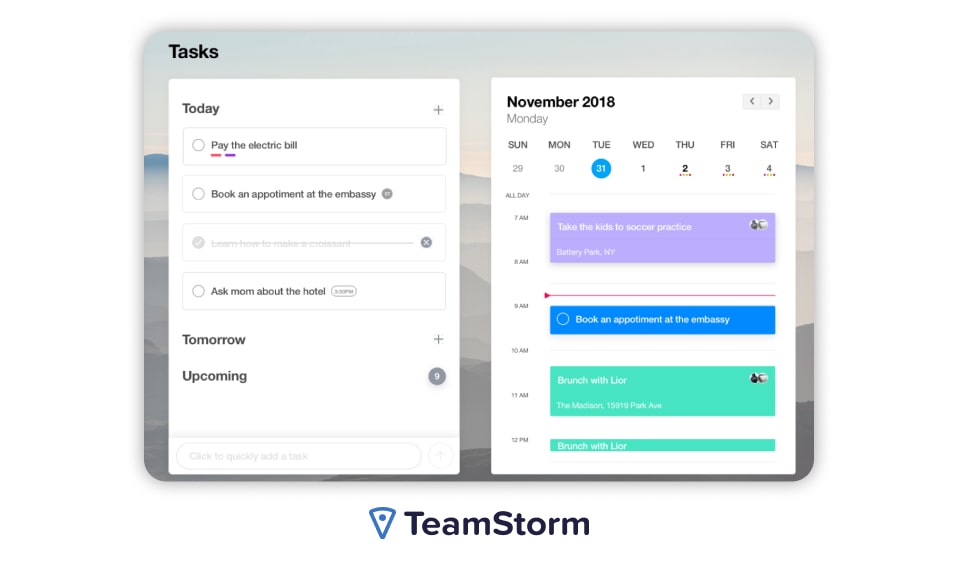


Рисунок 3 – Интерфейс программы «Any.do»

Анализ существующих аналогов позволяет выделить ключевые функции, востребованные пользователями: создание и редактирование задач, установка напоминаний, сортировка и фильтрация записей, возможность синхронизации между устройствами, а также наличие интуитивно понятного интерфейса. Эти возможности формируют базу ожиданий пользователей от любого современного ежедневника и становятся ориентиром при проектировании собственного решения.

Одновременно с этим выявлены и характерные недостатки существующих приложений, среди которых можно отметить перегруженность интерфейса, отсутствие офлайн-доступа, сложность первоначальной настройки, а также ограниченный функционал в бесплатных версиях. Нередко пользователи сталкиваются с необходимостью регистрации, привязки аккаунта или подписки, что снижает доступность и удобство работы с подобными сервисами. Кроме того, избыточная функциональность или сложная навигация могут отпугивать пользователей, предпочитающих простые и понятные решения.

Таким образом, изучение существующих программных решений в области электронного планирования служит не только основой для формирования функциональных требований, но и источником понимания пользовательских предпочтений, типичных проблем и путей их устранения.

1. ПОСТАНОВКА ЗАДАЧИ

Целью данного курсового проектирования является разработка современного электронного ежедневника с функциями ведения заметок и планирования задач в интерактивном календаре. Программное средство должно обеспечить удобную и безопасную работу пользователей с личной информацией, обеспечивая стабильное хранение и интуитивно понятный интерфейс. Функциональность приложения включает следующие ключевые блоки:

Аутентификация пользователя:

* система регистрации и входа с проверкой введённых данных;
* сохранение персональных заметок для каждого пользователя;
* защита маршрутов и данных от несанкционированного доступа.

Календарный интерфейс:

* интерактивный календарь с возможностью навигации по месяцам;
* визуальные индикаторы наличия заметок на определённые даты;
* корректное отображение дней в зависимости от месяца;
* выделение текущей даты и воскресений.

Управление заметками:

* создание и редактирование заметок для любой выбранной даты;
* удаление заметок с подтверждением действий;
* поддержка базового форматирования текста;
* автоматическое сохранение данных в локальное хранилище.

Современный пользовательский интерфейс:

* адаптивный дизайн, корректно отображающийся на различных экранах;
* простая и понятная навигация;
* плавные анимации и визуальные переходы;
* поддержка светлой и тёмной темы оформления;
* отображение состояния загрузки и обратная связь при действиях.

Разработка данного программного продукта направлена на создание простого, но функционального инструмента для повседневного использования, сочетающего удобство, безопасность и визуальную привлекательность.

Особое внимание будет уделено эргономике взаимодействия, логике пользовательского пути, а также устойчивости работы приложения. Программа должна быть комфортной в использовании, не перегруженной лишними элементами, а все действия — понятными на интуитивном уровне. Таким образом, создаваемое программное средство должно стать эффективным решением для планирования задач и заметок в личной или рабочей среде.

1. РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

## **Описание алгоритмов решения задачи**

Таблица 1 – Описание алгоритмов решения задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п.п. | Наименование алгоритма | Назначение алгоритма | Формальные параметры | Предлагаемый тип реализации |
| 1 | |  | | --- | |  |  |  | | --- | | Регистрация пользователя | | Осуществляет регистрацию нового пользователя с сохранением данных в локальное хранилище. Проверяет корректность ввода и наличие пользователя. Автоматически авторизует после успешной регистрации и перенаправляет на главную страницу. | email: string, password: string | React + useState, useNavigate, useLocalStorageState |
| 2 | Авторизация пользователя | Выполняет вход пользователя в систему с проверкой введённых данных и сохранением текущего пользователя. В случае успешной авторизации происходит переход к главной странице приложения. | email: string, password: string | React + useState, useNavigate, useLocalStorageState |
| 3 | Отображение информации "О приложении" | Представляет статичную справочную информацию о назначении приложения. Используется для |  | |  | | --- | |  |  |  | | --- | | React-страница без состояния, статический вывод текста | |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Отображение информации "О приложении" | информирования пользователя о функционале. |  |  |
| 4 | Календарь с заметками | Отображение календаря с возможностью добавления, редактирования и удаления заметок. Пользователь может кликать по дате для создания или изменения заметки, а также навигация по месяцам. | date: string, note: string | React + useState, useEffect, useLocalStorageState |
| 5 | Навигация по календарю | Может осуществляться переход на предыдущий месяц в календаре, переход на следующий месяц в календаре, получение дней месяца с учётом недель, начиная с понедельника, форматирование текущего месяца и года, обработка клика по дате, отображает модальное окно для добавления заметки. Использует алгоритмы: handlePreviousMonth, handleNextMonth, getDaysInMonth, formatMonthYear, handleDateClick |  | React + useNavigate + useState |
| 6 | Добавление заметки | Отображает модальное окно | note: string, date: string | React + useState |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | Добавление заметки | подтверждения сохранения заметки и сохраняет заметку для выбранной даты. Использует алгоритмы: saveNote, confirmSaveNote | note: string, date: string | React + useState |
| 7 | Изменение заметки | Открывает модальное окно редактирования заметки, подтверждает изменения в заметке и сохраняет изменения в заметке. Использует алгоритмы: saveEditedNote, handleEdit, confirmEditNote | editedNote: string, date: string | React + useState |
| 8 | Удаление заметки | Удаляет заметку для выбранной даты и подтверждает удаление заметки. Использует алгоритмы: deleteNote, confirmDelete | date: string | React + useState |
| 9 | Форматирование дат календаря | Форматирование дат с использованием русской локализации. Использует алгоритм: Date Format (Russian) | date: string | JavaScript |
| 10 | Обработка и сохранение данных | Обработка заметок и данных пользователся с помощью localStorage. Для больших объёмов данных может быть полезно оптимизировать | email: string, password: string, note: string, date: string | JavaScript |

## **Структура данных**

### 3.2.1 Структура типов программы

Каждая функция или переменная в коде имеет свое значение в управлении состоянием приложения и взаимодействии с пользователем. При разработке программного средства была использована динамические структуры данных – массивы (типа Array), объекты (типа Object), состояние компонента через useState и функции.

Таблица 2 – Структура типов программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| DiaryEntry | React компонент | Отображает заметку с возможностью редактирования и удаления. Включает дату, заметку и кнопки для действий (редактировать, удалить). |
| date | string | Строка, представляющая дату заметки в формате yyyy-MM-dd. |
| note | string | undefined | Текст заметки. Может быть пустым или отсутствовать. |
| onEdit | () => void | Функция для обработки редактирования заметки. |
| onDelete | () => void | Функция для обработки удаления заметки. |
| HomePage | React компонент | Основная страница дневника с календарём и списком заметок. |
| currentUser | { email: string } | null | Объект текущего пользователя с его email или null, если пользователь не авторизован. |
| currentMonth | Date | Текущая выбранная дата (месяц). |
| selectedDate | string | Строка с выбранной датой (формат yyyy-MM-dd). |
| |  | | --- | | allNotes |  |  | | --- | |  | | Record<string, Record<string, string>> | Объект, где ключ — это email пользователя, а значение — объект с заметками, где ключ — это дата, а значение — текст заметки. |
| newNote | string | Текст новой заметки, который вводит пользователь. |
| showAddNoteModal | boolean | Флаг для отображения модального окна добавления заметки. |

Продолжение таблицы 2

|  |  |  |
| --- | --- | --- |
| showConfirmationModal | boolean | Флаг для отображения окна подтверждения сохранения заметки. |
| editDate | string | null | Дата, для которой редактируется заметка. Если редактирования нет – значение null. |
| editNote | string | Новый текст заметки при редактировании. |
| showEditModal | boolean | Флаг для отображения модального окна редактирования заметки. |
| dateToDelete | string | null | Дата заметки, которую нужно удалить. Если заметка не выбрана для удаления, то значение null. |
| showEditConfirmationModal | boolean | Флаг для отображения окна подтверждения изменений при редактировании заметки. |
| editPendingNote | { date: string, note: string } | null | Объект, представляющий заметку, которая находится на стадии редактирования. |
| userNotes | Record<string, string> | Заметки текущего пользователя, где ключ — дата, значение — текст заметки. |
| handleLogout | () => void | Функция для выхода из аккаунта и перенаправления на страницу входа. |
| handlePreviousMonth | () => void | Функция для перехода на предыдущий месяц. |
| handleNextMonth | () => void | Функция для перехода на следующий месяц. |
| getDaysInMonth | (date: Date) => Date[] | Функция, которая возвращает список всех дней месяца, включая дни предыдущего и следующего месяца для корректного отображения в календаре. |
| formatMonthYear | (date: Date) => string | Функция, которая форматирует месяц и год в строку в формате "Месяц Год". |
| currentMonthDays | Date[] | Массив всех дней текущего месяца, который отображается в календаре. |

Продолжение таблицы 2

|  |  |  |
| --- | --- | --- |
| handleDateClick | (date: Date) => void | Функция для обработки клика по дате в календаре. Открывает модальное окно для добавления или редактирования заметки. |
| saveNote | () => void | Функция для сохранения новой заметки с подтверждением. |
| confirmSaveNote | () => void | Функция для подтверждения сохранения заметки в хранилище. |
| deleteNote | (date: string) => void | Функция для удаления заметки по выбранной дате. |
| isWeekend | (date: Date) => boolean | Функция для проверки, является ли день выходным (суббота или воскресенье). |
| handleEdit | (date: string) => void | Функция для начала редактирования заметки. |
| saveEditedNote | |  | | --- | |  |   () => void | Функция для сохранения редактируемой заметки. |
| confirmEditNote | |  | | --- | |  |   () => void | Функция для подтверждения изменений в заметке после редактирования. |
| confirmDelete | |  | | --- | |  |   () => void | Функция для подтверждения удаления заметки. |

### Структура данных программы

Таблица 3 – Структура данных программы

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Комментарии |
| KAutorization | TAutorization | Обеспечение графического интерфейса во время авторизации | Главная форма программы |
| KRegistrate | TRegistrate | Обеспечение графического интерфейса во время регистрации | Вспомогательная форма программы |
| KEntry | TEntry | Обеспечение графического интерфейса во время входа | Вспомогательная форма программы |
| KNote | TNote | Обеспечение графического интерфейса для хранения заметок пользователя | Вспомогательная форма программы |
| KCalendar | TCalendar | Обеспечение графического интерфейса для организации календаря | Вспомогательная форма программы |
| KUser | TUser | Обеспечение графического интерфейса для хранения данных о пользователе | Вспомогательная форма программы |
| KLogout | TLogout | Обеспечение графического интерфейса во время выхода пользователя из приложения | Вспомогательная форма программы |
| KAddNote | TAddNote | Обеспечение графического интерфейса для | Вспомогательная форма программы |

Продолжение таблицы 3

|  |  |  |  |
| --- | --- | --- | --- |
| KAddNote | TAddNote | добавления новой заметки | Вспомогательная форма программы |
| KEditNote | TEditNote | Обеспечение графического интерфейса для редактирования существующей заметки | Вспомогательная форма программы |
| KDeleteNote | TDeleteNote | Обеспечение графического интерфейса для удаления заметки | Вспомогательная форма программы |
| KSwitchCalendar | TSwitchCalendar | Обеспечение графического интерфейса для переключения между месяцами календаря | Вспомогательная форма программы |
| KSelectDate | TSelectDate | Обеспечение графического интерфейса для выбора даты на календаре | Вспомогательная форма программы |
| KGoToToday | TGoToToday | Обеспечение графического интерфейса для перехода к сегодняшней дате на календаре | Вспомогательная форма программы |
| KConfirmAction | TConfirmAction | Обеспечение графического интерфейса для подтверждений действий над заметкой | Вспомогательная форма программы |
| KHighlightWeekends | THighlightWeekends | Обеспечение графического интерфейса для выделение выходных дней на календаре | Вспомогательная форма программы |

### Структура данных алгоритма LoginPage

Таблица 4 – Структура данных алгоритма LoginPage

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| email | TMyString | Адрес электронной почты пользователя для входа | Формальный |
| password | TMyString | Пароль пользователя для входа | Формальный |
| error | TMyString | Сообщение об ошибке, если вход не удался | Локальный |
| isLoading | TBool | Индикатор загрузки, показывающий состояние загрузки страницы | Локальный |
| users | TArray<TUser> | Массив пользователей, сохраненных в localStorage | Локальный |
| setCurrentUser | TFunction | Функция для сохранения текущего пользователя в localStorage | Локальный |
| navigate | TFunction | Функция для перехода на другие страницы | Локальный |
| handleSubmit | TFunction | Функция, обрабатывающая отправку формы входа и валидацию | Локальный |
| user | TUser | Объект, представляющий данные пользователя для аутентификации | Локальный |

### Структура данных алгоритма RegisterPage

Таблица 5 – Структура данных алгоритма RegisterPage

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| email | TMyString | Адрес электронной почты пользователя для регистрации | Формальный |
| password | TMyString | Пароль пользователя для регистрации | Формальный |
| error | TMyString | Сообщение об ошибке при регистрации | Локальный |
| isLoading | TBool | Индикатор загрузки, показывающий состоя- | Локальный |

Продолжение таблицы 5

|  |  |  |  |
| --- | --- | --- | --- |
| isLoading | TBool | ние загрузки страницы | Локальный |
| users | TArray<TUser> | Массив пользователей, сохраненных в localStorage | Локальный |
| setUsers | TFunction | Функция для сохранения списка пользователей в localStorage | Локальный |
| setCurrentUser | TFunction | Функция для сохранения текущего пользователя в localStorage | Локальный |
| navigate | TFunction | Функция для перехода на другие страницы | Локальный |
| handleSubmit | TFunction | |  | | --- | |  |  |  | | --- | | Функция, обрабатывающая отправку формы регистрации | | Локальный |
| userExists | TBool | Булевое значение, показывающее, существует ли пользователь | Локальный |

### Структура данных алгоритма AboutPage

Таблица 6 – Структура данных алгоритма AboutPage

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| textContent | TMyString | Основной текст, описывающий приложение | Локальный |
| title | TMyString | Заголовок страницы «О приложении» | Локальный |

### Структура данных алгоритма handlePreviousMonth

Таблица 7 – Структура данных алгоритма handlePreviousMonth

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| currentMonth | TMyDate | Текущий месяц, отображаемый в календаре | Локальный |
| updatedMonth | TMyDate | Обновлённый месяц после перехода на предыдущий месяц | Локальный |
| currentYear | TMyNumber | Текущий год, отображаемый в календаре | Локальный |
| updatedYear | TMyNumber | Обновлённый год после перехода на преды- | Локальный |

Продолжение таблицы 7

|  |  |  |  |
| --- | --- | --- | --- |
| updatedYear | TMyNumber | дущий месяц | Локальный |

### Структура данных алгоритма handleNextMonth

Таблица 8 – Структура данных алгоритма handleNextMonth

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| currentMonth | TMyDate | Текущий месяц, отображаемый в календаре | Локальный |
| updatedMonth | TMyDate | Обновлённый месяц после перехода на следующий месяц | Локальный |
| currentYear | TMyNumber | Текущий год, отображаемый в календаре | Локальный |
| updatedYear | TMyNumber | Обновлённый год после перехода на следующий месяц | Локальный |

### Структура данных алгоритма getDaysInMonth

Таблица 9 – Структура данных алгоритма getDaysInMonth

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| year | TMyNumber | Год, для которого нужно получить количество дней в месяце | Формальный |
| month | TMyNumber | Месяц, для которого нужно получить количество дней | Формальный |
| daysInMonth | TMyNumber | Количество дней в указанном месяце | Локальный |

### Структура данных алгоритма formatMonthYear

Таблица 10 – Структура данных алгоритма formatMonthYear

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| year | TMyNumber | Год, который необходимо отформатировать в строку | Формальный |
| month | TMyNumber | Месяц, который необходимо отформатировать в строку | Формальный |
| formattedDate | TMyString | Строка с отформатиро- | Локальный |
| formattedDate | TMyString | ванным месяцем и годом | Локальный |

### Структура данных алгоритма handleDateClick

Таблица 11 – Структура данных алгоритма handleDateClick

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| year | TMyNumber | Год, в котором находится выбранный день | Формальный |
| month | TMyNumber | Месяц, в котором находится выбранный день | Формальный |
| day | TMyNumber | День месяца, по которому был произведен клик | Формальный |
| selectedDate | TMyDate | Дата, которая была выбрана пользователем (обновленная) | Локальный |

### Структура данных алгоритма saveNote

Таблица 12 – Структура данных алгоритма saveNote

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| noteContent | TMyString | Содержимое заметки, которое пользователь хочет сохранить | Формальный |
| noteDate | TMyDate | Дата, к которой относится заметка (например, выбранная дата на календаре) | Формальный |
| notes | TMyArray | Массив всех сохраненных заметок в приложении | Локальный |

### Структура данных алгоритма confirmSaveNote

Таблица 13 – Структура данных алгоритма confirmSaveNote

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| message | TMyString | Сообщение, подтверждающее успешность или ошибку при сохранении заметки | Формальный |

Продолжение таблицы 13

|  |  |  |  |
| --- | --- | --- | --- |
| status | TMyBoolean | Статус успешности сохранения заметки (true — успешно, false — ошибка) | Локальный |

### Структура данных алгоритма saveEditedNote

Таблица 14 – Структура данных алгоритма saveEditedNote

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| editedNoteContent | TMyString | Содержимое отредактированной заметки | Формальный |
| noteId | TMyNumber | Идентификатор заметки, которую нужно отредактировать | Формальный |
| notes | TMyArray | Массив всех заметок в приложении | Локальный |

### Структура данных алгоритма handleEdit

Таблица 15 – Структура данных алгоритма handleEdit

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| noteContent | TMyString | Содержимое заметки, которое будет отображено в поле редактирования | Локальный |
| noteId | TMyNumber | Идентификатор заметки, которую нужно отредактировать | Формальный |

### Структура данных алгоритма confirmEditNote

Таблица 16 – Структура данных алгоритма confirmEditNote

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| editedNoteContent | TMyString | Содержимое отредактированной заметки | Формальный |
| noteId | TMyNumber | Идентификатор заметки, которую нужно отредактировать и подтвердить изменения | Формальный |
| notes | TMyArray | Массив всех заметок в приложении | Локальный |

### Структура данных алгоритма deleteNote

Таблица 17 – Структура данных алгоритма deleteNote

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| notes | |  | | --- | | TMyArray |  |  | | --- | |  | | Массив всех заметок в приложении | Локальный |
| noteId | TMyNumber | Идентификатор заметки, которую нужно удалить | Формальный |

### Структура данных алгоритма confirmDelete

Таблица 18 – Структура данных алгоритма confirmDelete

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| notes | |  | | --- | | TMyArray |  |  | | --- | |  | | Массив всех заметок в приложении | Локальный |
| noteId | TMyNumber | Идентификатор заметки, которую нужно удалить | Формальный |

### Структура данных алгоритма useLocalStorageState

Таблица 19 – Структура данных алгоритма useLocalStorageState

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| key | |  | | --- | | TMyString |  |  | | --- | |  | | Ключ для сохранения данных в localStorage | Формальный |
| initialState | TMyArray / TMyObject | Начальное значение, если данные в localStorage отсутствуют | Формальный |
| [state, setState] | [TMyState, Function] | Массив с состоянием и функцией для его обновления | Локальный |

## **Схема алгоритмов решения задач по ГОСТ 19.701-90**

### 3.3.1 Схема алгоритма LoginPage

Функция LoginPage реализует процесс авторизации пользователя в приложении с использованием email и пароля. Она предоставляет форму для ввода данных, проверяет их на валидность, выполняет симуляцию запроса к серверу и, в случае успешной аутентификации, сохраняет текущего пользователя в localStorage и перенаправляет его на главную страницу. При первом рендере компонента создаются четыре состояния: email, password, error и isLoading. Состояния email и password отвечают за хранение введённых данных пользователем. Состояние error используется для вывода сообщения об ошибке, если что-то пошло не так. Состояние isLoading используется для отображения состояния загрузки, когда происходит процесс авторизации. Затем компонент с помощью хука useLocalStorageState получает из localStorage список всех зарегистрированных пользователей, используя ключ 'users'. Этот список необходим для проверки введённых данных в процессе аутентификации. Также с помощью хука useLocalStorageState создаётся или обновляется состояние текущего пользователя (ключ 'currentUser'), который будет сохраняться в localStorage после успешной авторизации.

Когда пользователь отправляет форму, вызывается функция handleSubmit, которая выполняет все действия для авторизации. Внутри этой функции сначала проверяется, были ли заполнены оба поля — email и пароль. Если какое-то из них пустое, то функция сразу же выводит ошибку о том, что необходимо заполнить все поля, и завершает выполнение.

Если оба поля заполнены, выполняется симуляция API-запроса с помощью setTimeout, чтобы создать задержку, имитируя работу сервера. После задержки программа проверяет, существует ли пользователь с введённым email. Если такого пользователя нет или пароль не совпадает с сохранённым, то выводится ошибка "Неправильный email или пароль". В случае успешной проверки — данные текущего пользователя сохраняются в localStorage (ключ 'currentUser'), и пользователь перенаправляется на главную страницу приложения с помощью navigate('/').

В результате работы компонента, если авторизация прошла успешно, пользователь автоматически переходит на главную страницу приложения, где ему будет доступен весь функционал, связанный с личным дневником. В случае ошибки, пользователю показывается соответствующее сообщение, и он может повторить попытку ввода данных.

Таким образом, функция LoginPage обеспечивает процесс аутентификации пользователя, включая проверку введённых данных, симуляцию запроса к серверу, сохранение данных о текущем пользователе и перенаправление на главную страницу.

### 3.3.2 Схема алгоритма RegisterPage

Функция RegisterPage предназначена для регистрации нового пользователя в приложении. Этот компонент отображает форму регистрации и обрабатывает ввод данных, их валидацию и сохранение нового пользователя в локальном хранилище.

Когда компонент RegisterPage монтируется, он инициализирует несколько состояний с помощью хука useState: email для хранения введённого пользователем адреса электронной почты, password для хранения введённого пароля, error для отображения ошибок, возникших при регистрации, и isLoading для отображения состояния загрузки, чтобы заблокировать интерфейс и предотвратить повторные действия.

Для работы с данными пользователей используется хук useLocalStorageState, который позволяет получить список уже зарегистрированных пользователей (users) и установить состояние (setUsers). Это обеспечивает сохранение данных в localStorage, что позволяет пользователям сохранять информацию при перезагрузке страницы.

При отправке формы вызывается функция handleSubmit, которая сначала проверяет, что все обязательные поля заполнены. Если одно из полей пустое, пользователю показывается ошибка. Далее происходит валидация пароля: если он менее 6 символов, появляется сообщение с требованием увеличить его длину. Затем выполняется проверка на уникальность email — если такой email уже существует в списке пользователей, выводится соответствующее сообщение об ошибке. После этого происходит симуляция задержки, чтобы создать ощущение взаимодействия с сервером, после чего, если все проверки пройдены, новый пользователь добавляется в список зарегистрированных пользователей.

После успешной регистрации текущий пользователь сохраняется в localStorage с помощью хука setCurrentUser, что автоматически авторизует его. Затем пользователь перенаправляется на главную страницу с помощью хука navigate. Если при регистрации возникли ошибки, компонент выводит сообщение об ошибке, чтобы пользователь мог внести необходимые исправления.

Таким образом, RegisterPage представляет собой систему регистрации, которая включает в себя валидацию данных, предотвращение дублирования, а также автоматический вход после успешной регистрации.

### 3.3.3 Схема алгоритма AboutPage

AboutPage – это компонент, предназначенный для отображения информации о приложении. В данном случае он служит для показа модального окна с кратким описанием функционала и назначения приложения.

Компонент AboutPage не имеет динамического состояния или логики обработки данных. Он просто рендерит информацию о приложении в виде текста на странице. При рендеринге компонент показывает заголовок, который приветствует пользователя, и основной текст, объясняющий назначение и возможности приложения. В тексте сообщается, что приложение предназначено для ведения ежедневных записей, позволяет создавать заметки и взаимодействовать с календарём.

Этот компонент также использует стандартные стили приложения для оформления и размещения содержимого на странице. Структура компонента проста и интуитивно понятна: блок с основным текстом, который располагается вверху модального окна и стильными шрифтами. Используются классы Tailwind CSS для адаптивного дизайна, чтобы модальное окно было интуитивно понятным и содержало необходимую информацию и функционал.

Таким образом, AboutPage является информационным экраном, который не выполняет сложных функций, а лишь предоставляет пользователю информацию о приложении.

### 3.3.4 Схема алгоритма handlePreviousMonth

Функция handlePreviousMonth отвечает за изменение текущего отображаемого месяца в календаре на предыдущий.

При вызове функция обновляет состояние текущего месяца, уменьшая его на один. Для этого она использует объект состояния, который хранит информацию о текущем месяце и году. Функция изменяет значение месяца, при этом если месяц уменьшился до 0 (то есть был январь), то она переходит на декабрь предыдущего года. Таким образом, при переходе с января на декабрь она также корректирует год, уменьшив его на 1.

Этот функционал используется в интерфейсе календаря для обеспечения навигации между месяцами. Пользователь может переходить назад по календарю, к предыдущим месяцам, что делает использование календаря более гибким и удобным.

При изменении состояния компонента автоматически происходит рендеринг нового месяца, что позволяет динамически обновлять отображаемый календарь, сохраняя правильность данных (например, количество дней в месяце).

### 3.3.5 Схема алгоритма handleNextMonth

Функция handleNextMonth отвечает за изменение текущего отображаемого месяца в календаре на следующий.

При вызове функции она обновляет состояние текущего месяца, увеличивая его на единицу. Если текущий месяц — декабрь (то есть 12-й месяц), функция устанавливает месяц в январь следующего года (1-й месяц) и увеличивает год на 1. В противном случае просто увеличивается значение месяца, что позволяет переходить к следующему месяцу в текущем году.

Этот механизм используется в интерфейсе календаря для обеспечения навигации между месяцами. Пользователь может легко перейти к следующему месяцу, что делает использование календаря более интуитивно понятным и удобным.

После изменения состояния компонент автоматически перерисовывает календарь, обновляя отображение соответствующего месяца с учетом всех изменений.

### 3.3.6 Схема алгоритма getDaysInMonth

Функция getDaysInMonth используется для определения количества дней в текущем месяце. Она принимает два параметра: год и месяц, и возвращает число дней, которые присутствуют в этом месяце.

Функция работает следующим образом: сначала она создает объект Date, устанавливая его на первый день следующего месяца, и затем с помощью метода getDate() получает последний день предыдущего месяца. Это дает точное количество дней в текущем месяце, независимо от того, является ли месяц коротким (например, февраль) или длинным.

Эта функция используется в календаре для отображения всех дней в месяце. Она гарантирует, что календарь всегда будет корректно отображать правильное количество дней в зависимости от месяца и года.

### 3.3.7 Схема алгоритма formatMonthYear

Функция formatMonthYear предназначена для форматирования текущего месяца и года в строку, которая используется для отображения на календаре. В качестве входных данных она принимает два параметра: месяц, который представлен целым числом в диапазоне от 0 до 11, и год, представленный числом. Месяцы в JavaScript нумеруются с 0, то есть январь имеет значение 0, февраль — 1 и так далее.

Процесс работы функции начинается с того, что она использует массив строк, содержащих названия месяцев на русском языке. Месяцы в этом массиве расположены в порядке от января до декабря, что позволяет легко получить название месяца по его индексу, переданному в параметре функции. Например, если передан параметр 0 (январь), то из массива будет извлечено название месяца "Январь".

После того как название месяца найдено, функция объединяет его с переданным годом в строку, формируя строку вида "Месяц Год", например "Июль 2025". Эта строка затем возвращается функцией и используется для отображения в заголовке календаря на интерфейсе пользователя. Таким образом, пользователи видят, какой месяц и год отображаются в календаре, что позволяет им ориентироваться в текущем времени.

Данная функция является важной частью отображения даты в интерфейсе, обеспечивая корректное и удобное восприятие информации о текущем времени для пользователя.

### 3.3.8 Схема алгоритма handleDateClick

Функция handleDateClick обрабатывает клики пользователя по датам в календаре. Когда пользователь нажимает на дату в календаре, эта функция выполняет несколько важных действий, связанных с выбором конкретной даты.

Функция принимает параметр, который представляет собой объект с данными о выбранной дате. Обычно этот объект содержит такие свойства, как день, месяц и год, а также может включать дополнительные данные, например, информацию о том, является ли эта дата текущим днем или выходным. В зависимости от того, какая дата была выбрана, функция выполняет соответствующие действия.

Основной задачей этой функции является обновление состояния приложения, чтобы отразить выбор пользователем даты. В частности, она может устанавливать выбранную дату как активную для дальнейшей работы, например, для отображения заметок, записанных на этот день, или других данных, связанных с датой. Функция может использовать состояние компонента для обновления UI и вызова других действий, таких как фильтрация заметок по выбранной дате.

После того как пользователь кликает на дату, функция может также инициировать дополнительные действия, такие как отображение модального окна для создания новой заметки на выбранную дату или изменения уже существующих записей. Это позволяет пользователю взаимодействовать с приложением, добавляя или редактируя информацию, связанную с конкретным днем.

Таким образом, handleDateClick является ключевым элементом в пользовательском интерфейсе, обеспечивая интерактивность календаря и связывая его с остальной частью приложения, например, с функциями заметок.

### 3.3.9 Схема алгоритма saveNote

Функция saveNote предназначена для сохранения новой заметки, созданной пользователем, в приложении. Когда пользователь завершает создание заметки, эта функция обрабатывает все необходимые действия для её сохранения в хранилище данных, например, в localStorage или в состоянии приложения, в зависимости от реализации.

Функция принимает объект с данными заметки, которые пользователь ввел в форму. Это могут быть такие данные, как текст заметки, дата, на которую она была создана, и возможно, дополнительные параметры, такие как метки или приоритет. Важной частью функции является проверка, что все обязательные поля заполнены корректно. Например, функция может проверять, что текст заметки не пуст, и если это так, выбрасывать ошибку или показывать пользователю сообщение об этом.

После того как все данные прошли проверку, функция сохраняет заметку в соответствующую структуру данных. В случае с использованием localStorage, это может быть сохранение в локальное хранилище браузера, чтобы заметки были доступны даже после перезагрузки страницы. В случае использования состояния приложения, функция обновляет состояние, добавляя новую заметку в список текущих записей.

Сохранение заметки также может сопровождаться отображением уведомления пользователю, подтверждающего успешное добавление, или редиректом на страницу с заметками. Важно, что saveNote работает таким образом, чтобы сохранить все данные и гарантировать, что заметка будет отображаться на соответствующей дате или в списке записей, в зависимости от реализации календаря и интерфейса приложения.

Таким образом, saveNote является основной функцией для сохранения заметок, обеспечивая их доступность и правильное отображение в интерфейсе приложения.

### 3.3.10 Схема алгоритма confirmSaveNote

Функция confirmSaveNote подтверждает сохранение новой заметки после того, как пользователь ввел все необходимые данные и инициировал процесс сохранения. Она используется для того, чтобы предоставить пользователю возможность удостовериться, что все введенные данные верны, перед окончательным сохранением заметки.

Когда пользователь нажимает кнопку для сохранения заметки, confirmSaveNote сначала проверяет, все ли данные введены корректно и нет ли ошибок в процессе. Например, если пользователь забыл ввести текст заметки или выбрал некорректную дату, функция может отобразить сообщение об ошибке или предупредить о необходимости заполнить обязательные поля.

Одной из важнейших задач confirmSaveNote является обработка возможных действий отмены. В случае если пользователь решит отказаться от сохранения или не уверен в своих действиях, функция предоставляет возможность отменить операцию, возвращая пользователя на предыдущий экран или предоставляя ему другие опции для редактирования данных заметки.

После подтверждения всех данных и завершения всех проверок, функция вызывает функцию сохранения заметки, которая сохраняет данные в хранилище (например, в localStorage, или обновляет состояние приложения). Это может сопровождаться визуальным подтверждением успешного сохранения заметки, таким как уведомление о том, что заметка была успешно добавлена, или с редиректом на страницу с календарем, где заметка теперь будет отображаться.

Таким образом, confirmSaveNote играет ключевую роль в процессе сохранения данных, предоставляя пользователю возможность подтвердить свои действия перед окончательным сохранением, что снижает вероятность ошибок и улучшает пользовательский опыт.

### 3.3.11 Схема алгоритма saveEditedNote

Функция saveEditedNote отвечает за сохранение изменений, внесенных в уже существующую заметку. Она используется после того, как пользователь редактирует содержание заметки и подтверждает свое намерение сохранить изменения.

Когда пользователь открывает заметку для редактирования, она загружает данные этой заметки в соответствующие поля ввода. После внесения изменений и нажатия на кнопку сохранения, saveEditedNote выполняет несколько ключевых действий для обработки данных.

Первоначально, функция проверяет, были ли изменения в тексте заметки. Это важно, поскольку если изменения не были сделаны, то не нужно выполнять никаких операций по сохранению. В случае, если изменения есть, функция проверяет их на соответствие требованиям, таким как обязательные поля или формат данных (например, наличие текста или корректное заполнение даты). Если данные не проходят проверку, отображается сообщение об ошибке, и пользователь уведомляется о необходимости исправить данные.

После успешной проверки данных и их корректности, saveEditedNote обновляет информацию о заметке в хранилище. Это может быть выполнено путем обновления состояния приложения или записи в localStorage, если используется локальное хранилище. Важно, чтобы при сохранении изменений старое значение заметки было заменено новым, чтобы пользователю не пришлось работать с устаревшей информацией.

После успешного сохранения функции могут следовать визуальные подтверждения: уведомление о том, что изменения были сохранены, или обновление интерфейса, чтобы отобразить актуальную версию заметки. В некоторых случаях после сохранения заметки может произойти перенаправление пользователя на предыдущую страницу (например, на календарь), где изменения будут сразу же отображаться.

Таким образом, saveEditedNote обеспечивает сохранение внесенных изменений в заметку, гарантируя, что данные корректно обновляются и сохраняются, а также уведомляя пользователя об успешном завершении операции.

### 3.3.12 Схема алгоритма handleEdit

Функция handleEdit предназначена для обработки действия редактирования существующей заметки. Она вызывается, когда пользователь решает изменить уже сохраненную заметку, например, при нажатии кнопки редактирования, отображенной на интерфейсе.

При вызове этой функции, она выполняет несколько шагов для подготовки заметки к редактированию. Во-первых, handleEdit получает идентификатор заметки, которую нужно отредактировать. Этот идентификатор используется для поиска и извлечения содержимого заметки из хранилища данных (например, из localStorage или состояния приложения). Важно, чтобы система могла точно идентифицировать нужную заметку среди других, чтобы подгрузить именно ту, которую пользователь хочет отредактировать.

Затем handleEdit устанавливает данные заметки в поля ввода, подготовленные для редактирования, таким образом, чтобы пользователь мог сразу начать вносить изменения. Это может включать текст заметки, дату и другие параметры, которые были сохранены ранее. Когда заметка загружена в интерфейс, пользователь может увидеть актуальную информацию и приступить к редактированию.

Кроме того, handleEdit может обновить состояние интерфейса, чтобы отобразить форму редактирования вместо обычного просмотра заметки. Таким образом, процесс редактирования будет интуитивно понятным для пользователя, предоставляя ему соответствующие поля для ввода и кнопки для подтверждения изменений или отмены редактирования.

После того как пользователь внесет все изменения и подтвердит их (например, нажав на кнопку "Сохранить"), данные, введенные пользователем, будут переданы функции saveEditedNote для дальнейшей обработки и сохранения обновленной информации в хранилище. Если же пользователь решит отменить редактирование, можно вернуть форму в исходное состояние без изменений, обеспечивая возможность отказаться от редактирования.

Таким образом, handleEdit служит для того, чтобы инициировать процесс редактирования заметки, подготовить данные для изменения и отобразить соответствующий интерфейс для пользователя

### 3.3.13 Схема алгоритма confirmEditNote

Функция confirmEditNote предназначена для подтверждения изменений, внесенных пользователем в заметку во время редактирования. Она вызывается, когда пользователь завершает редактирование и нажимает кнопку подтверждения изменений, например, "Сохранить изменения".

Когда confirmEditNote активируется, она получает данные, введенные пользователем в поля формы редактирования, включая текст заметки и другие параметры, если они были изменены. Затем эта функция выполняет несколько важных шагов для корректного сохранения этих данных.

Сначала confirmEditNote проверяет, что все необходимые поля были правильно заполнены. Если какие-либо обязательные данные отсутствуют или введены некорректно, функция может отобразить ошибку и предупредить пользователя, что некоторые поля должны быть заполнены, прежде чем изменения можно будет сохранить.

После успешной проверки данных, confirmEditNote обновляет заметку в хранилище (например, в localStorage, если данные сохраняются локально, или в состоянии приложения). Функция ищет нужную заметку по уникальному идентификатору, который был передан при вызове функции редактирования. Это позволяет точно определить, какую запись нужно обновить, и сохраняет новые данные в правильном месте.

Если все прошло успешно и данные были сохранены, функция обычно перенаправляет пользователя на страницу с обновленным списком заметок или возвращает его на страницу с дневником, где уже отображается обновленная информация.

В случае ошибки, например, если возникли проблемы при сохранении данных или при работе с хранилищем, confirmEditNote может вывести уведомление о возникшей проблеме и предложить пользователю повторить попытку.

Таким образом, confirmEditNote играет важную роль в процессе сохранения изменений в заметке, обеспечивая точность данных и корректность их хранения.

### 3.3.14 Схема алгоритма deleteNote

Функция deleteNote предназначена для удаления заметки из приложения. Она вызывается, когда пользователь решает удалить одну из своих заметок. Обычно этот процесс инициируется через интерфейс приложения, когда пользователь нажимает кнопку или ссылку "Удалить" рядом с заметкой.

При вызове deleteNote, функция получает уникальный идентификатор заметки, которую нужно удалить. Это может быть ID заметки, переданный из интерфейса пользователя, который позволяет точно определить, какую заметку необходимо удалить из хранилища данных, будь то локальное хранилище или состояние приложения.

После того как deleteNote получает необходимый идентификатор, она проверяет наличие соответствующей заметки в хранилище. Если заметка найдена, функция удаляет ее из списка заметок. Это может быть выполнено с помощью метода удаления в хранилище данных (например, localStorage.removeItem() или удаление элемента из массива в состоянии приложения).

Кроме того, перед фактическим удалением, deleteNote может запросить подтверждение от пользователя. Это необходимо, чтобы избежать случайных удалений данных, особенно если заметка содержит важную информацию. Обычно после вызова deleteNote показывается модальное окно с запросом "Вы уверены, что хотите удалить эту заметку?" и кнопками для подтверждения или отмены действия.

Если пользователь подтверждает удаление, функция окончательно удаляет заметку. В противном случае, если пользователь отменяет действие, заметка остаётся в хранилище и никаких изменений не происходит.

После успешного удаления заметки deleteNote может обновить интерфейс, чтобы пользователь увидел актуальную информацию (например, заметка исчезает из списка). Если удаление прошло с ошибками, например, если заметка не была найдена или произошла проблема с доступом к хранилищу, deleteNote может вывести уведомление об ошибке и предложить пользователю повторить попытку.

Таким образом, функция deleteNote отвечает за удаление заметки из системы, обеспечивая как защиту от случайных действий, так и надежное выполнение операции удаления.

### 3.3.15 Схема алгоритма confirmDelete

Функция confirmDelete — это функция, предназначенная для подтверждения удаления заметки пользователем. Когда пользователь инициирует удаление заметки, confirmDelete становится частью процесса, обеспечивая механизм защиты от случайных удалений.

Когда пользователь нажимает кнопку для удаления заметки, вызывается функция, которая в свою очередь вызывает confirmDelete. Основная цель этой функции — запросить у пользователя подтверждение на удаление выбранной заметки. Это важный шаг, который позволяет избежать ошибок и случайных удалений, особенно если заметка содержит важную информацию.

confirmDelete отображает модальное окно или всплывающее уведомление, в котором содержится предупреждение, например: "Вы уверены, что хотите удалить эту заметку?" В этом окне есть две кнопки: одна для подтверждения удаления ("Да, удалить"), и другая для отмены действия ("Отмена"). Это дает пользователю возможность обдумать свои действия перед окончательным удалением заметки.

Если пользователь нажимает кнопку "Да, удалить", confirmDelete передает управление функции, которая фактически удаляет заметку, как правило, с использованием функции deleteNote. В случае, если пользователь нажимает "Отмена", никаких изменений не происходит, и заметка остается в хранилище данных без изменений.

После того как пользователь подтвердил или отменил удаление, confirmDelete завершает свою работу и закрывает модальное окно или уведомление, возвращая пользователя к предыдущему состоянию интерфейса.

Таким образом, confirmDelete — это важная функция в процессе удаления заметок, которая добавляет в приложение защиту от нежелательных действий и предоставляет пользователю возможность подтвердить или отменить решение перед выполнением необратимой операции.

### 3.3.16 Схема алгоритма useLocalStorageState

React hook (хук) useLocalStorageState используется для хранения и извлечения данных из localStorage, что позволяет сохранять информацию о состоянии приложения (например, о текущем пользователе или списке заметок) между сессиями. Это удобно, потому что данные остаются доступными даже после перезагрузки страницы, что улучшает пользовательский опыт.

Когда компонент использует useLocalStorageState, он принимает два аргумента: key (строка) и initialValue (начальное значение). key — это уникальный идентификатор, по которому данные будут храниться в localStorage, а initialValue — это значение, которое будет использоваться в качестве начального состояния, если в localStorage еще нет данных для данного ключа. Это начальное значение может быть простым значением, например строкой или числом, или более сложной структурой, такой как объект или массив.

При первом рендере компонента хук проверяет, существуют ли данные в localStorage под заданным key. Если данные уже есть, хук извлекает их и использует как текущее состояние компонента. Если данных нет, используется переданное initialValue, которое устанавливает начальное состояние. Это помогает избежать необходимости вручную проверять localStorage на каждом рендере компонента.

useLocalStorageState возвращает массив с двумя элементами: текущее состояние и функцию для его обновления. Текущее состояние можно использовать для отображения данных в компоненте, а функция обновления состояния позволяет изменять эти данные. Когда состояние обновляется через эту функцию, новое значение автоматически сохраняется в localStorage по указанному ключу. Это гарантирует, что данные синхронизированы как с состоянием компонента, так и с localStorage.

Кроме того, хук поддерживает сериализацию и десериализацию данных, что особенно полезно для хранения более сложных структур данных, таких как объекты или массивы. Обычно хук useLocalStorageState использует JSON.parse() для извлечения данных и JSON.stringify() для их сохранения, что позволяет хранить любые данные, совместимые с JSON. Это дает возможность работать с объектами и массивами, не прибегая к дополнительным методам для преобразования данных.

Таким образом, useLocalStorageState значительно упрощает работу с localStorage в приложении на React. Он позволяет сохранять данные между сессиями, синхронизировать их с состоянием компонента и обеспечивать их постоянную доступность даже после перезагрузки страницы. Это делает хук полезным для реализации таких функций, как сохранение пользовательских настроек, аутентификационных данных или состояния интерфейса, которое должно сохраняться при повторном посещении страницы.

## **3.4** **Графический интерфейс**

### 3.4.1 Описание графических компонентов формы Authorization

Форма Authorization представляет из себя приветствующее окно, где пользователю предлагается войти в систему при наличии аккаунта, заполнил поля данными «Почта» и «Пароль». При необходимости у пользователя есть возможность зарегистрироваться в системе и далее получить к ней доступ.

Имеет вид, представленный на рисунке 3.1.

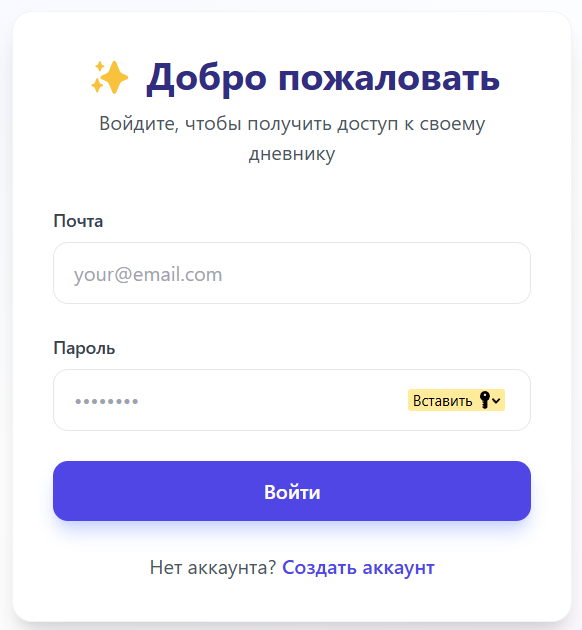


Рисунок 3.1 – Вид формы Autorization

Составляющие главной формы Autorization:

* поле «Почта» для ввода зарегистрированной почты пользователя;
* поле «Пароль» для ввода символов пароля, соответствующего зарегистрированной почте пользователя;
* кнопка «Войти», которая перенаправляет на главную страницу приложения;
* метка «Нет аккаунта? Создать аккаунт», которая перенаправляет на форму регистрации для создания аккаунта.

### 3.4.2 Описание графических компонентов формы Registration

Форма Registration предоставляет пользователю возможность зарегистрировать новый аккаунт для доступа к системе, где пользователю необходимо заполнить поля «Почта» и «Пароль», которые должны соответствовать требованиям системы. После успешной регистрации у пользователя появляется возможность использования приложения.

Имеет вид, представленный на рисунке 3.2.

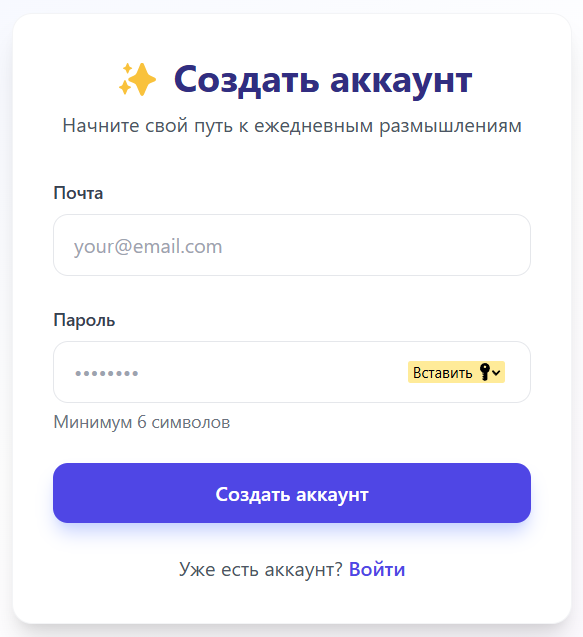


Рисунок 3.2 – Вид формы Registration

Составляющие главной формы Registration:

* поле «Почта» для ввода новой почты пользователя;
* поле «Пароль» для ввода символов пароля, соответствующего новой почте пользователя;
* кнопка «Создать аккаунт», которая перенаправляет на главную страницу приложения;
* метка «Уже есть аккаунт? Войти», которая перенаправляет на форму авторизации пользователя для входа в систему.

### 3.4.3 Описание графических компонентов формы Homepage

Форма Homepage предоставляет доступ к главной странице приложения, где пользователю доступны форма календаря с визуальной анимацией, плавными переходами, подсветкой выходных дней месяца, отображением точек под датой при наличии заметок и стилизация активной даты. Также доступна для просмотра форма со списком заметок, каждая из которых помечена в календарном дне. Отображены соответствующие иконки для удаления и редактирования заметки. Имеется возможность выхода из системы.

Имеет вид, представленный на рисунке 3.3.



Рисунок 3.3 – Вид формы Homepage

Составляющие главной формы Homepage:

* хедер с названием приложения «Мой дневник», названием почты пользователя и кнопка «Выход» для необходимого выхода из системы;
* форма календаря с текущим названием месяца и года;
* стрелки для перехода между месяцами и годами календаря;
* метка «Сегодня» для удобного возврата к текущей дате и ее выделение навигационной формой;
* разметка календаря с названиями дней недели и числами месяца;
* точки отображения существования заполненной заметки под днем месяца;
* форма «Мои заметки» со списком существующих заметок;
* форма заметки с соответствующей датой заполнения и текстовым наполнением заметки;
* кнопка «Карандаш» для формы заметки при переходе к модальному окну редактирования заметки;
* кнопка «Крестик» для формы заметки при переходе к модальному окну удаления заметки.

### 3.4.4 Описание графических компонентов формы AddNote

Форма AddNote предоставляет возможность добавления заметки под выбранной датой в модальном окне. При необходимости можно отменить действие или сохранить введенный текст.

Имеет вид, представленный на рисунке 3.4.

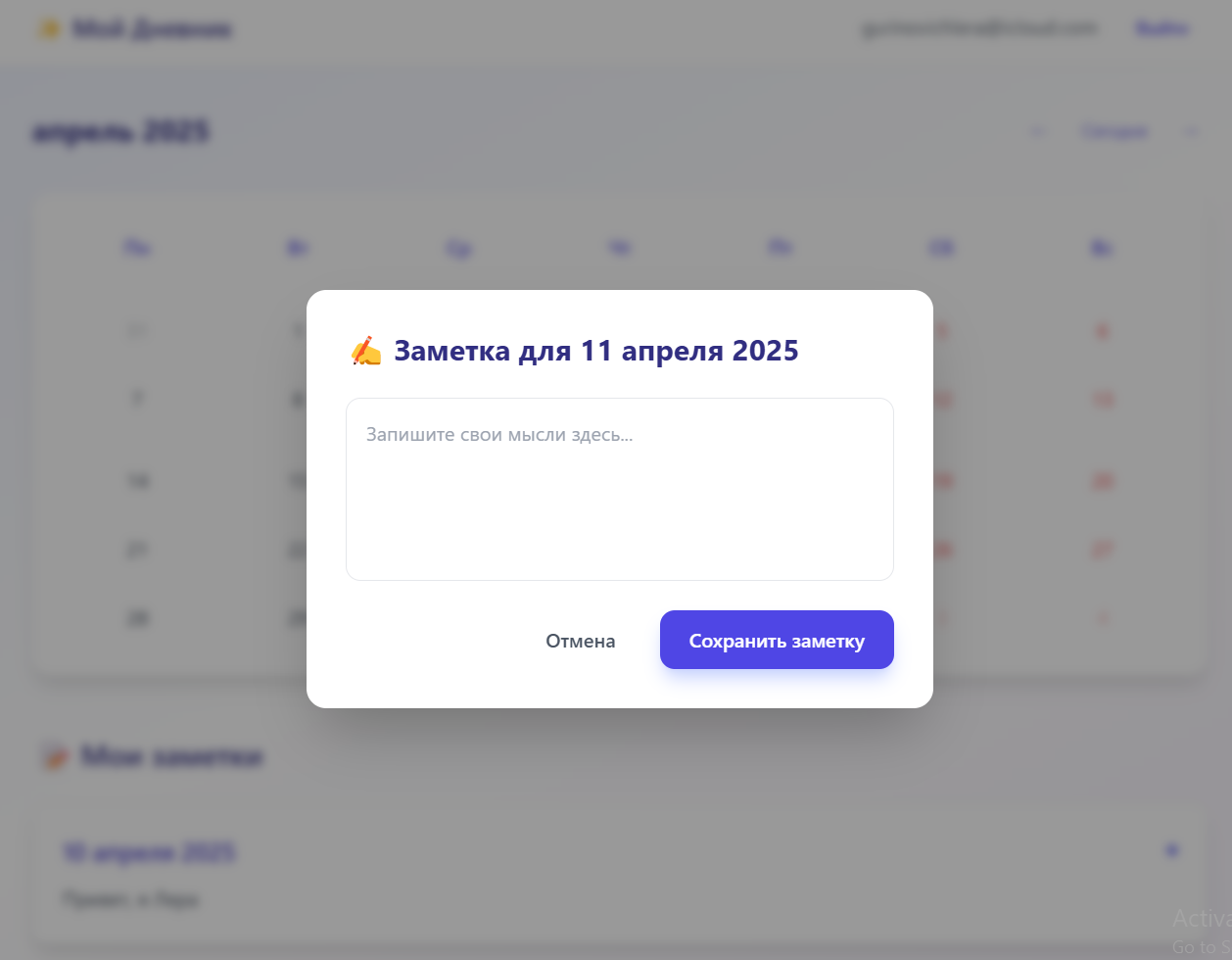


Рисунок 3.4 – Вид формы AddNote

Составляющие главной формы AddNote:

* заголовок «Заметка для dd MMMM yyyy» с отображением выбранной даты для заметки;
* поле «Запишите свои мысли здесь» для ввода текста заметки;
* кнопка «Отмена» для выхода из заполнения модального окна заметки;
* кнопка «Сохранить заметку» для сохранения изменений в модальном окне заполняемой заметки.

### 3.4.5 Описание графических компонентов формы EditNote

Форма EditNote предоставляет возможность редактирования выбранной заметки в модальном окне. При необходимости можно отменить действие или сохранить изменения.

Имеет вид, представленный на рисунке 3.5.

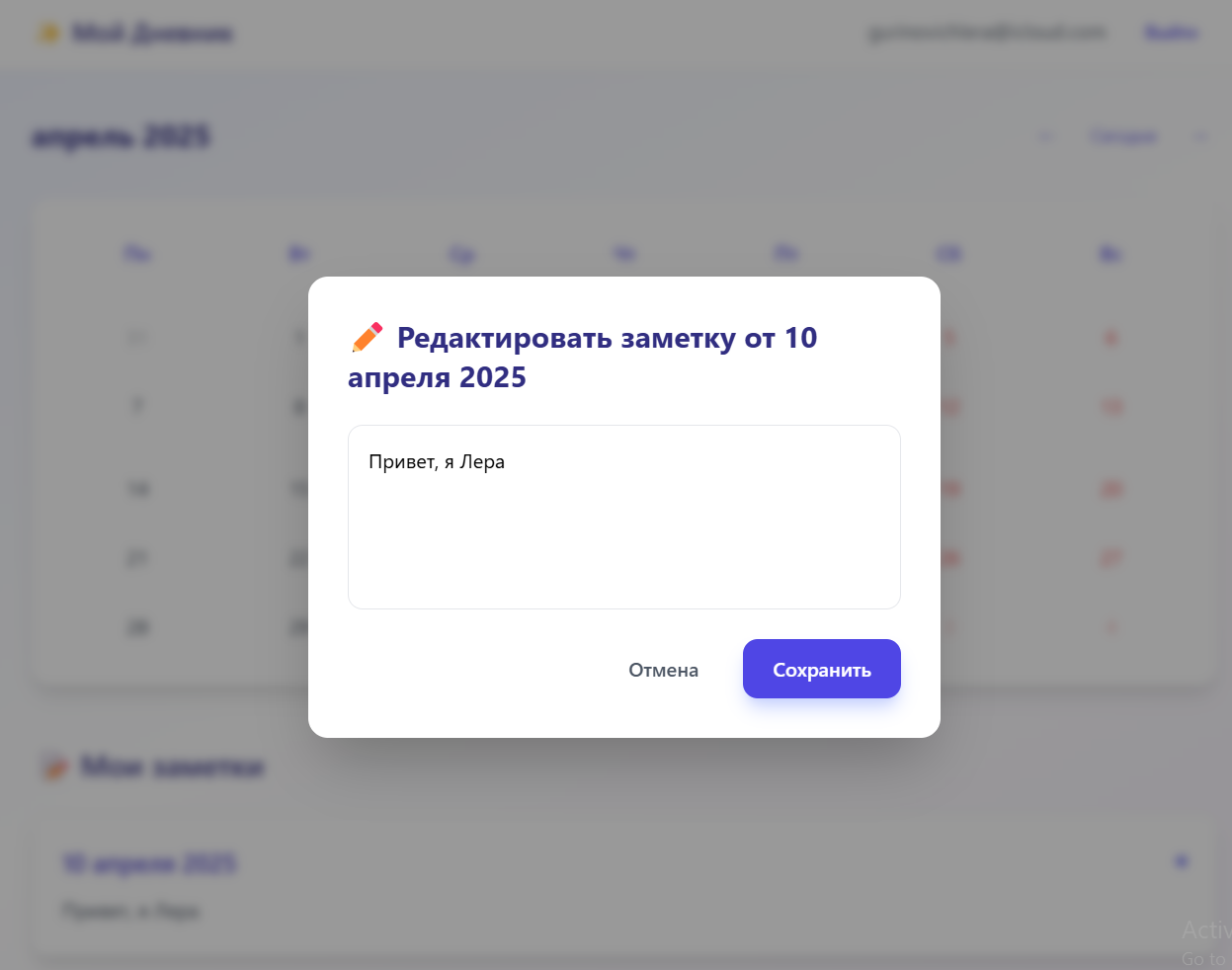


Рисунок 3.5 – Вид формы EditNote

Составляющие главной формы EditNote:

* заголовок «Редактировать заметку от dd MMMM yyyy» с отображением даты для выбранной заметки;
* поле с текстом существующей заметки для ввода нового текста;
* кнопка «Отмена» для выхода из заполнения модального окна заметки;
* кнопка «Сохранить заметку» для сохранения изменений в модальном окне заполняемой заметки.

### 3.4.6 Описание графических компонентов формы DeleteNote

Форма DeleteNote предоставляет возможность удаления выбранном заметки в модальном окне. При необходимости можно отменить действие или подтвердить действие удаления.

Имеет вид, представленный на рисунке 3.6.

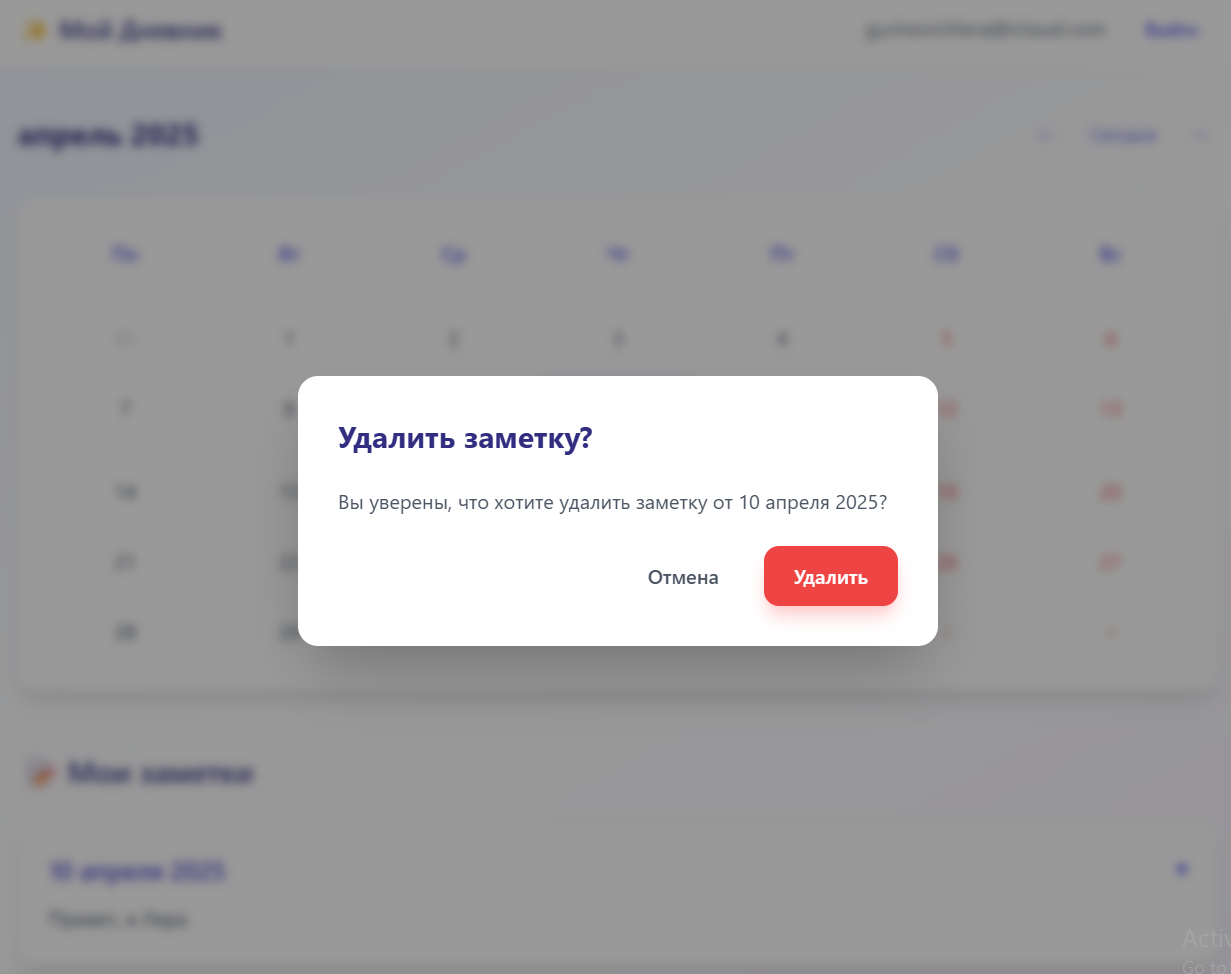


Рисунок 3.6 – Вид формы DeleteNote

Составляющие главной формы DeleteNote:

* заголовок «Удалить заметку» для вопросительного подтверждения действия;
* описание с текстом «Вы уверены, что хотите удалить заметку от dd MMMM yyyy» с отображением даты для выбранной заметки;
* кнопка «Отмена» для выхода из модального окна удаления заметки;
* кнопка «Удалить» для подтверждения удаления выбранной заметки в модальном окне.

# **4** **ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА**

## **4.1 Авторизация пользователя**

### 4.1.1 Тест 1

Таблица 20 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при вводе правильной почты и пароля |
| Исходный набор данных: | Существующие почта и пароль: gurinovichlera@icloud.com, 1234567 |
| Ожидаемый результат: | Переход на главную страницу приложения с формами календаря и заметок, названием авторизированной почты и кнопкой «Выйти» |
| Полученный результат: |  |

Продолжение таблицы 20

|  |  |
| --- | --- |
| Полученный результат: |  |

### 4.1.2 Тест 2

Таблица 21 – Тест 2

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при вводе неправильной почты и пароля |
| Исходный набор данных: | Несуществующая почта и пароль: gurinoviccc@icloud.com, 1234 |

Продолжение таблицы 21

|  |  |
| --- | --- |
| Ожидаемый результат: | Ошибка «Неправильный email или пароль» |
| Полученный результат: |  |

### 4.1.3 Тест 3

Таблица 22 – Тест 3

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при клике на «Создать аккаунт» |
| Исходный набор данных: | Форма авторизации и ссылочный текст «Создать аккаунт» |

Продолжение таблицы 22

|  |  |
| --- | --- |
| Исходный набор данных: | Форма авторизации и ссылочный текст «Создать аккаунт» |
| Ожидаемый результат: | Перенаправление на страницу с формой регистрации |
| Полученный результат: |  |

## **4.2 Регистрация пользователя**

### 4.2.1 Тест 1

Таблица 23 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при вводе корректной почты и пароля |
| Исходный набор данных: | Существующие почта и пароль: gurinovichlera@gmail.com, 1234567 |
| Ожидаемый результат: | Переход на главную страницу приложения с формами календаря и заметок, названием авторизированной почты и кнопкой «Выйти» |
| Полученный результат: |  |

Продолжение таблицы 23

|  |  |
| --- | --- |
| Полученный результат: |  |

### 4.2.2 Тест 2

Таблица 24 – Тест 2

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при вводе уже зарегистрированной почты для регистрации |
| Исходный набор данных: | Зарегистрированная почта и пароль: gurinovichlera@gmail.com, 123456 |

Продолжение таблицы 24

|  |  |
| --- | --- |
| Ожидаемый результат: | Ошибка «Этот email уже зарегистрирован» |
| Полученный результат: |  |

### 4.2.3 Тест 3

Таблица 25 – Тест 3

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при вводе незарегистрированной почты и неправильного пароля |
| Исходный набор данных: | Незарегистрированная почта и пароль: gurinovichlera1@gmail.com, 123 |

Продолжение таблицы 25

|  |  |
| --- | --- |
| Ожидаемый результат: | Ошибка «Пароль должен содержать не менее 6 символов» |
| Исходный набор данных: |  |

### 4.2.4 Тест 4

Таблица 25 – Тест 4

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при клике на «Войти» |
| Исходный набор данных: | Форма регистрации и ссылочный текст «Войти |

Продолжение таблицы 25

|  |  |
| --- | --- |
| Исходный набор данных: | Форма регистрации и ссылочный текст «Войти |
| Ожидаемый результат: | Перенаправление на страницу с формой авторизации |
| Полученный результат: |  |

## **4.3 Навигация календаря**

### 4.3.1 Тест 1

Таблица 26 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка организации календаря в соответствии с григорианским календарем |
| Исходный набор данных: | Форма календаря для апреля 2025 года |
| Ожидаемый результат: | Корректное отображение дней недели и чисел для апреля 2025 года |
| Полученный результат: |  |

### 4.3.2 Тест 2

Таблица 27 – Тест 2

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка навигации календаря по месяцам и годам с помощью стрелок «Назад» и «Вперед» |

Продолжение таблицы 27

|  |  |
| --- | --- |
| Исходный набор данных: | Форма календаря для апреля 2025 года |
| Ожидаемый результат: | Корректное отображение дней недели и чисел для мая 2025 года и января 2026 года |
| Полученный результат: |  |

### 4.3.3 Тест 3

Таблица 28 – Тест 3

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка перехода к сегодняшней дате по кнопке «Сегодня» |
| Исходный набор данных: | Форма календаря для июля 2025 года |
| Ожидаемый результат: | Переход к форме календаря на сегодняшнее число, год и месяц (месяц: апрель, год: 2025, число: 11) |
| Полученный результат: |  |

### 4.3.4 Тест 4

Таблица 29 – Тест 4

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка подсветки выбранного числа месяца при наведении на него |
| Исходный набор данных: | Форма календаря для апреля 2025 года |

Продолжение таблицы 29

|  |  |
| --- | --- |
| Исходный набор данных: | Форма календаря для апреля 2025 года, наведение на число 25 апреля |
| Ожидаемый результат: | Подсветка числа месяца при наведении на него (25 апреля) |
| Полученный результат: |  |

### 4.3.5 Тест 5

Таблица 30 – Тест 5

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка отображения точек под датой при наличии заметок |
| Исходный набор данных: | Форма календаря для апреля 2025 года, заполненная заметка для 18 апреля 2025 года |

Продолжение таблицы 31

|  |  |
| --- | --- |
| Исходный набор данных: | Форма календаря для апреля 2025 года, заполненная заметка для 18 апреля 2025 года |
| Ожидаемый результат: | Отображение точки под датой 18 апреля, для которой заполнена заметка |
| Полученный результат: |  |

### 4.3.6 Тест 6

Таблица 31 – Тест 6

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка отсутствия отображения точек под датой после удаления заметки |
| Исходный набор данных: | Форма календаря для апреля 2025 года, удаленная заметка для 18 апреля 2025 года |

Продолжение таблицы 31

|  |  |
| --- | --- |
| Ожидаемый результат: | Отсутствие отображение точки под датой 18 апреля 2025 года |
| Полученный результат: |  |

## **4.4 Операции над заметками**

### 4.4.1 Тест 1

Таблица 32 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при создании заметки |
| Исходный набор данных: | Форма календаря для апреля 2025 года, клик по дате 11 апреля 2025 года |
| Ожидаемый результат: | Появление модального окна с возможностью написания текста заметки и кнопками «Отменить» и «Сохранить заметку» |

Продолжение таблицы 32

|  |  |
| --- | --- |
| Полученный результат: |  |

### 4.4.2 Тест 2

Таблица 33 – Тест 2

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при сохранении заметки |
| Исходный набор данных: | Модальное окно с созданием заметки для 11 апреля 2025 года, клик по кнопке «Сохранить заметку», текст заметки «Сегодня был хороший день» |
| Ожидаемый результат: | Появление формы с подтверждением сохранения заметки |

Продолжение таблицы 32

|  |  |
| --- | --- |
| Полученный результат: |  |

### 4.4.3 Тест 3

Таблица 34 – Тест 3

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при подтверждении сохранения заметки |
| Исходный набор данных: | Форма с подтверждением сохранения заметки для 11 апреля 2025 года, клик по кнопке «Подтвердить» |
| Ожидаемый результат: | Появление формы сохраненной заметки в списке «Мои заметки» на основной странице приложения |

Продолжение таблицы 33

|  |  |
| --- | --- |
| Полученный результат: |  |

### 4.4.4 Тест 4

Таблица 35 – Тест 4

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при отмене сохранения заметки |
| Исходный набор данных: | Форма с подтверждением сохранения заметки для 11 апреля 2025 года, клик по кнопке «Отменить» |
| Ожидаемый результат: | Возвращение на основную страницу приложения с формой календаря и пустым списком «Мои заметки» |
| Полученный результат: |  |

Продолжение таблицы 35

|  |  |
| --- | --- |
| Полученный результат: |  |

### 4.4.5 Тест 5

Таблица 36 – Тест 5

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при сохранении редактирования заметки |
| Исходный набор данных: | Существующая заметка для 11 апреля 2025 года, клик по кнопке «Карандаш», измененный текст «Сегодня был очень хороший день», клик по кнопке «Сохранить» |
| Ожидаемый результат: | Появление формы с подтверждением изменения заметки |
| Полученный результат: |  |

Продолжение таблицы 36

|  |  |
| --- | --- |
| Полученный результат: |  |

### 4.4.6 Тест 6

Таблица 37 – Тест 6

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при подтверждении сохранения редактирования заметки |
| Исходный набор данных: | Форма с подтверждением сохранения изменения заметки, клик по кнопке «Подтвердить» |
| Ожидаемый результат: | Появление формы заметки с измененным текстом «Сегодня был очень хороший день» в списке «Мои заметки» |
| Полученный результат: |  |

### 4.4.7 Тест 7

Таблица 38 – Тест 7

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при отмене сохранения редактирования заметки |
| Исходный набор данных: | Форма с подтверждением сохранения изменения заметки, клик по кнопке «Отмена» |
| Ожидаемый результат: | Появление формы заметки с первоначальным текстом «Сегодня был хороший день» в списке «Мои заметки» |
| Полученный результат: |  |

### 4.4.8 Тест 8

Таблица 39 – Тест 8

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при удалении заметки |
| Исходный набор данных: | Существующая заметка для 11 апреля 2025 года, клик по кнопке «Удалить» |
| Ожидаемый результат: | Появление формы с подтверждением удаления заметки |

Продолжение таблицы 39

|  |  |
| --- | --- |
| Полученный результат: |  |

### 4.4.9 Тест 9

Таблица 40 – Тест 9

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при отмене удаления заметки |
| Исходный набор данных: | Форма с подтверждением удаления заметки, клик по кнопке «Отмена» |
| Ожидаемый результат: | Отображение заметки для 11 апреля 2025 года в списке «Мои заметки» |
| Полученный результат: |  |

### 4.4.10 Тест 10

Таблица 41 – Тест 10

|  |  |
| --- | --- |
| Тестовая ситуация: | Проверка правильности работы программы при подтверждении удаления заметки |
| Исходный набор данных: | Форма с подтверждением удаления заметки, клик по кнопке «Удалить» |
| Ожидаемый результат: | Отображение пустого списка «Мои заметки» |
| Полученный результат: |  |

# **5** **РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА**

## **5.1 Запуск**

Это приложение размещено на веб-сервере, что позволяет пользователю легко получить к нему доступ: достаточно ввести в адресной строке браузера ссылку на домен веб-сайта — <https://lllerony.github.io/My-Diary/>. После нажатия «Enter» на клавиатуре в строке браузера, пользователь будет перенаправлен на страницу с авторизацией, которая представлена на рисунке 5.1.

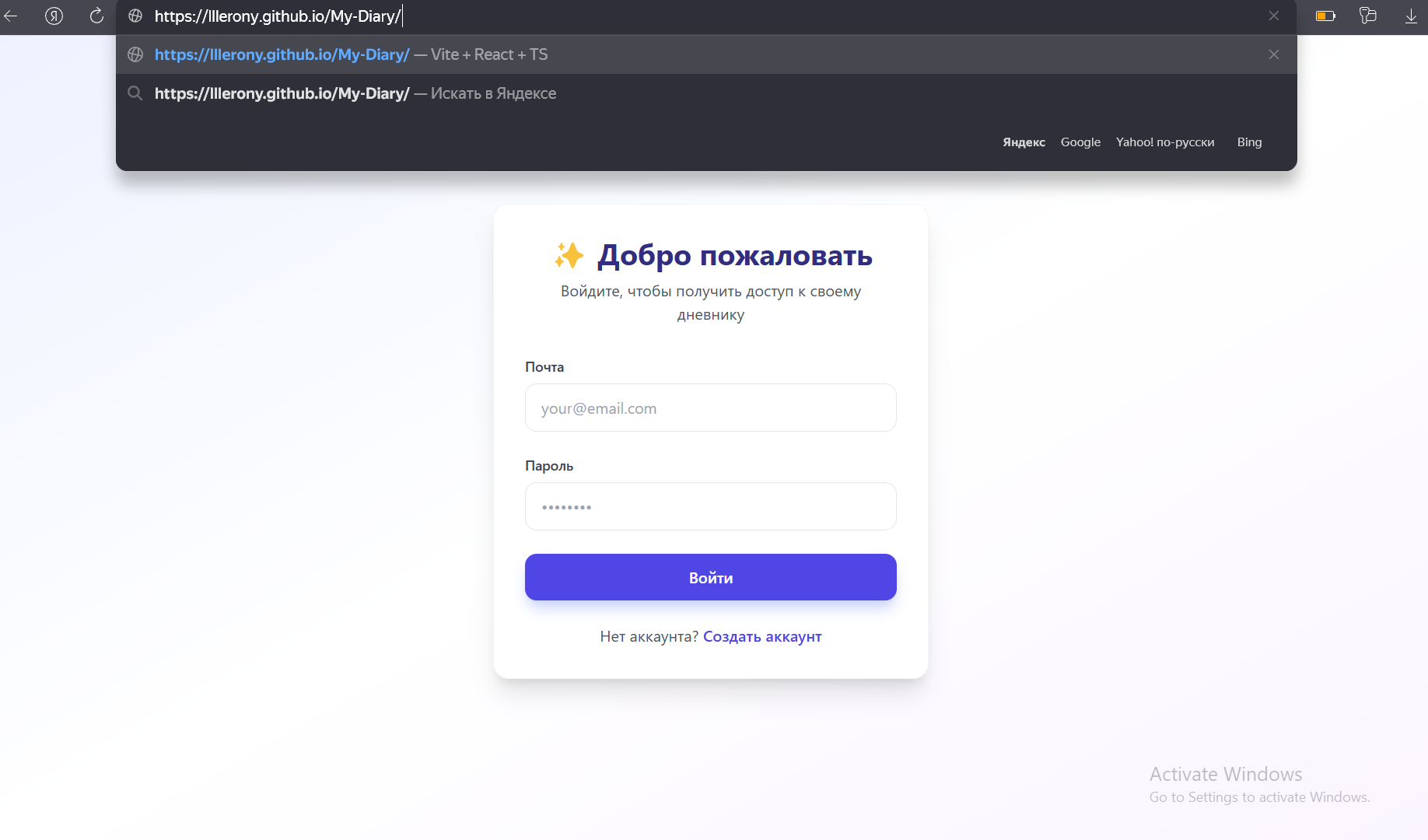


Рисунок 5.1 – Запуск веб-сайта

## **5.2 Авторизация**

При переходе на страницу авторизации пользователю предлагается ввести свои учетные данные – адрес электронной почты и пароль. Если введённая информация окажется некорректной, например, будет указана неверная почта или пароль, то после нажатия на кнопку «Войти» система уведомит пользователя об ошибке и предложит повторить попытку. Окно авторизации с демонстрацией неверных указанных данных представлено на рисунке 5.2.

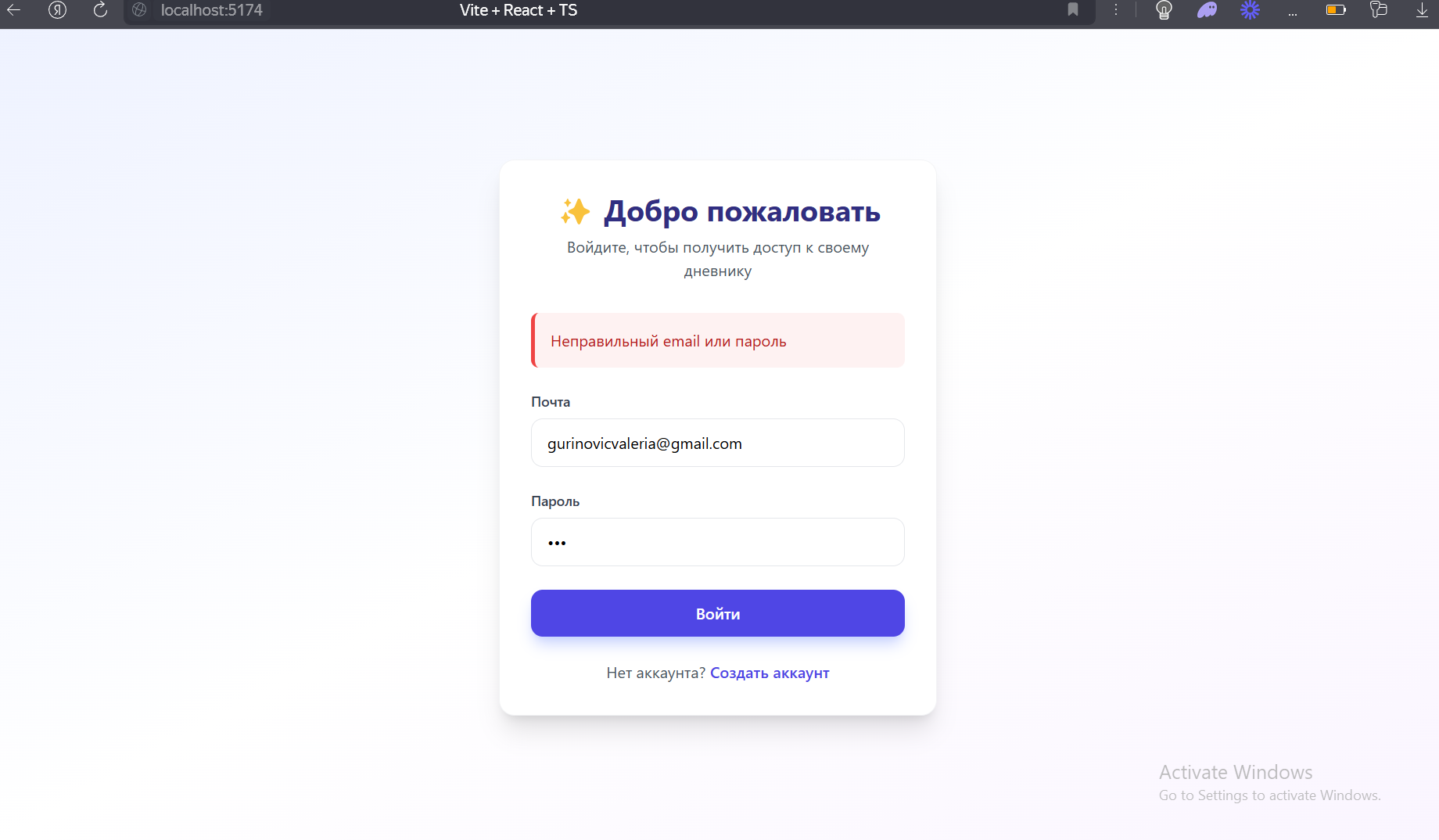


Рисунок 5.2 – Окно авторизации с некорректными данными

Если введённая информация для авторизации окажется корректной, то после нажатия на кнопку «Войти» логика системы перенаправит пользователя на главную страницу приложения. Окно авторизации с демонстрацией верных указанных данных представлено на рисунке 5.3.

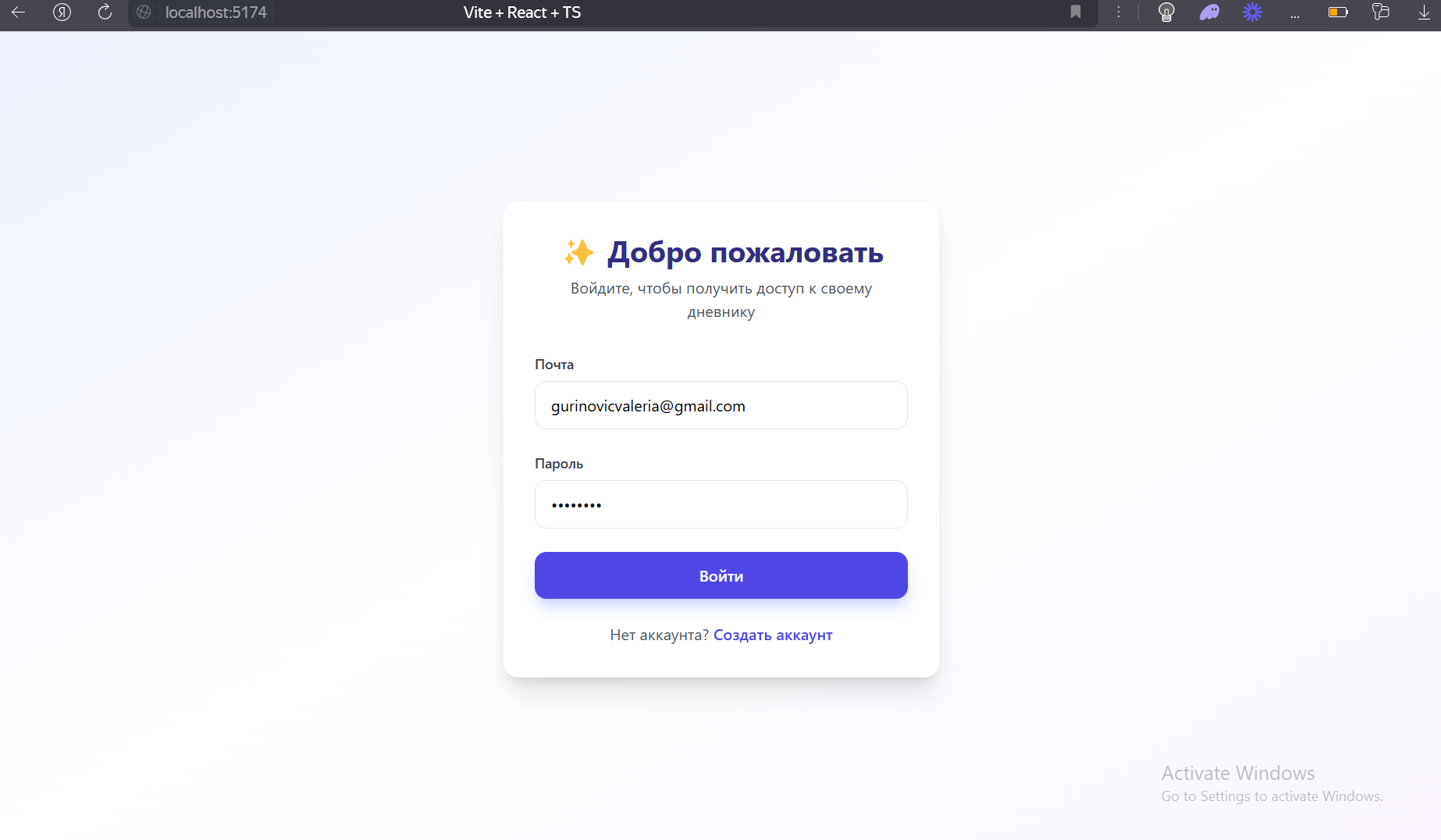


Рисунок 5.3 – Окно авторизации с корректными данными

## **5.3 Регистрация**

При отсутствии существующего аккаунта, пользователь может создать аккаунт, перейдя в окне авторизации по ссылочному тексту «Создать аккаунт». После пользователь будет перенаправлен на страницу регистрации, где пользователю предлагается ввести свои учетные данные – адрес электронной почты и пароль, страница которой представлена на рисунке 5.4.

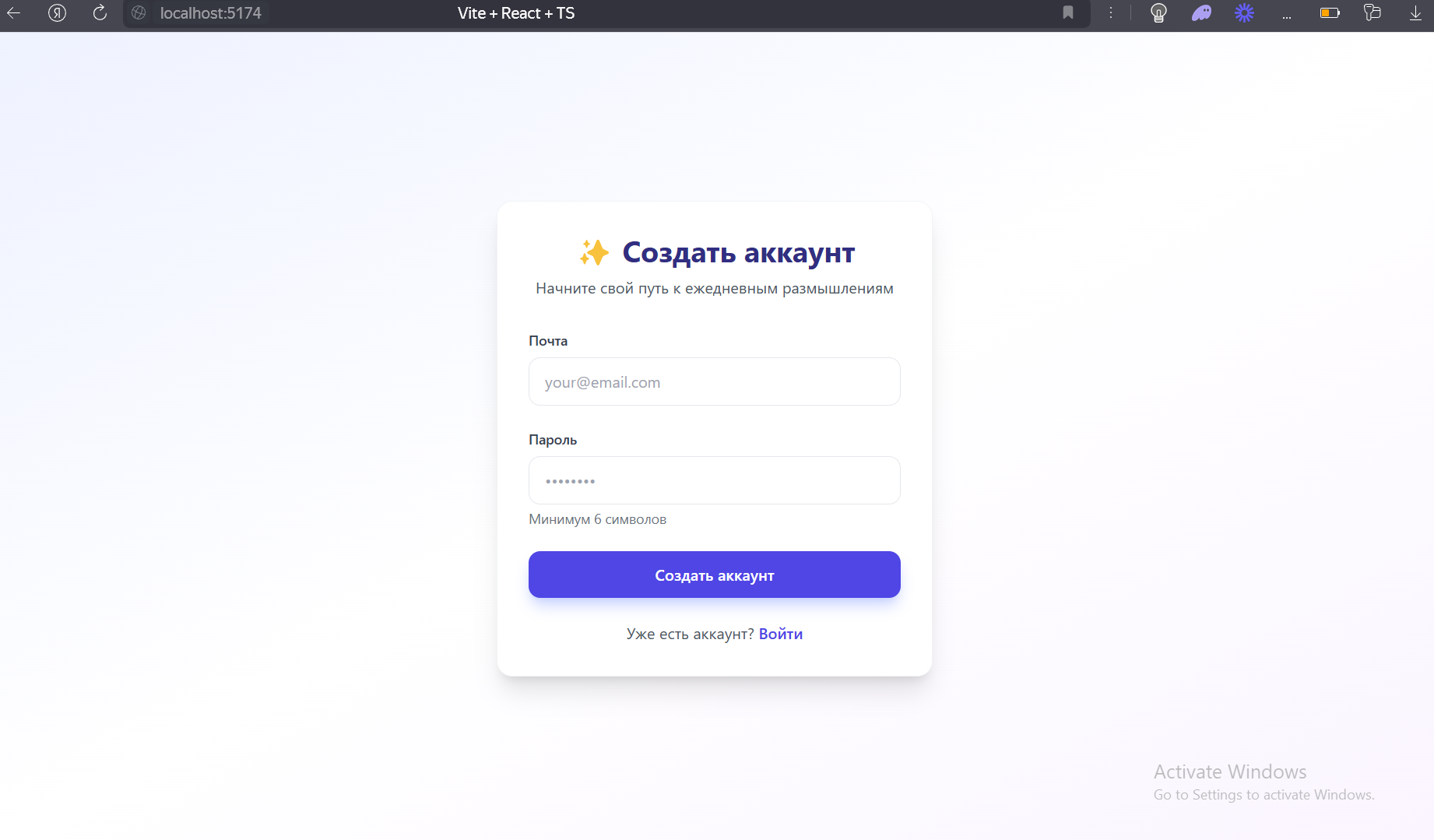


Рисунок 5.4 – Окно регистрации

Если пользователь введет не валидный пароль, которые должен быть не менее 6 символов, то после нажатия на кнопку «Создать аккаунт» система уведомит пользователя об ошибке и предложит повторить попытку. Окно регистрации с демонстрацией неверно указанного пароля представлено на рисунке 5.5.

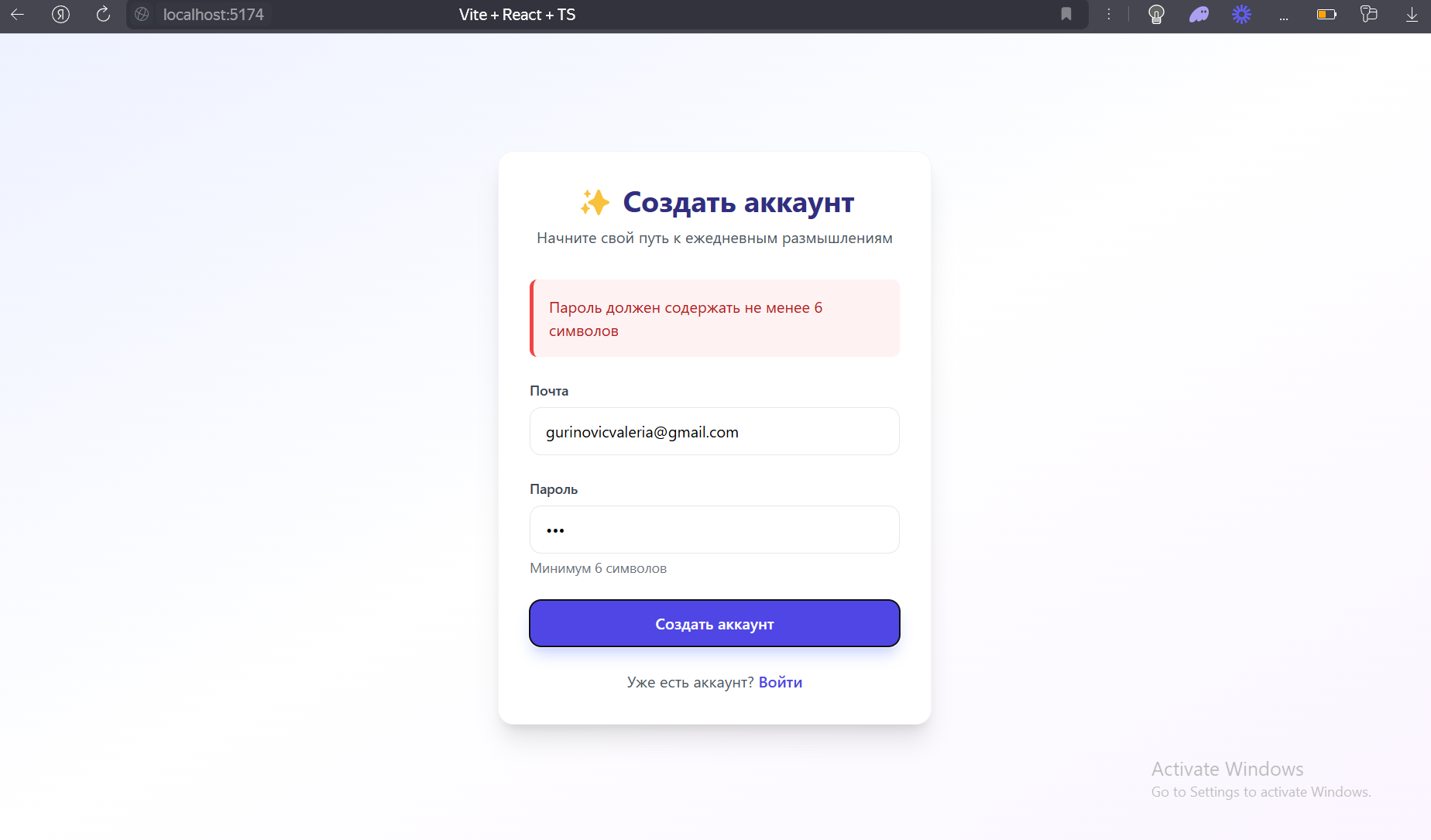


Рисунок 5.5 – Окно регистрации с некорректным паролем

Если пользователь введет почту, которая ранее была зарегистрирована, то после нажатия на кнопку «Создать аккаунт» система уведомит пользователя об ошибке и предложит повторить попытку. Окно регистрации с демонстрацией неверно указанной почтой представлено на рисунке 5.6.

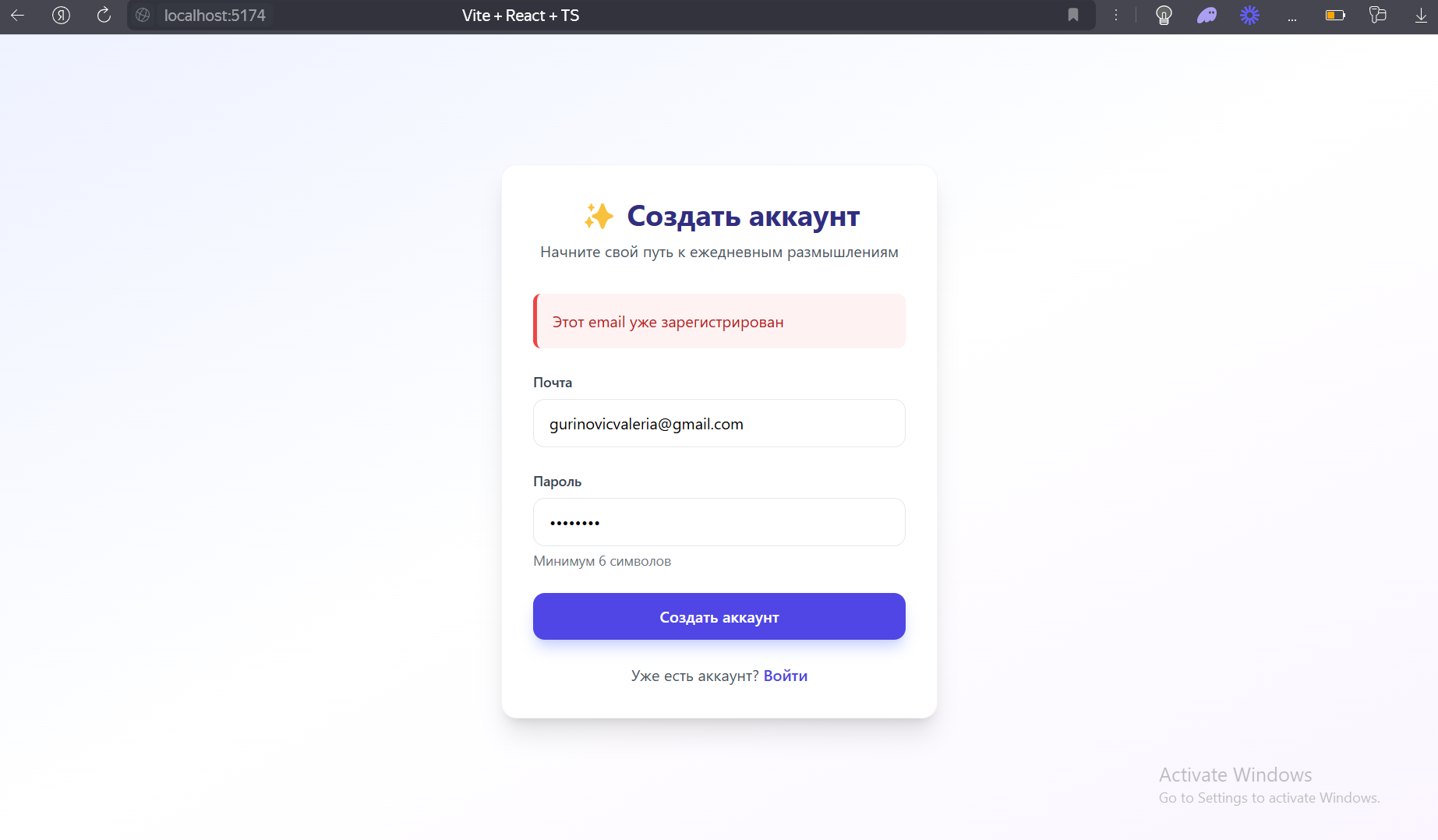


Рисунок 5.6 – Окно регистрации с некорректной почтой

Если введённая информация для регистрации окажется корректной, то после нажатия на кнопку «Создать аккаунт» логика системы перенаправит пользователя на главную страницу приложения. Окно регистрации с демонстрацией верных указанных данных представлено на рисунке 5.7.

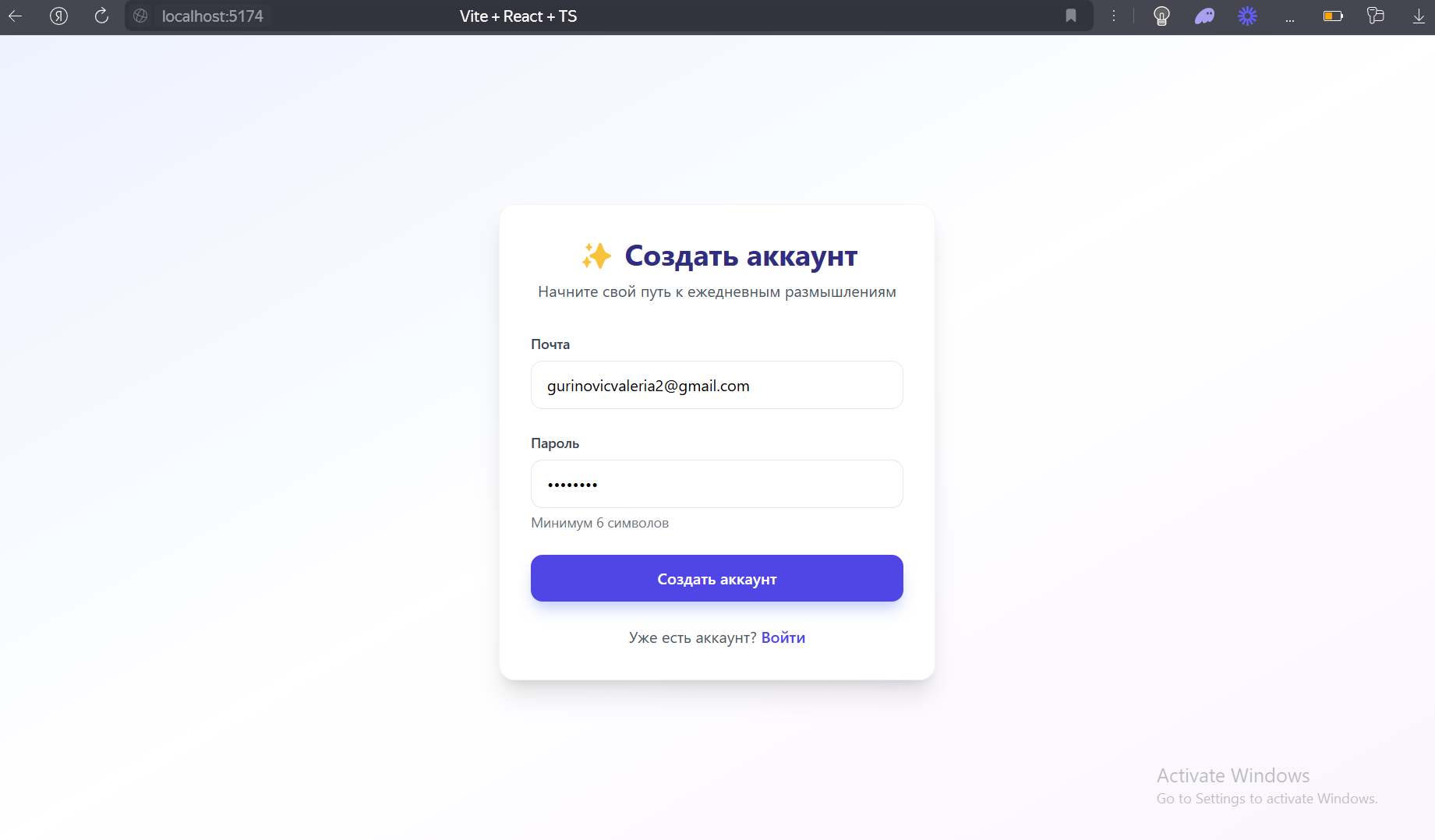


Рисунок 5.7 – Окно регистрации с корректными данными

## **5.4 Главная страница**

После успешной авторизации пользователь получает полный доступ ко всем возможностям приложения. Вверху страницы ему доступны функции выхода и просмотра информации о приложении. Пользователь может просматривать название активной почты, под которой он вошел в систему. Также на главной странице перед ним открывается интерактивный календарь с наглядной подсветкой текущего дня, отображением всех дней недели и дат текущего месяца. Ниже располагается секция со списком созданных заметок, связанных с выбранными датами. Главная страница приложения представлена на рисунке 5.8.

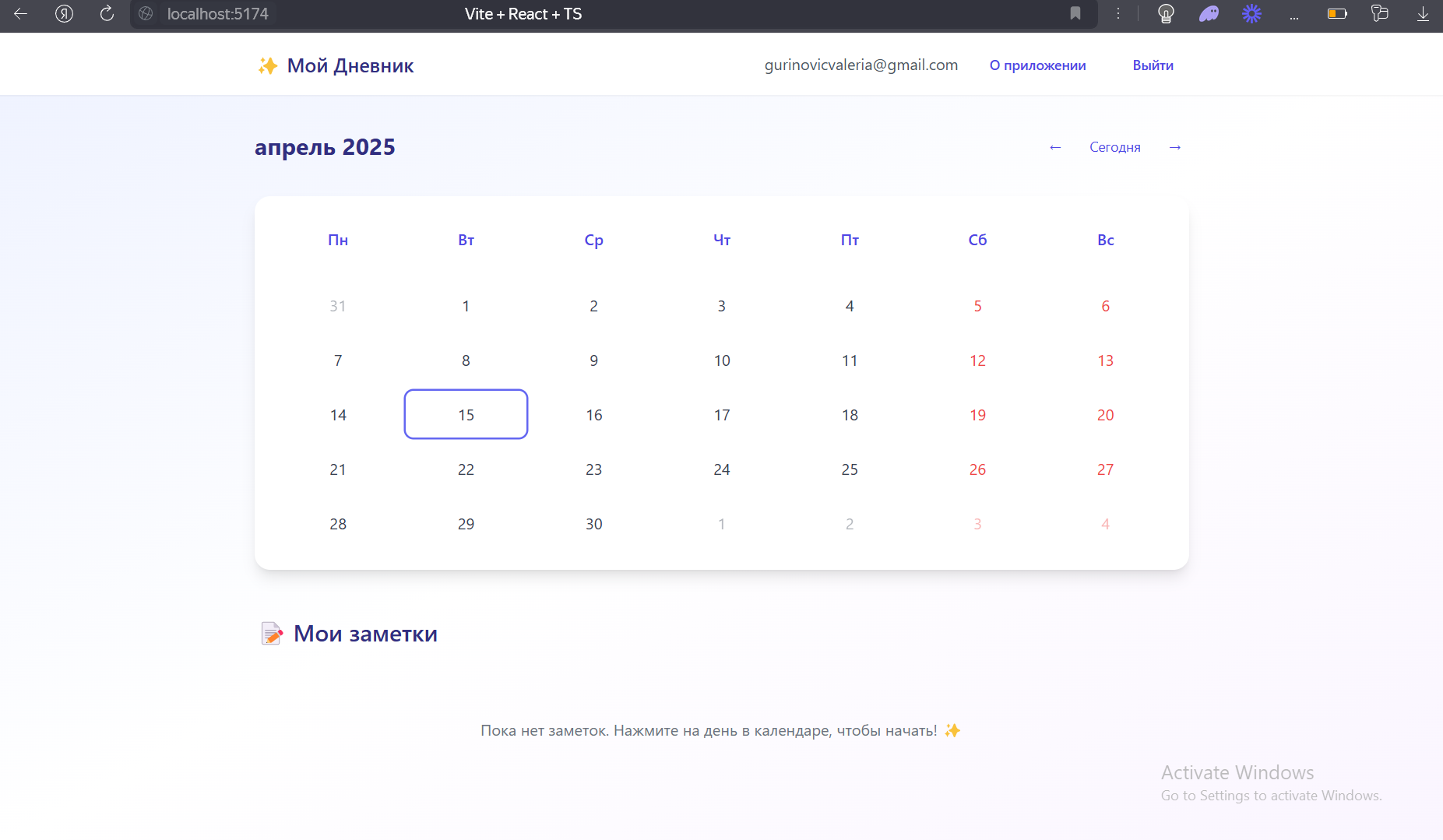


Рисунок 5.8 – Главная страница приложения

### 5.4.1 Календарь

Пользователю предоставлена удобная навигационная панель для управления календарём: с её помощью можно легко перемещаться между месяцами и годами, используя стрелки «Вперёд» и «Назад» для просмотра нужного временного периода. Пример использования навигационной панели календаря представлена на рисунке 5.9.

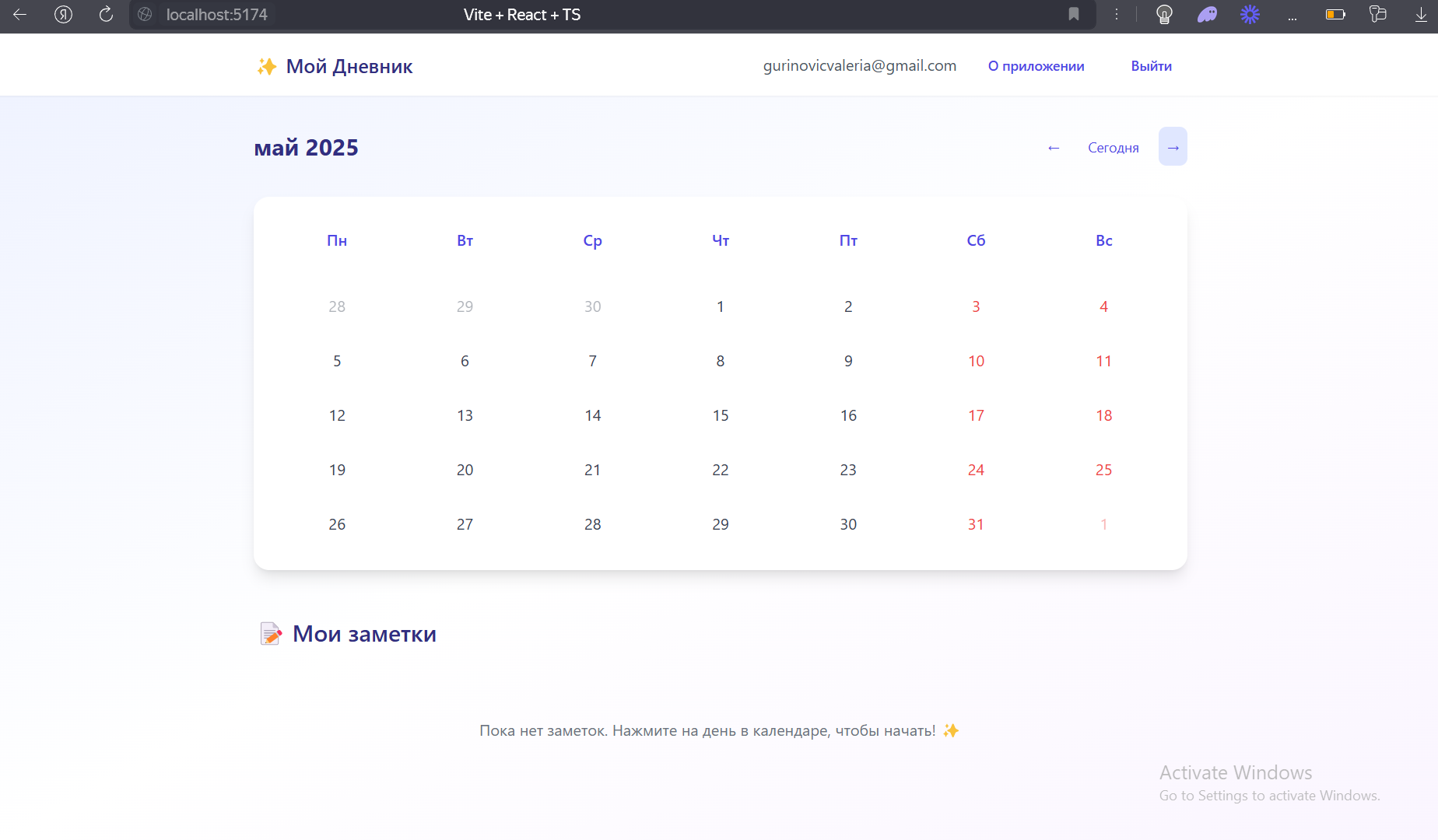


Рисунок 5.9 – Навигационная панель календаря

Также у пользователя имеется возможность перейти к текущей дате вне зависимости от переключения временного периода с помощью кнопки «Сегодня». Пример использования данной кнопки представлена на рисунке 5.10.

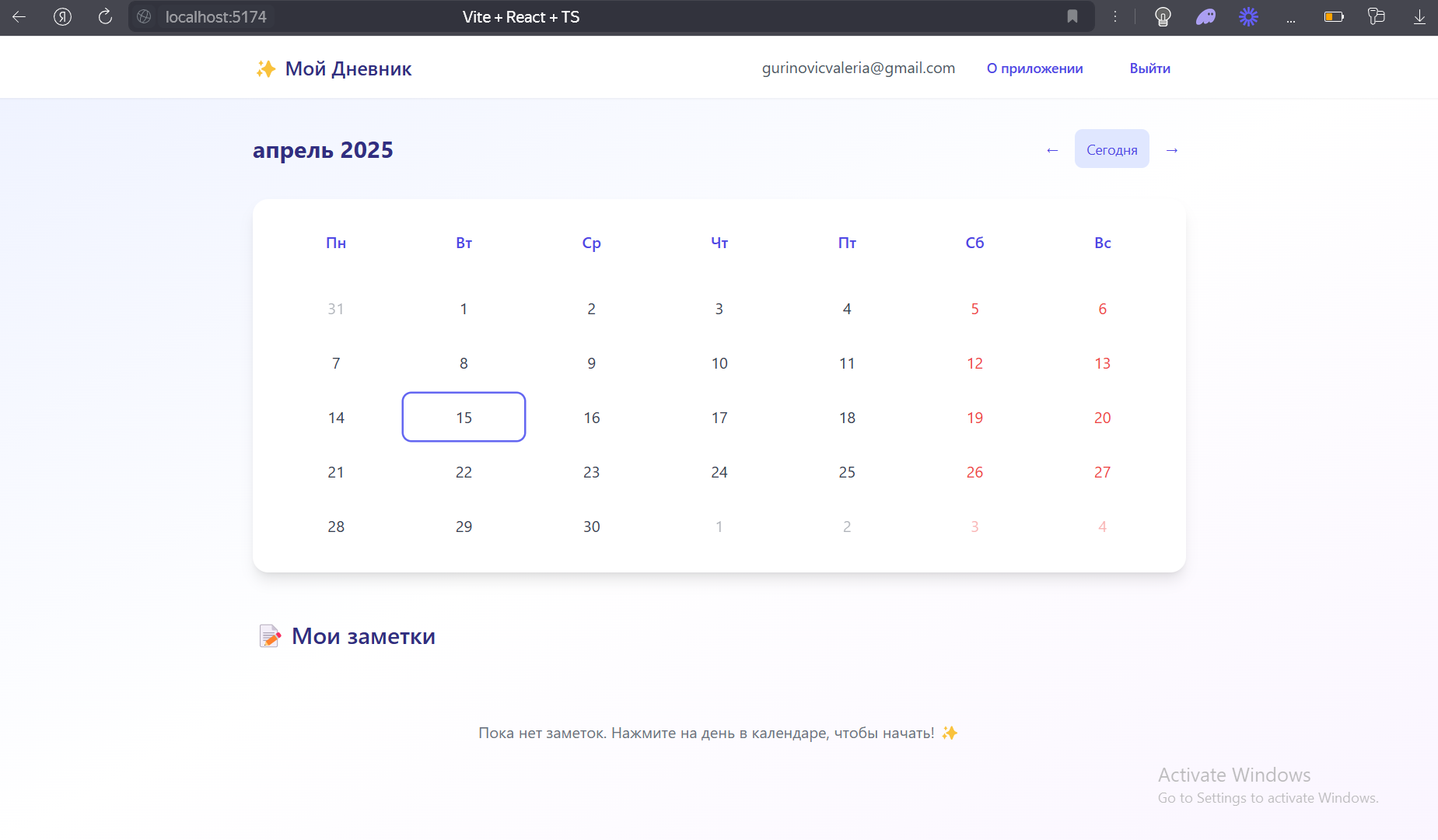


Рисунок 5.10 – Пример использования кнопки «Сегодня»

При нажатии на дату календаря, пользователю будет предоставлено модальное окно для заполнения заметки для выбранной даты, которое представлено на рисунке 5.11.

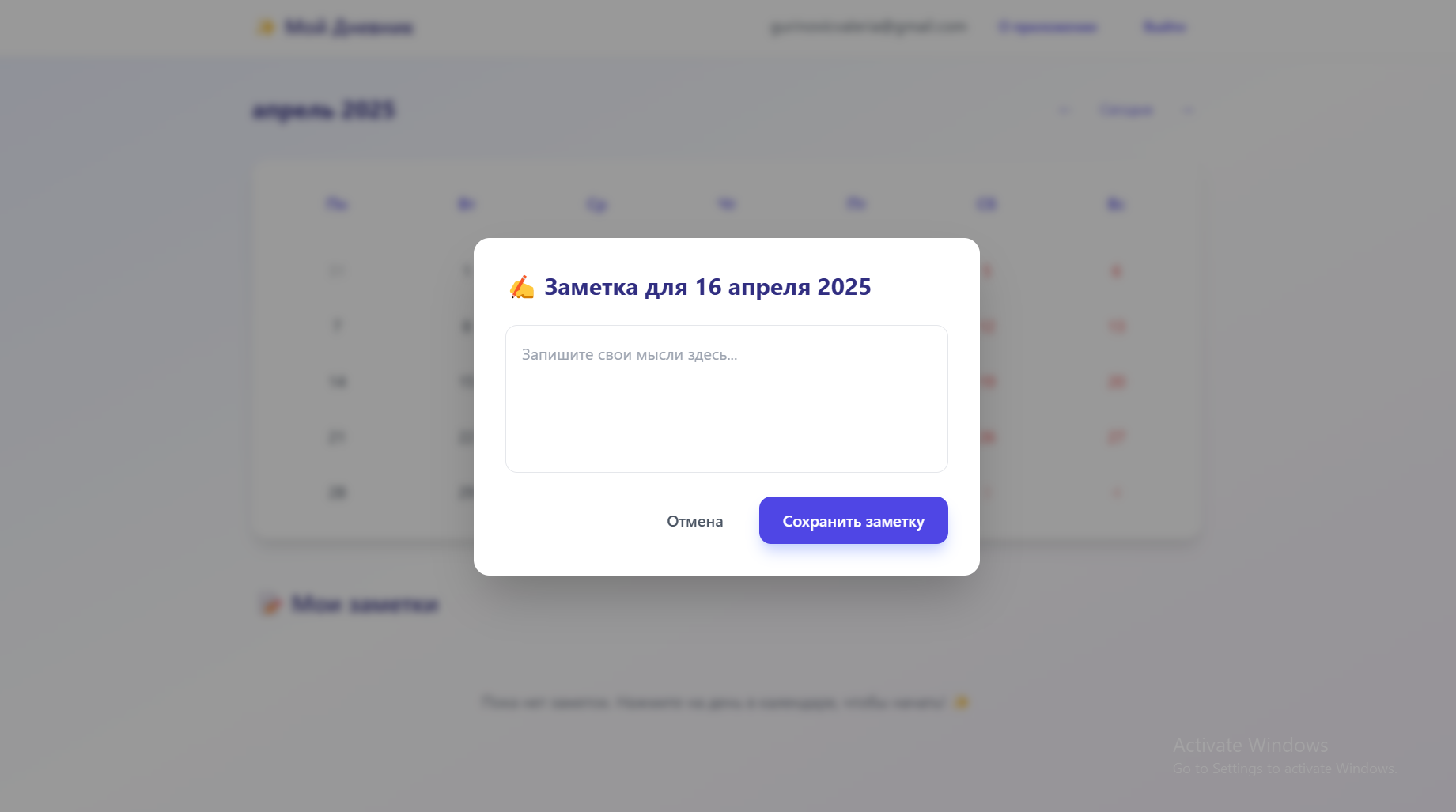


Рисунок 5.11 – Пример модального окна заметки для выбранной даты

После того как пользователь добавит заметку для выбранной даты, соответствующее число в календаре будет отмечено аккуратной точкой под ним — это визуально подскажет, что на этот день есть сохранённая заметка. Пример отображения заполненной даты в календаре представлено на рисунке 5.12.

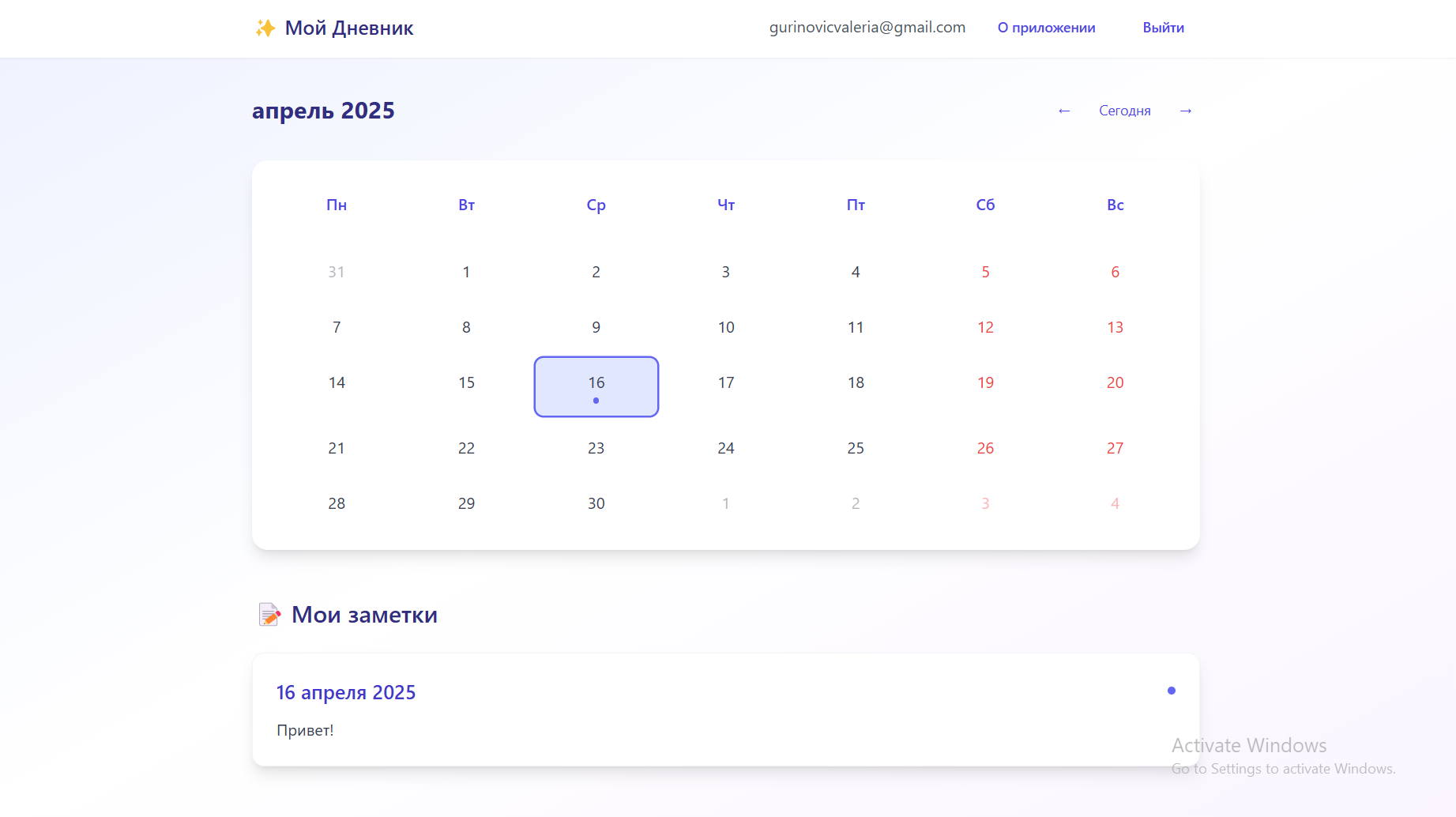


Рисунок 5.12 – Пример отображения даты с заметкой

### 5.4.2 Заметки

Чтобы добавить заметку к выбранной дате, пользователю необходимо кликнуть по нужному дню в календаре. После этого откроется модальное окно для ввода текста заметки, как показано на рисунке 5.11. В данном окне пользователь может ввести содержание заметки и нажать кнопку «Сохранить заметку». После этого отобразится окно подтверждения, содержащее кнопки «Отменить» и «Подтвердить», позволяющее подтвердить или отменить добавление заметки, пример которой представлен на рисунке 5.13. При нажатии на кнопку «Отменить» заметка не будет добавлена в список «Мои заметки», при «Подтвердить» – будет добавлена.

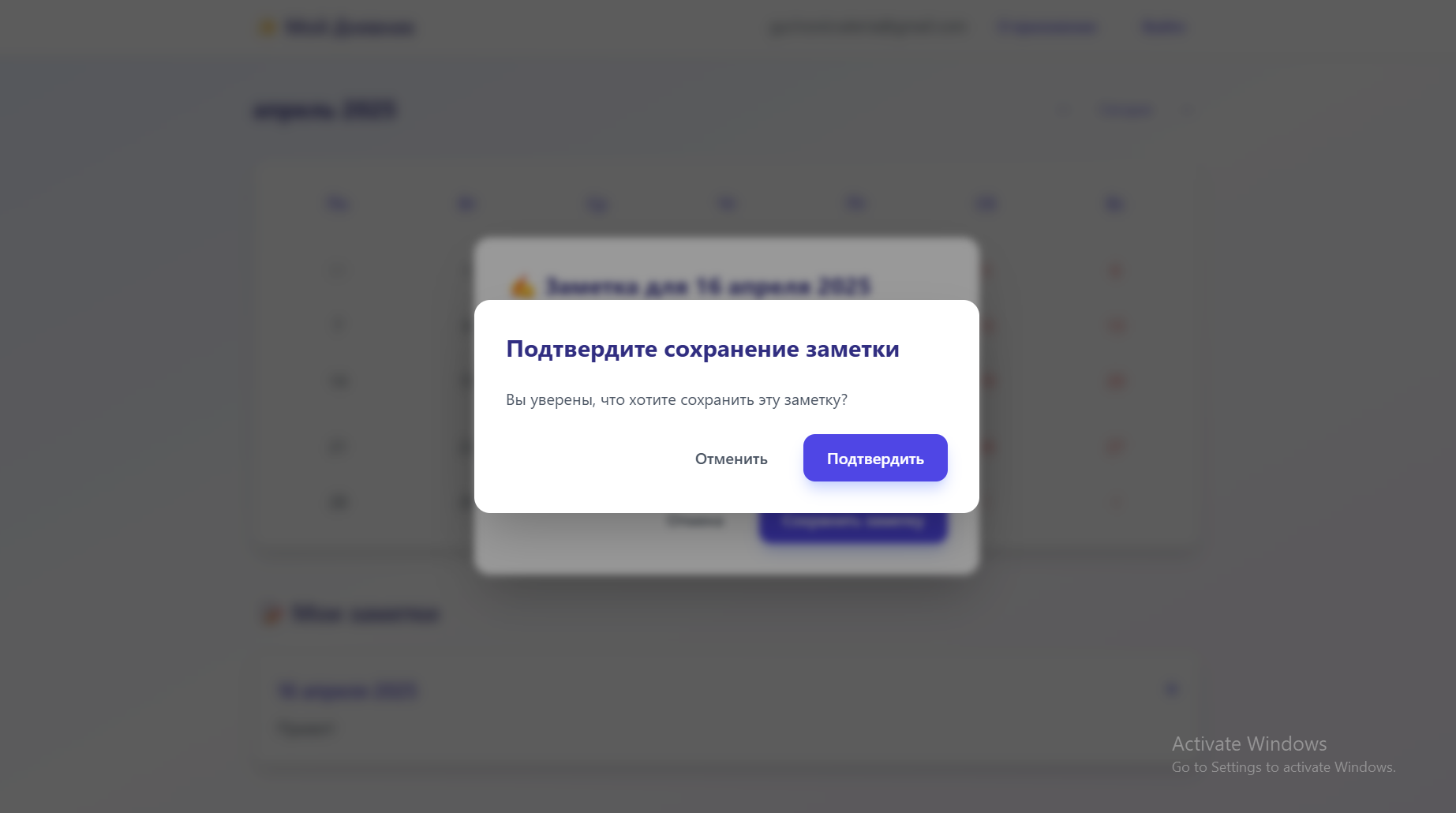


Рисунок 5.13 – Подтверждающее окно для сохранения заметки

При наведении курсора на заметку пользователю отображается функциональная панель, содержащая интуитивно понятные элементы управления: кнопку «Редактировать» с иконкой в виде карандаша и кнопку «Удалить» с иконкой в виде крестика. Данная панель представлена на рисунке 5.14.

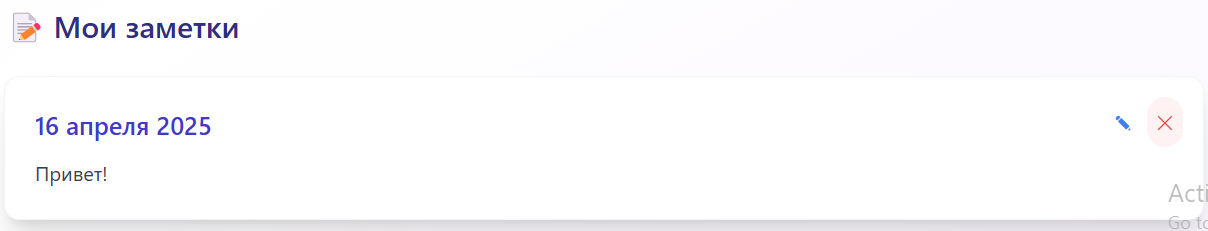


Рисунок 5.14 – Функциональная панель заметки

При нажатии на кнопку «Редактировать» для пользователя будет отображено модальное окно с возможностью редактирования текста заметки. После необходимых изменений текста заметки пользователь может нажать на кнопку «Отмена» для отмены действий изменений или «Сохранить» для сохранения нового текста измененной заметки. Модальное окно с редактированием заметки представлено на рисунке 5.15.

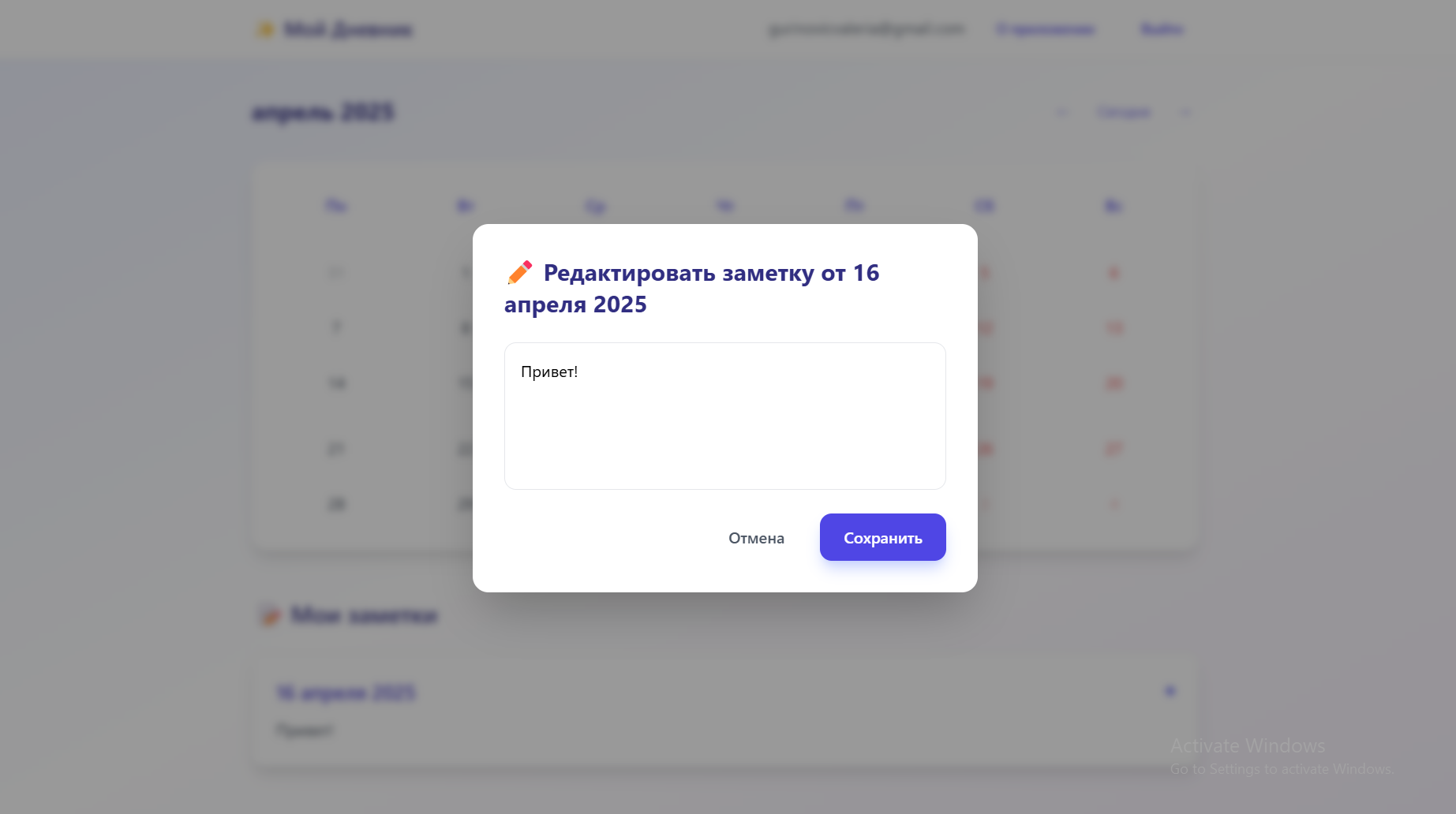


Рисунок 5.15 – Модальное окно для редактирования заметки

При нажатии на кнопку «Удалить» пользователю отобразится окно подтверждения с двумя вариантами действий – «Отмена» и «Удалить». С его помощью пользователь сможет либо отказаться от удаления и сохранить заметку, либо подтвердить действие и окончательно удалить её. В случае подтверждения заметка будет удалена из списка «Мои заметки». Окно подтверждения удаления заметки представлено на рисунке 5.16.

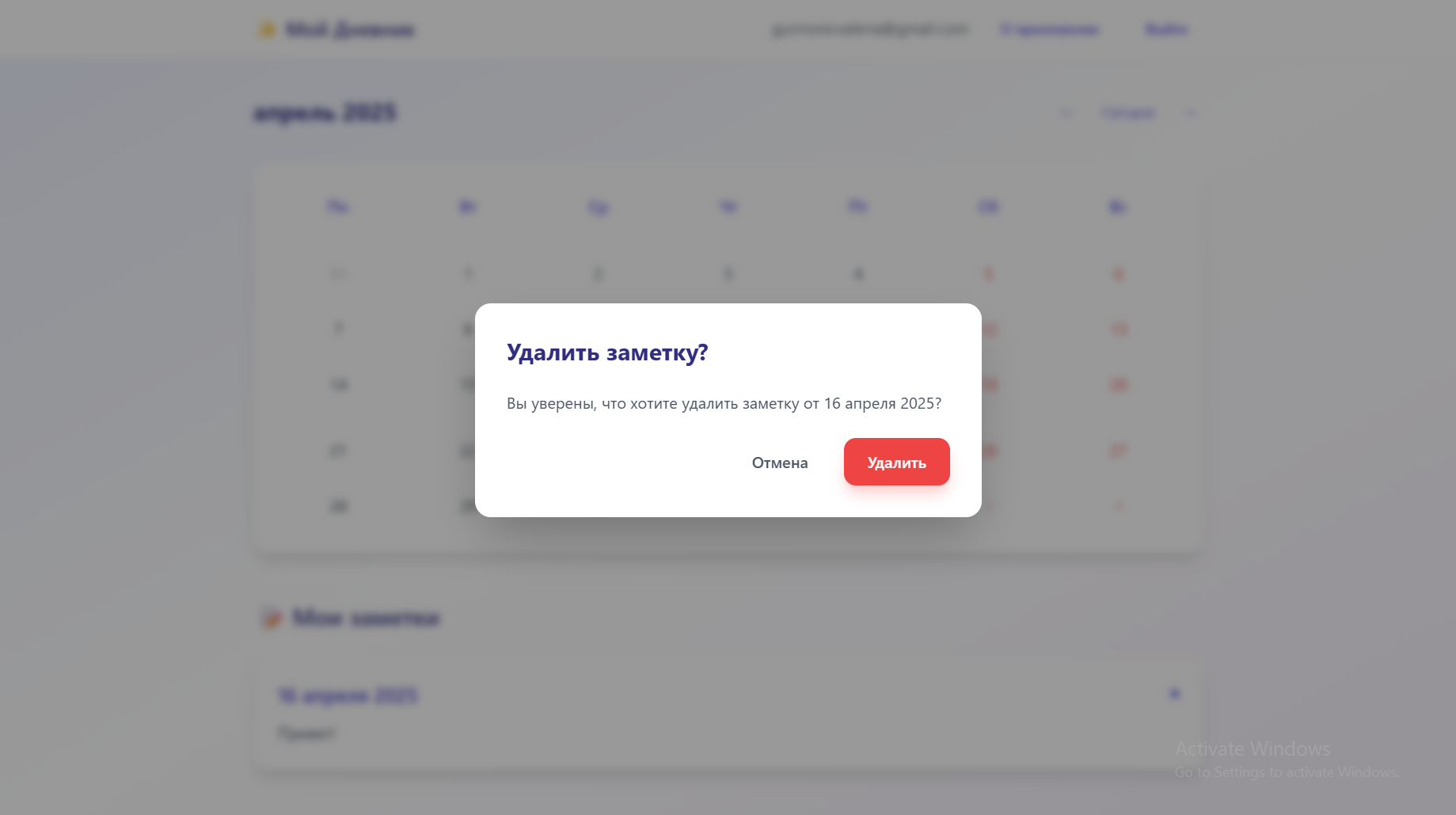


Рисунок 5.16 – Подтверждающее окно для удаления заметки

### 5.4.3 О приложении

После нажатия на кнопку «О приложении» перед пользователем появляется модальное окно, содержащее краткую справочную информацию о функционале и назначении приложения. При желании пользователь может закрыть это окно, воспользовавшись кнопкой «Закрыть». Модальное окно с информацией о приложении представлено на рисунке 5.17.



Рисунок 5.17 – Модальное окно для просмотра информации о приложении

## **5.5 Выход**

Для завершения работы в системе пользователь может воспользоваться кнопкой «Выход», после нажатия по которой он будет автоматически перенаправлен на страницу авторизации. Если в браузере были сохранены данные для входа, поля формы авторизации заполнятся ими автоматически, облегчая повторный вход, иначе – поля будут пустыми. Пример страницы авторизации с сохраненными данными после выхода из системы представлен на рисунке 5.18.

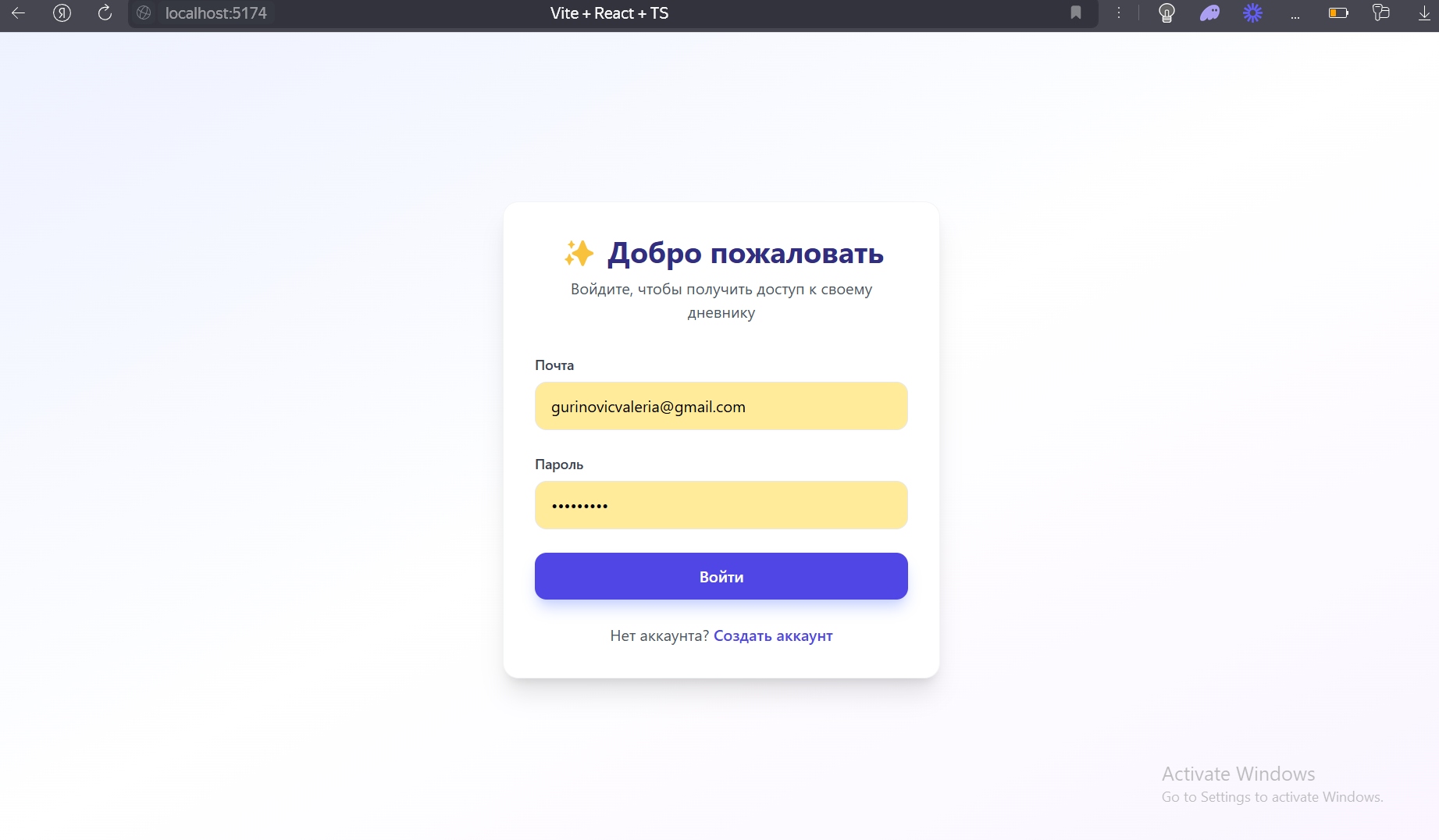


Рисунок 5.18 – Окно авторизации с сохраненными данными после выхода

из системы

ЗАКЛЮЧЕНИЕ

В ходе выполнения практического задания по разработке приложения «Электронный ежедневник» были успешно решены все задачи, поставленные во введении.

В разделе «Анализ существующих аналогов» проведено исследование популярных электронных ежедневников, выявлены их преимущества и недостатки. Это позволило определить ключевые требования к проектируемому приложению и сформировать обоснованную концепцию.

В разделе «Постановка задачи» были обозначены цели проекта: создание интуитивно понятного, функционального и надежного средства для планирования задач и управления расписанием. Определена целевая аудитория и сформулированы ключевые функции, которые должны быть реализованы.

В разделе «Разработка программного средства» выбран стек технологий, разработана архитектура, реализованы модули для работы с задачами, календарем, напоминаниями и интерфейсом пользователя. Приложение спроектировано с учетом гибкости и масштабируемости, что позволяет в будущем расширять функциональность без необходимости полной переработки системы.

В разделе «Тестирование и проверка работоспособности» проведена оценка приложения на предмет соответствия заявленным требованиям, выявлены и устранены ошибки, обеспечена стабильная работа всех функций.

В разделе «Руководство по установке и использованию» представлены подробные инструкции по настройке, запуску и использованию электронного ежедневника, что делает его доступным и удобным для конечного пользователя.

Разработанное приложение предоставляет пользователям следующие возможности:

– удобный доступ к приложению через браузер по веб-ссылке без необходимости установки дополнительного программного обеспечения;

– регистрация новых пользователей с валидацией введённых данных, таких как электронная почта и надёжный пароль;

– авторизация пользователей с проверкой корректности учётных данных и возможностью уведомления об ошибках при вводе;

– переход к главной странице после успешной авторизации или регистрации;

– просмотр и навигация по интерактивному календарю с возможностью выбора нужной даты и отображением текущего дня;

– создание, редактирование и удаление личных текстовых заметок, связанных с определёнными датами в календаре;

– визуальное обозначение дат с сохранёнными заметками посредством точек под числами в календаре;

– отображение функциональной панели управления заметками при наведении курсора (редактирование и удаление);

– подтверждение перед сохранением или удалением заметок для исключения случайных действий;

– возможность вернуться к текущей дате с помощью кнопки «Сегодня»;

– просмотр справочной информации о назначении и возможностях приложения в модальном окне «О приложении»;

– завершение сеанса с помощью кнопки «Выход» и последующим возвратом на страницу авторизации.

Таким образом, разработанное приложение эффективно выполняет задачи по управлению временем и личной продуктивностью, облегчает повседневное планирование и снижает использование бумажных носителей, что также оказывает положительное влияние на окружающую среду.

В перспективе возможно дальнейшее развитие функционала, включая:

* интеграция с внешними сервисами (например, Google Calendar или Microsoft Outlook);
* расширенная аналитика по заметкам и активности пользователя (статистика, графики, диаграммы);
* система персонализированных рекомендаций на основе пользовательской активности;
* добавление напоминаний и уведомлений;
* реализация мобильной версии или отдельного приложения для iOS и Android;
* возможность синхронизации данных между устройствами через облако.

Приложение обладает гибкой архитектурой, устойчивой работой и открыто для масштабирования, что делает его надёжной основой для дальнейших усовершенствований и интеграции в цифровую экосистему пользователя.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] Глухова Л.А. Основы алгоритмизации и программирования: лабораторный практикум / Л.А. Глухова, Е.П. Фадеева, Е.Е. Фадеева. – Минск: БГУИР, 2007. – 4 ч.

[2] Глухова Л.А. Основы алгоритмизации и программирования: лабораторный практикум / Л.А. Глухова, Е.П. Фадеева, Е.Е. Фадеева. – Минск: БГУИР, 2007. – 3 ч.

[3] Серебряная Л.В. Структуры и алгоритмы обработки данных: учеб.-метод. пособие / Л.В. Серебряная, И.М. Марина. – Минск: БГУИР, 2013. – 5 с.

[4] Топ 15 онлайн-ежедневников для работы и личных дел [Электронный ресурс]. – Режим доступа: https://blog.teamstorm.io/podbiraem-luchshij-onlajn-ezhednevnik-dlya-rabochih-zadach-i-lichnyh-del/. – Дата доступа: 10.04.2025.

[5] Электронные ежедневники: преимущества и недостатки [Электронный ресурс]. – Режим доступа: https://sky.pro/wiki/lifestyle/elektronnye-ezhednevniki-preimushestva-i-nedostatki/. – Дата доступа: 10.04.2025.

[6] Почему электронный планировщик лучше бумажного [Электронный ресурс]. – Режим доступа: <https://singularity-app.ru/blog/avtomatizaciya-tajm-menedzhmenta/>. – Дата доступа: 10.04.2025.

Приложение А

(обязательное)

Исходный код программы

import { useState, useEffect } from 'react'

import { useNavigate } from 'react-router-dom'

import { useLocalStorageState } from '@/hooks/use-local-storage'

import {

  format,

  addMonths,

  subMonths,

  startOfMonth,

  endOfMonth,

  eachDayOfInterval,

  isSameMonth,

  startOfWeek,

  endOfWeek

} from 'date-fns'

import { ru } from 'date-fns/locale'

import { Link } from 'react-router-dom'

const DiaryEntry = ({

  date,

  note,

  onEdit,

  onDelete,

}: {

  date: string

  note?: string

  onEdit: () => void

  onDelete: () => void

}) => (

  <div className="p-6 bg-white rounded-xl shadow-lg mb-4 hover:shadow-xl transition-shadow duration-300 border border-gray-100 break-words relative group">

    <button

      onClick={(e) => {

        e.stopPropagation();

        onEdit();

      }}

      className="absolute top-4 right-12 opacity-0 group-hover:opacity-100 transition-opacity p-2 hover:bg-blue-50 rounded-full text-blue-500"

      aria-label="Редактировать"

    >

      ✎

    </button>

    <button

      onClick={(e) => {

        e.stopPropagation();

        onDelete();

      }}

      className="absolute top-4 right-4 opacity-0 group-hover:opacity-100 transition-opacity p-2 hover:bg-red-50 rounded-full text-red-500"

      aria-label="Удалить"

    >

      ✕

    </button>

    <div className="flex justify-between items-center mb-3">

      <h3 className="text-xl font-semibold text-indigo-700">

        {format(new Date(date), "dd MMMM yyyy", { locale: ru })}

      </h3>

      <div className="w-2 h-2 bg-indigo-500 rounded-full flex-shrink-0"></div>

    </div>

    <p className="text-gray-700 whitespace-pre-wrap leading-relaxed overflow-auto max-h-[300px]">

      {note || 'Нет заметок на этот день'}

    </p>

  </div>

)

export default function HomePage() {

  const navigate = useNavigate()

  const [currentUser] = useLocalStorageState<{ email: string } | null>('currentUser', null)

  const [currentMonth, setCurrentMonth] = useState(new Date())

  const [selectedDate, setSelectedDate] = useState(format(new Date(), 'yyyy-MM-dd'))

  const [allNotes, setAllNotes] = useLocalStorageState<Record<string, Record<string, string>>>('userNotes', {})

  const [newNote, setNewNote] = useState('')

  const [showAddNoteModal, setShowAddNoteModal] = useState(false)

  const [showConfirmationModal, setShowConfirmationModal] = useState(false)

  const [editDate, setEditDate] = useState<string | null>(null)

const [editNote, setEditNote] = useState('')

const [showEditModal, setShowEditModal] = useState(false)

const [dateToDelete, setDateToDelete] = useState<string | null>(null)

const [showEditConfirmationModal, setShowEditConfirmationModal] = useState(false)

const [editPendingNote, setEditPendingNote] = useState<{ date: string, note: string } | null>(null)

const [showAboutModal, setShowAboutModal] = useState(false)

  const userNotes = currentUser ? (allNotes[currentUser.email] || {}) : {}

  const handleLogout = () => {

    navigate('/login')

  }

  const handleShowAbout = () => {

    setShowAboutModal(true)

  }

  const handleCloseAbout = () => {

    setShowAboutModal(false)

  }

  const handlePreviousMonth = () => {

    setCurrentMonth(prev => subMonths(prev, 1))

  }

  const handleNextMonth = () => {

    setCurrentMonth(prev => addMonths(prev, 1))

  }

  const getDaysInMonth = (date: Date) => {

    const start = startOfWeek(startOfMonth(date), { locale: ru, weekStartsOn: 1 })

    const end = endOfWeek(endOfMonth(date), { locale: ru, weekStartsOn: 1 })

    return eachDayOfInterval({ start, end })

  }

  const formatMonthYear = (date: Date) => {

    const monthIndex = date.getMonth() as 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11

    const month = ru.localize?.month(monthIndex, { width: 'wide' })

    const year = format(date, 'yyyy')

    return `${month} ${year}`

  }

  const currentMonthDays = getDaysInMonth(currentMonth)

  const handleDateClick = (date: Date) => {

    const formattedDate = format(date, 'yyyy-MM-dd')

    setSelectedDate(formattedDate)

    setShowAddNoteModal(true)

  }

  const saveNote = () => {

    setShowConfirmationModal(true)

  }

  const confirmSaveNote = () => {

    if (!newNote.trim() || !currentUser) return

    const updatedUserNotes = {

      ...userNotes,

      [selectedDate]: newNote.trim()

    }

    setAllNotes(prev => {

      const newNotes = {

        ...prev,

        [currentUser.email]: updatedUserNotes

      }

      window.localStorage.setItem('userNotes', JSON.stringify(newNotes))

      return newNotes

    })

    setNewNote('')

    setShowAddNoteModal(false)

    setShowConfirmationModal(false)

  }

  const deleteNote = (date: string) => {

    if (!currentUser) return;

    const updatedUserNotes = { ...userNotes };

    delete updatedUserNotes[date];

    setAllNotes(prev => {

      const newNotes = {

        ...prev,

        [currentUser.email]: updatedUserNotes

      };

      window.localStorage.setItem('userNotes', JSON.stringify(newNotes));

      return newNotes;

    });

  };

  if (!currentUser) {

    navigate('/login')

    return null

  }

  const isWeekend = (date: Date) => {

    const day = date.getDay()

    return day === 0 || day === 6

  }

  const handleEdit = (date: string) => {

    setEditDate(date);

    setEditNote(userNotes[date] || '');

    setShowEditModal(true);

    setEditPendingNote({ date, note: userNotes[date] || '' });

  }

  const saveEditedNote = () => {

    if (!editDate) return

    setEditPendingNote({ date: editDate, note: editNote })

    setShowEditModal(false)

    setShowEditConfirmationModal(true)

  }

  const confirmEditNote = () => {

    if (!editPendingNote || !currentUser) return

    const { date, note } = editPendingNote

    const updatedUserNotes = {

      ...userNotes,

      [date]: note.trim()

    }

    setAllNotes(prev => {

      const newNotes = {

        ...prev,

        [currentUser.email]: updatedUserNotes

      }

      window.localStorage.setItem('userNotes', JSON.stringify(newNotes))

      return newNotes

    })

    setEditPendingNote(null)

    setShowEditConfirmationModal(false)

  }

  const confirmDelete = () => {

    if (!currentUser || !dateToDelete) return

    const updatedUserNotes = { ...userNotes }

    delete updatedUserNotes[dateToDelete]

    setAllNotes(prev => {

      const newNotes = {

        ...prev,

        [currentUser.email]: updatedUserNotes,

      }

      window.localStorage.setItem('userNotes', JSON.stringify(newNotes))

      return newNotes

    })

    setDateToDelete(null)

    setEditPendingNote(null)

    setShowEditConfirmationModal(false)

  }

  return (

    <div className="min-h-screen bg-gradient-to-br from-indigo-50 via-white to-purple-50">

    <nav className="bg-white shadow-sm">

      <div className="max-w-5xl mx-auto px-4 sm:px-6 lg:px-8">

        <div className="flex justify-between items-center h-16">

          <div className="flex items-center">

            <h1 className="text-xl font-semibold text-indigo-900">✨ Мой Дневник</h1>

          </div>

          <div className="flex items-center space-x-4">

          <span className="text-gray-600">{currentUser?.email}</span>

          <button onClick={handleShowAbout} className="px-4 py-2 text-sm font-medium text-indigo-600 hover:text-indigo-700 focus:outline-none">О приложении</button>

            <button onClick={handleLogout} className="px-4 py-2 text-sm font-medium text-indigo-600 hover:text-indigo-700 focus:outline-none">Выйти</button>

          </div>

        </div>

      </div>

    </nav>

      <div className="max-w-5xl mx-auto p-8">

        <div className="flex justify-between items-center mb-8">

          <h2 className="text-2xl font-bold text-indigo-900">{formatMonthYear(currentMonth)}</h2>

          <div className="flex space-x-2">

            <button onClick={handlePreviousMonth} className="p-2 rounded-lg hover:bg-indigo-100 text-indigo-600 transition-colors">←</button>

            <button onClick={() => setCurrentMonth(new Date())} className="px-3 py-1 rounded-lg hover:bg-indigo-100 text-indigo-600 transition-colors text-sm">Сегодня</button>

            <button onClick={handleNextMonth} className="p-2 rounded-lg hover:bg-indigo-100 text-indigo-600 transition-colors">→</button>

          </div>

        </div>

        <div className="bg-white rounded-2xl shadow-lg p-6 mb-8">

          <div className="grid grid-cols-7 gap-2 mb-6">

            {['Пн', 'Вт', 'Ср', 'Чт', 'Пт', 'Сб', 'Вс'].map(day => (

              <div key={day} className="text-center font-medium py-2 text-indigo-600">{day}</div>

            ))}

          </div>

          <div className="grid grid-cols-7 gap-2">

            {currentMonthDays.map((date, index) => {

              const dateStr = format(date, 'yyyy-MM-dd')

              const hasNote = userNotes[dateStr]

              const isCurrentMonth = isSameMonth(date, currentMonth)

              return (

                <div

                  key={index}

                  onClick={() => handleDateClick(date)}

                  className={`

                    p-3 text-center rounded-lg cursor-pointer transition-all duration-200

                    ${hasNote ? 'bg-indigo-100 hover:bg-indigo-200' : 'hover:bg-gray-100'}

                    ${isWeekend(date) ? 'text-red-500' : 'text-gray-700'}

                    ${dateStr === selectedDate ? 'ring-2 ring-indigo-500' : ''}

                    ${!isCurrentMonth ? 'opacity-40' : ''}

                  `}

                >

                  {format(date, 'd')}

                  {hasNote && <div className="w-1.5 h-1.5 bg-indigo-500 rounded-full mx-auto mt-1"></div>}

                </div>

              )

            })}

          </div>

        </div>

        {showAddNoteModal && (

          <div className="fixed inset-0 bg-black bg-opacity-40 backdrop-blur-sm flex items-center justify-center z-50 p-4">

            <div className="bg-white p-6 md:p-8 rounded-2xl w-full max-w-[32rem] shadow-2xl transform transition-all max-h-[90vh] flex flex-col">

              <h2 className="text-2xl font-bold mb-6 text-indigo-900">

                ✍️ Заметка для {format(new Date(selectedDate), "dd MMMM yyyy", { locale: ru })}

              </h2>

              <textarea

                value={newNote}

                onChange={(e) => setNewNote(e.target.value)}

                className="w-full p-4 border border-gray-200 rounded-xl mb-6 focus:ring-2 focus:ring-indigo-500 focus:border-transparent transition-all resize-none flex-grow"

                placeholder="Запишите свои мысли здесь..."

                style={{ minHeight: '150px', maxHeight: '50vh' }}

              />

              <div className="flex justify-end gap-3 mt-auto">

                <button onClick={() => setShowAddNoteModal(false)} className="px-4 md:px-6 py-3 text-gray-600 hover:bg-gray-100 rounded-xl transition-colors duration-200 font-medium">Отмена</button>

                <button onClick={saveNote} className="px-4 md:px-6 py-3 bg-indigo-600 text-white rounded-xl hover:bg-indigo-700 transition-colors duration-200 font-medium shadow-lg shadow-indigo-200">Сохранить заметку</button>

              </div>

            </div>

          </div>

        )}

{showAboutModal && (

        <div className="fixed inset-0 bg-black bg-opacity-40 backdrop-blur-sm flex items-center justify-center z-50 p-4">

          <div className="bg-white p-6 md:p-8 rounded-2xl w-full max-w-[32rem] shadow-2xl transform transition-all max-h-[90vh] flex flex-col">

            <h2 className="text-2xl font-bold mb-6 text-indigo-900">✨ О приложении</h2>

            <p className="text-gray-600 mb-6">

              Это приложение представляет собой дневник, который позволяет пользователю записывать свои мысли на каждый день. Вы можете организовывать свои планы и мысли с помощью календаря и заметок. Приятного использования!

            </p>

            <div className="flex justify-end gap-3">

              <button onClick={handleCloseAbout} className="px-4 md:px-6 py-3 text-gray-600 hover:bg-gray-100 rounded-xl transition-colors duration-200 font-medium">Закрыть</button>

            </div>

          </div>

        </div>

      )}

        {showConfirmationModal && (

          <div className="fixed inset-0 bg-black bg-opacity-40 backdrop-blur-sm flex items-center justify-center z-50 p-4">

            <div className="bg-white p-6 md:p-8 rounded-2xl w-full max-w-[32rem] shadow-2xl transform transition-all">

              <h2 className="text-2xl font-bold mb-6 text-indigo-900">Подтвердите сохранение заметки</h2>

              <p className="text-gray-600 mb-6">Вы уверены, что хотите сохранить эту заметку?</p>

              <div className="flex justify-end gap-3">

                <button onClick={() => setShowConfirmationModal(false)} className="px-4 md:px-6 py-3 text-gray-600 hover:bg-gray-100 rounded-xl transition-colors duration-200 font-medium">Отменить</button>

                <button onClick={confirmSaveNote} className="px-4 md:px-6 py-3 bg-indigo-600 text-white rounded-xl hover:bg-indigo-700 transition-colors duration-200 font-medium shadow-lg shadow-indigo-200">Подтвердить</button>

              </div>

            </div>

          </div>

        )}

        {showEditModal && (

  <div className="fixed inset-0 bg-black bg-opacity-40 backdrop-blur-sm flex items-center justify-center z-50 p-4">

    <div className="bg-white p-6 md:p-8 rounded-2xl w-full max-w-[32rem] shadow-2xl transform transition-all max-h-[90vh] flex flex-col">

      <h2 className="text-2xl font-bold mb-6 text-indigo-900">

        ✏️ Редактировать заметку от {format(new Date(editDate!), "dd MMMM yyyy", { locale: ru })}

      </h2>

      <textarea

        value={editNote}

        onChange={(e) => setEditNote(e.target.value)}

        className="w-full p-4 border border-gray-200 rounded-xl mb-6 focus:ring-2 focus:ring-indigo-500 focus:border-transparent transition-all resize-none flex-grow"

        placeholder="Измените заметку..."

        style={{ minHeight: '150px', maxHeight: '50vh' }}

      />

      <div className="flex justify-end gap-3 mt-auto">

        <button onClick={() => setShowEditModal(false)} className="px-4 md:px-6 py-3 text-gray-600 hover:bg-gray-100 rounded-xl transition-colors duration-200 font-medium">Отмена</button>

        <button onClick={saveEditedNote} className="px-4 md:px-6 py-3 bg-indigo-600 text-white rounded-xl hover:bg-indigo-700 transition-colors duration-200 font-medium shadow-lg shadow-indigo-200">Сохранить</button>

      </div>

    </div>

  </div>

)}

{dateToDelete && (

  <div className="fixed inset-0 bg-black bg-opacity-40 backdrop-blur-sm flex items-center justify-center z-50 p-4">

    <div className="bg-white p-6 md:p-8 rounded-2xl w-full max-w-[32rem] shadow-2xl transform transition-all">

      <h2 className="text-2xl font-bold mb-6 text-indigo-900">Удалить заметку?</h2>

      <p className="text-gray-600 mb-6">

        Вы уверены, что хотите удалить заметку от {format(new Date(dateToDelete), "dd MMMM yyyy", { locale: ru })}?

      </p>

      <div className="flex justify-end gap-3">

        <button onClick={() => setDateToDelete(null)} className="px-4 md:px-6 py-3 text-gray-600 hover:bg-gray-100 rounded-xl transition-colors duration-200 font-medium">Отмена</button>

        <button onClick={confirmDelete} className="px-4 md:px-6 py-3 bg-red-500 text-white rounded-xl hover:bg-red-600 transition-colors duration-200 font-medium shadow-lg shadow-red-200">Удалить</button>

      </div>

    </div>

  </div>

)}

{showEditConfirmationModal && editPendingNote && (

  <div className="fixed inset-0 bg-black bg-opacity-40 backdrop-blur-sm flex items-center justify-center z-50 p-4">

    <div className="bg-white p-6 md:p-8 rounded-2xl w-full max-w-[32rem] shadow-2xl transform transition-all">

      <h2 className="text-2xl font-bold mb-6 text-indigo-900">Подтвердите изменения</h2>

      <p className="text-gray-600 mb-6">

        Вы уверены, что хотите сохранить изменения для заметки от {format(new Date(editPendingNote.date), "dd MMMM yyyy", { locale: ru })}?

      </p>

      <div className="flex justify-end gap-3">

        <button onClick={() => {

          setShowEditConfirmationModal(false)

          setShowEditModal(true)

        }} className="px-4 md:px-6 py-3 text-gray-600 hover:bg-gray-100 rounded-xl transition-colors duration-200 font-medium">Отмена</button>

        <button onClick={confirmEditNote} className="px-4 md:px-6 py-3 bg-indigo-600 text-white rounded-xl hover:bg-indigo-700 transition-colors duration-200 font-medium shadow-lg shadow-indigo-200">Подтвердить</button>

      </div>

    </div>

  </div>

)}

        <div className="mt-12 space-y-6">

          <h2 className="text-2xl font-semibold text-indigo-900 mb-6">📝 Мои заметки</h2>

          {Object.entries(userNotes).length === 0 ? (

            <div className="text-center py-12 text-gray-500">

              Пока нет заметок. Нажмите на день в календаре, чтобы начать! ✨

            </div>

          ) : (

            Object.entries(userNotes)

              .sort((a, b) => new Date(b[0]).getTime() - new Date(a[0]).getTime())

              .map(([date, note]) => (

                <DiaryEntry

                key={date}

                date={date}

                note={note}

                onEdit={() => handleEdit(date)}

                onDelete={() => setDateToDelete(date)}

              />

              ))

          )}

        </div>

      </div>

    </div>

  )

}