АННОТАЦИЯ

Дипломный проект на тему: веб-приложение для автоматизации командной работы состоит из программного средства, пояснительной записки и графической части.

Программное средство предназначено для быстрой и удобной командной работы над проектами.

Программное средство разработано в среде программирования Visual Studio Code с использованием языков программирования: Python, SQL, JavaScript, языка гипертекстовой разметки HTML и языка описания внешнего вида документа CSS. База данных приложения спроектирована и реализована в СУБД SQLite с использованием Flask-SQLAlchemy.

Пояснительная записка выполнена на 75 листах, содержит шесть разделов и два приложения.

Графическая часть выполнена на двух листах, включает схему базы данных и диаграмму прецедентов.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc199681678)

[1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 6](#_Toc199681679)

[2. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 8](#_Toc199681680)

[2.1. Постановка задачи 8](#_Toc199681681)

[2.2. Описание организации данных 9](#_Toc199681682)

[2.3. Описание программных средств 12](#_Toc199681683)

[2.4. Проектирование интерфейса 14](#_Toc199681684)

[2.5. Описание физической структуры программы 22](#_Toc199681685)

[3. ТЕСТИРОВАНИЕ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ 24](#_Toc199681686)

[4. РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО   
СРЕДСТВА 35](#_Toc199681687)

[5. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО СРЕДСТВА 41](#_Toc199681688)

[6. ОХРАНА ТРУДА 48](#_Toc199681689)

[6.1. Правовые и организационные вопросы охраны труда 48](#_Toc199681696)

[6.2. Производственная санитария и гигиена труда 49](#_Toc199681697)

[6.3. Техника безопасности (электробезопасность) 51](#_Toc199681698)

[6.4. Пожарная безопасность 53](#_Toc199681699)

[ЗАКЛЮЧЕНИЕ 55](#_Toc199681700)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 56](#_Toc199681701)

[ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ 57](#_Toc199681702)

[ПРИЛОЖЕНИЕ 1 (ОБЯЗАТЕЛЬНОЕ) – УКРУПНЕННЫЕ НОРМЫ ЗАТРАТ ТРУДА НА РАЗРАБОТКУ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 58](#_Toc199681703)

[ПРИЛОЖЕНИЕ 2 (ОБЯЗАТЕЛЬНОЕ) – ЛИСТИНГ ПРОГРАММЫ 61](#_Toc199681704)

# 

# ВВЕДЕНИЕ

В современном мире, характеризующемся высокой динамикой и сложностью проектов, эффективная командная работа является ключом к успеху. Возрастающая популярность удаленных и гибридных форматов сотрудничества лишь усиливает потребность в инструментах, способных обеспечить прозрачность, координацию и продуктивность команд. Веб-приложение для автоматизации командной работы призвано стать решением для тех, кто стремится к упорядочиванию рабочих процессов, улучшению взаимодействия и повышению общей эффективности проектной деятельности. Актуальность данной темы обусловлена широким распространением гибких методологий управления проектами (Agile, Kanban) и высоким спросом на цифровые инструменты, такие как Trello, Asana, Jira, Yougile, которые помогают визуализировать рабочие процессы и повышать эффективность.

Анализ сферы показывает, что многие существующие системы управления проектами обладают избыточной функциональностью для небольших команд или учебных целей, либо требуют значительных затрат на внедрение. Веб-приложение для автоматизации командной работы предлагает упрощенный, но эффективный подход, фокусируясь на ключевых возможностях Kanban-систем и обеспечивая легкость освоения и использования. Проект сочетает в себе базовую функциональность управления задачами с дополнительными возможностями, такими как комментарии, теги и управление участниками.

Цель проекта − создание удобного и функционального веб-приложения, которое позволит пользователям регистрироваться, создавать доски (проекты), управлять на них колонками (статусами задач) и карточками (задачами), а также использовать дополнительный функционал для эффективной совместной работы.

Для достижения этой цели были поставлены следующие задачи:

− исследовать рынок существующих систем управления проектами;

− определить ключевые требования и функции для веб-приложения;

− разработать интерфейс, который будет интуитивно понятен и удобен для пользователя;

− обеспечить интеграцию с базой данных;

− реализовать CRUD-операции для всех основных сущностей, систему аутентификации пользователей, управление участниками досок, систему комментариев и тегов;

− обеспечить возможности поиска, фильтрации и сортировки карточек на доске.

Структура дипломного проекта включает в себя следующие разделы:

1. «Назначение и область применения» – описывается назначение программы, краткая характеристика области применения программы. Проводится анализ предметной области и инфологическое проектирование разрабатываемой базы данных. Приводится обзор существующих подходов к решению задачи, их достоинства и недостатки, отличие разработанной программы от аналогов. Рассматриваются существующие аналоги.
2. «Технические характеристики» – рассматриваются следующие вопросы: требования к функциям (раздел «Постановка задачи»); формирование требований к системе в целом; требования к организации данных в проекте, описание их организации, структуры, построение концептуальной и логической модели данных, способа кодирования (при необходимости) и предварительной подготовки (раздел «Описание организации данных»); обоснование выбора программного обеспечения, используемого для разработки программного средства (раздел «Описание программных средств»); проектирование программного средства. (разделы «Проектирование интерфейса», «Описание физической структуры программы»).
3. «Тестирование и анализ полученных результатов» – описывается проверка работы программы на различных тестах и приводится анализ ожидаемых и полученных результатов, выводы по результатам тестирования.
4. «Руководство по использованию программного средства» – описывается последовательность действий пользователя, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведены описание функций, формата и возможных вариантов команд, с помощью которых пользователь управляет выполнением программы, а также ответы программы на эти команды.
5. «Экономическое обоснование разработки программного средства» – проводится оценка экономической эффективности разработки программы. Включает в себя расчет затрат на создание программного средства, прогнозируемую прибыль от его использования, оценку окупаемости проекта, а также анализ конкурентоспособности программного продукта на рынке.
6. «Охрана труда» – описываются меры по обеспечению безопасности и здоровья разработчиков в процессе работы над проектом, включая организацию рабочего места, профилактику профессиональных заболеваний и соблюдение норм трудового законодательства.
7. «Заключение» – описываются основные результаты, характеризующие полноту решения поставленных задач и подводящие итог содержания дипломного проекта

# 1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

Веб-приложение для автоматизации командной работы разрабатывается с целью предоставления пользователям удобного и интуитивно понятного инструмента для управления проектами и задачами в стиле Kanban, а также для организации эффективного взаимодействия внутри команды. Оно предназначено для использования малыми и средними командами, фрилансерами, студенческими группами и индивидуальными пользователями, которым требуется структурировать свою работу, отслеживать прогресс и улучшать совместную деятельность.

Преимущества веб-приложения включают:

1. Интуитивно понятный интерфейс, облегчающий создание досок, колонок, карточек и управление ими.
2. Возможность визуализации рабочего процесса и легкого перемещения задач (карточек) между стадиями (колонками) с помощью перетаскивания.
3. Четкое отображение информации по задачам: заголовок, описание, ответственные за задачу, теги, комментарии.
4. Функциональность для управления участниками доски, назначения ответственных за задачу на задачи, комментирования задач, использования тегов для категоризации и фильтрации.

Требования к аппаратным и операционным ресурсам для использования веб-приложения минимальны и включают:

* операционную систему (Windows, macOS, Linux), поддерживающую современные веб-браузеры;
* минимум 2GB оперативной памяти;
* надежное подключение к интернету для доступа к приложению, синхронизации данных и взаимодействия в реальном времени.

В процессе разработки веб-приложения для автоматизации командной работы был проведен анализ конкурентов, чтобы выявить их сильные и слабые стороны, а также определить уникальные преимущества разрабатываемого продукта. Одним из ключевых ориентиров и популярным решением на рынке является Trello.

Достоинства Trello:

1. Интуитивное визуальное управление: канбан-доски обеспечивают ясное представление о ходе выполнения задач и общем состоянии проекта.
2. Эффективные инструменты для командной работы: возможность назначать участников на карточки, оставлять комментарии, устанавливать сроки и использовать чек-листы способствует продуктивному взаимодействию.
3. Кроссплатформенная доступность и синхронизация: веб-версия и мобильные приложения для iOS и Android позволяют работать с досками и обновлять информацию с любого устройства в реальном времени.

Недостатки Trello:

1. Избыточность для простых задач: для небольших проектов или индивидуального использования некоторый функционал может быть излишним и усложнять интерфейс.
2. Ограничения бесплатной версии: количество улучшений, автоматизаций и другие продвинутые функции ограничены в бесплатном тарифе.
3. Меньший контроль над данными: как SaaS-решение, все данные хранятся на серверах Trello, что может быть не всегда приемлемо (в отличие от потенциальной возможности самостоятельного хостинга или контроля над локальной БД).

Уникальные преимущества веб-приложения для автоматизации командной работы:

* целенаправленная простота: веб-приложение сфокусировано на ключевом функционале Kanban-системы, обеспечивая легкость освоения и использования без перегрузки дополнительными, не всегда необходимыми функциями, что идеально для небольших команд;
* отсутствие лицензионных ограничений: приложение не требует покупки или подписок, предоставляя полный доступ ко всему заявленному функционалу без ограничений;
* интегрированный базовый набор для совместной работы: функции управления участниками, комментарии, теги и назначение ответственных за задачу являются частью основного функционала без необходимости подключения сторонних расширений.

Анализ конкурентов показывает, что на рынке существуют и другие мощные системы управления проектами, такие как Asana, Jira, Yougile, которые предлагают обширный набор инструментов для планирования, отслеживания и отчетности. Однако веб-приложение для автоматизации командной работы выделяется на их фоне своей ориентацией на упрощенную реализацию основных принципов Kanban, что делает его доступным решением для пользователей, которым не требуется вся сложность корпоративных систем.

# 2. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

# 2.1. Постановка задачи

Необходимо разработать веб-приложение для автоматизации командной работы, предназначенное для широкого круга пользователей, нуждающихся в эффективном инструменте для управления проектами и задачами. Целевая аудитория включает малые и средние команды в компаниях или стартапах, которым требуется простая и наглядная система для организации совместной работы над проектами.

Реализовать две роли, такие как: пользователь и администратор. Каждая роль имеет свои определенные возможности.

Для роли пользователя предусмотреть возможность регистрации и аутентификации, создания собственных досок, управления ими (редактирование названия, удаление, управление участниками). Пользователи могут быть приглашены на доски других пользователей. На доске пользователи (владелец и участники, имеющие право редактирования) могут создавать, просматривать, редактировать и удалять колонки и карточки, а также назначать ответственного за выполнение задачи. Карточки можно перетаскивать между колонками. Реализовать функции поиска карточек по названию и описанию, а также фильтрацию карточек по ответственным за задачу и тегам, и сортировку карточек внутри колонок. Пользователи могут добавлять, редактировать и удалять свои комментарии к карточкам, а также управлять тегами на уровне доски.

Для каждой карточки реализовать отображение заголовка, цветных меток, тегов, индикатора наличия описания и фото профиля пользователя, назначенного как ответственного за выполнение задачи, которая описана в карточке. В модальном окне карточки доступна детальная информация: заголовок, описание, список ответственных за задачу, список тегов карточки и комментарии.

В роли администратора предусмотреть доступ к панели администратора после авторизации. Панель администратора позволяет просматривать таблицу со всеми зарегистрированными пользователями (ID, изображение пользователя, имя, email, статус администратора). Для администратора реализовать возможность редактировать имя пользователя и его роли. Также для администратора осуществить возможность удалять пользователей, соблюдая ограничения: нельзя удалить себя, если он единственный администратор, и нельзя удалить пользователя, являющегося владельцем хотя бы из существующих досок.

# 2.2. Описание организации данных

Данные для веб-приложения необходимо хранить в базе данных SQLite. Для хранения данных о пользователях предусмотрена таблица «user». Информация о досках (проектах) будет храниться в таблице «board». Для управления колонками (статусами задач) на досках создается таблица «column». Данные о карточках (задачах) содержатся в таблице «card». Теги, используемые для классификации карточек, хранятся в таблице «tag». Комментарии к карточкам будут находиться в таблице «comment». Также используются ассоциативные таблицы «board\_members», «card\_assignees» и «card\_tags» для организации связей многие-ко-многим между пользователями и досками, пользователями и карточками, а также карточками и тегами соответственно.

Структура данных таблицы «user» представлена в таблице 2.1.

Таблица 2.1 – Структура данных таблицы «user»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Описание |
| id | Число | Уникальный идентификатор пользователя |
| username | Текст | Имя пользователя |
| email | Текст | Email пользователя |
| password\_hash | Текст | Хеш для хранения пароля |
| avatar\_url | Текст | Путь к файлу изображения пользователя |
| is\_admin | Логический | Определяет является ли пользователь администратором |

Структура данных таблицы «board» представлена в таблице 2.2.

Таблица 2.2 – Структура данных таблицы «board»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Описание |
| id | Число | Уникальный идентификатор доски |
| name | Текст | Название доски |
| user\_id | Число | Идентификатор, определяющий пользователя-владельца доски |

Структура данных таблицы «column» представлена в таблице 2.3.

Таблица 2.3 – Структура данных таблицы «column»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Описание |
| id | Число | Уникальный идентификатор колонки |
| name | Текст | Название колонки |
| position | Число | Порядковый номер колонки на доске для сортировки и перетаскивания |
| board\_id | Число | Определяет идентификатор доски, к которой принадлежит колонка |

Структура данных таблицы «card» представлена в таблице 2.4.

Таблица 2.4 – Структура данных таблицы «card»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Описание |
| id | Число | Уникальный идентификатор карточки |
| title | Текст | Заголовок карточки |
| description | Текст | Описание карточки |
| column\_id | Число | Определяет идентификатор колонки, к которой принадлежит карточка |

Структура данных таблицы «tag» представлена в таблице 2.5.

Таблица 2.5 – Структура данных таблицы «tag»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Описание |
| id | Число | Уникальный идентификатор тега |
| name | Текст | Название тега |
| color | Текст | Цвет тега, который хранится в HEX-формате |
| board\_id | Число | Определяет идентификатор доски, к которой принадлежит тег |

Структура данных таблицы «comment» представлена в таблице 2.6.

Таблица 2.6 – Структура данных таблицы «comment»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Описание |
| id | Число | Уникальный идентификатор комментария |
| text | Текст | Текст комментария |
| timestamp | Дата и время | Дата и время создания комментария |
| user\_id | Число | Идентификатор автора комментария |
| card\_id | Число | Идентификатор карточки, к которой относится комментарий |

Структура данных таблицы «board\_members» представлена в таблице 2.7.

Таблица 2.7 – Структура данных таблицы «board\_members»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Описание |
| user\_id | Число | Идентификатор пользователя |
| board\_id | Число | Идентификатор доски |

Структура данных таблицы «card\_assignees» представлена в таблице 2.7.

Таблица 2.8 – Структура данных таблицы «card\_assignees»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Описание |
| user\_id | Число | Идентификатор пользователя |
| card\_id | Число | Идентификатор карточки |

Структура данных таблицы «card\_tags» представлена в таблице 2.7.

Таблица 2.9 – Структура данных таблицы «card\_tags»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Описание |
| user\_id | Число | Идентификатор пользователя |
| tag\_id | Число | Идентификатор тега |

Входными данными программы являются: данные для регистрации и аутентификации пользователя, названия для создаваемых досок, колонок, тегов, заголовки и описания для карточек, текст комментариев, файлы изображения пользователя, поисковые запросы и параметры фильтрации.

Выходными данными программы являются: информация о пользователях (профиль, список участников), досках (список досок, содержимое доски с колонками и карточками), карточках (детальная информация в модальном окне, включая описание, ответственных за задачу, теги, комментарии), тегах и комментариях; результаты поиска и фильтрации, сообщения об успешном выполнении операций или ошибках.

# 2.3. Описание программных средств

Программное средство разработано в среде программирования Visual Studio Code. Серверная часть реализован на языке программирования Python 3 с использованием веб-фреймворка Flask. Клиентская часть создана с использованием языка гипертекстовой разметки HTML5, языка описания внешнего вида CSS3 и языка JavaScript. База данных приложения спроектирована и реализована в СУБД SQLite с использованием Flask-SQLAlchemy.

HTML5 (HyperText Markup Language) – это стандартный язык разметки для создания веб-страниц. Он позволяет встраивать различные элементы, такие как текст, изображения, ссылки, формы и многое другое, предоставляя семантическую структуру для веб-контента.

Плюсы HTML5:

1. Универсальность: HTML-документы могут быть открыты в любом веб-браузере на любой платформе.
2. Легкость изучения: HTML имеет простой синтаксис, который легко понять и использовать начинающим разработчикам.
3. Совместимость: HTML документы легко интегрируются с другими веб-технологиями, такими как CSS и JavaScript.

Минусы HTML:

1. Ограниченные возможности стилизации: без использования CSS, возможности стилизации HTML ограничены, что делает веб-страницы менее привлекательными.
2. Статичность: HTML сам по себе не позволяет создавать динамические веб-страницы, которые могут реагировать на действия пользователя.

CSS (Cascading Style Sheets) – это язык стилей, используемый для определения внешнего вида и форматирования документов, написанных на языке разметки.

Плюсы CSS:

1. Повышение гибкости: CSS позволяет легко изменять стиль элементов на всех страницах сайта, изменяя только один файл стилей.
2. Улучшение загрузки страниц: стили, определенные в CSS, могут быть кэшированы браузерами, что ускоряет загрузку страниц.
3. Разделение содержимого и дизайна: CSS позволяет отделить содержимое веб-страницы от её дизайна, что упрощает обслуживание и обновление сайта.

Минусы CSS:

1. Сложность: некоторые аспекты CSS могут быть сложными для понимания и использования, особенно для новичков.
2. Неоднородность браузеров: разные браузеры могут по-разному интерпретировать CSS-код, что может привести к несоответствию внешнего вида сайта.
3. Ограничения: несмотря на широкие возможности стилизации, есть определенные ограничения в CSS, которые могут потребовать использования JavaScript для реализации некоторых функций.

JavaScript – это язык программирования, чаще всего используемый для создания интерактивных веб-страниц на стороне клиента. В проекте используется ванильный JavaScript для реализации клиентской логики и AJAX-взаимодействий.

Плюсы JavaScript:

1. Интерактивность и динамизм: позволяет создавать богатые пользовательские интерфейсы, реагирующие на действия пользователя без перезагрузки страницы.
2. Широкое распространение: поддерживается всеми современными браузерами, имеет огромное сообщество и множество ресурсов.
3. Интеграция: легко интегрируется с HTML и CSS, а также с различными серверными технологиями.

К минусам JavaScript можно отнести:

1. Безопасность: код, выполняемый на стороне клиента, может быть подвержен определенным уязвимостям, если не соблюдать меры предосторожности.
2. Сложность управления состоянием: в крупных одностраничных приложениях управление состоянием может стать сложным без использования специализированных библиотек или фреймворков.
3. Производительность: неоптимизированный JavaScript-код может замедлять работу веб-страницы, особенно на менее производительных устройствах.

Visual Studio Code – это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом, плюсами которой являются: возможность бесплатного использования, кроссплатформенность, так как работает на MacOS, Ubuntu и Windows, а также встроенные подсказки.

К недостаткам возможно отнести высокий порог вхождения, так как среда разработки содержит множество встроенных функций, которые сложны в самостоятельном освоении.

Flask-SQLAlchemy – это расширение для Flask, которое упрощает использование SQLAlchemy в приложениях Flask. SQLAlchemy является мощным ORM (Object-Relational Mapper) и инструментарием SQL для Python, предоставляя гибкий способ работы с базами данных через объекты Python.

SQLite – это программная библиотека, реализующая самодостаточный, безсерверный, не требующий конфигурации транзакционный движок баз данных SQL.

SQLite является встраиваемой реляционной системой управления базами данных.

База данных (БД) – это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации.

Преимуществами БД являются: возможность систематизировать и хранить данные в одном месте, а также гибкость в редактировании, добавлении и удалении данных.

Основным недостатком использования БД является трудоемкость в заполнении, а также часты ошибки и неполадки при подключении.

# 2.4. Проектирование интерфейса

В данном проекте веб-приложение будет включать следующие основные страницы и представления для пользователя и администратора: страница авторизации, страница регистрации, панель администратора, страница моих досок, страница профиля пользователя, страница доски и модальное окно карточки.

На странице авторизации пользователь сможет войти в систему, используя email и пароль, представлено на рисунке 2.1. Также доступна опция «Запомнить меня» и переход на страницу регистрации. Администратор использует эту же страницу для входа. Элементы страницы авторизации представлены в таблице 2.10.

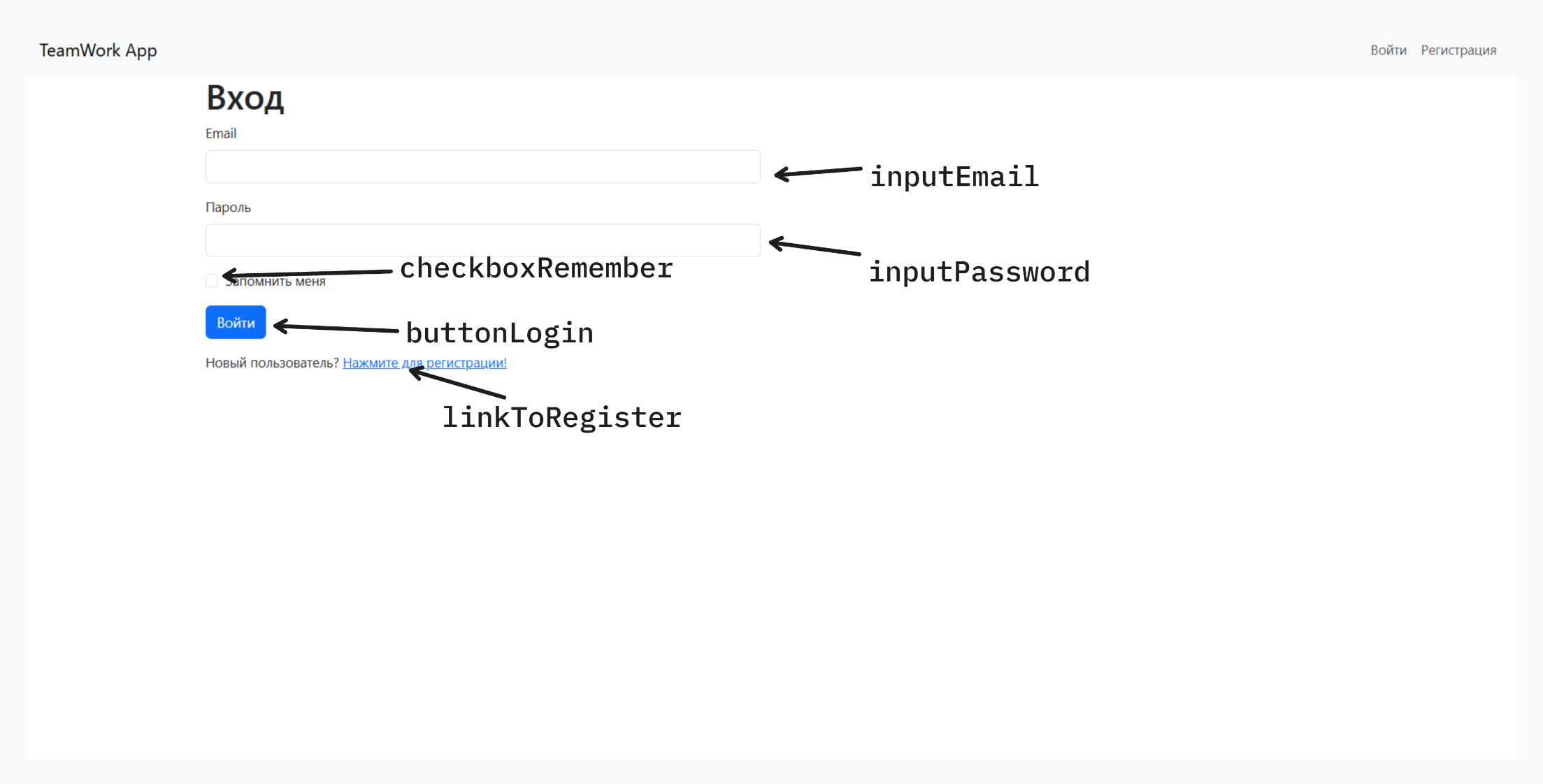


Рис.2.1. Страница авторизации

Таблица 2.10 – Элементы управления страницы авторизации

|  |  |  |
| --- | --- | --- |
| Элементы управления | Свойства | Назначение |
| inputEmail | Name = inputEmail | Поле для ввода email |
| inputPassword | Name = inputPassword | Поле для ввода пароля |
| checkboxRemember | Name = checkboxRemember | Флажок «Запомнить меня» |
| buttonLogin | Name = buttonLogin | Выполняет функцию авторизации пользователя |
| linkToRegister | Name = linkToRegister | Переход на форму регистрации пользователя |

На странице регистрации новый пользователь сможет создать аккаунт, указав имя пользователя, email и пароль, представлено на рисунке 2.2. Также доступен переход на страницу авторизации. Элементы страницы регистрации представлены в таблице 2.11.

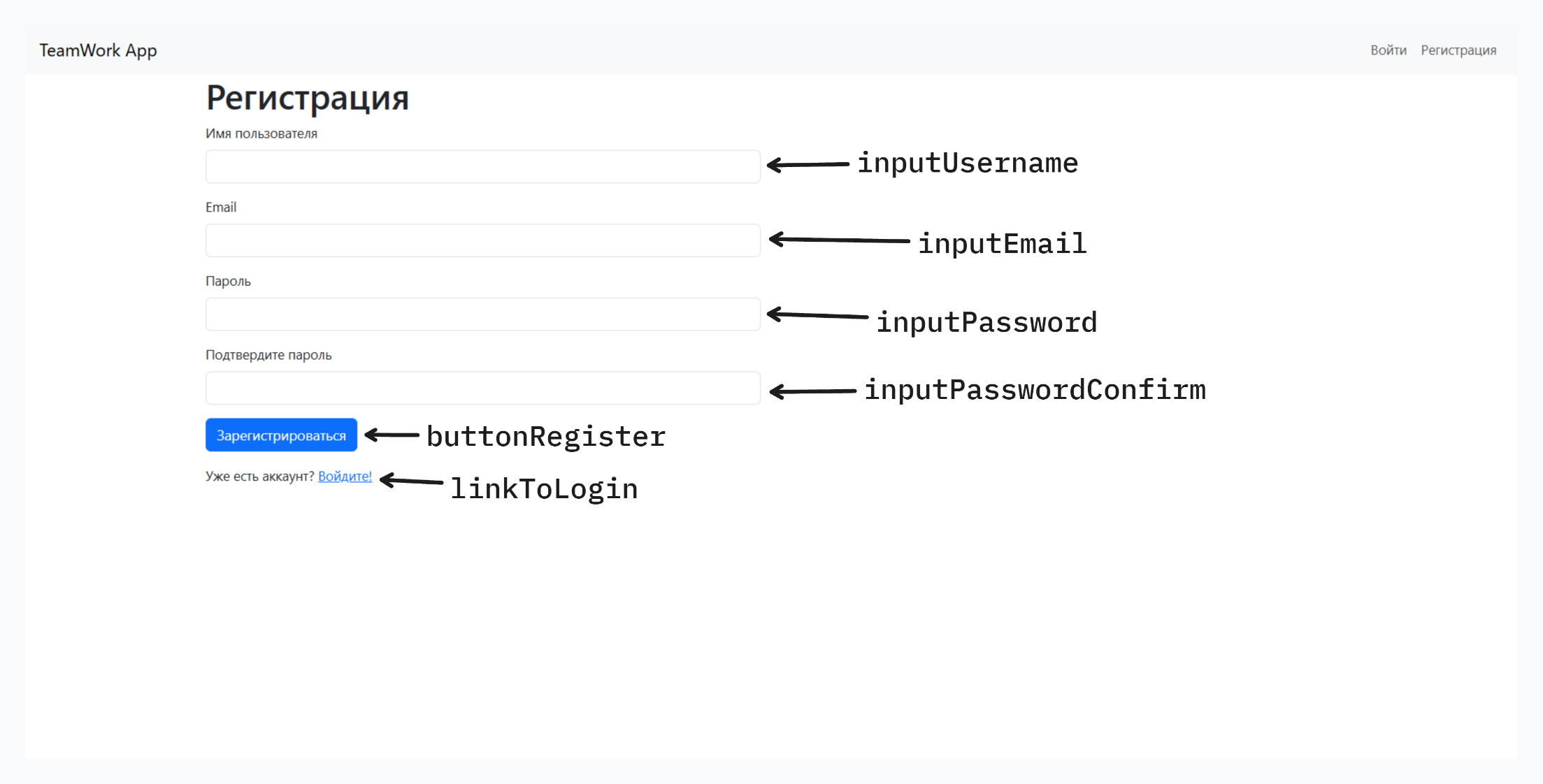


Рис.2.2. Страница регистрации

Таблица 2.11 – Элементы управления страницы регистрации

|  |  |  |
| --- | --- | --- |
| Элементы управления | Свойства | Назначение |
| inputUsername | Name = inputUsername | Поле для ввода имени пользователя |
| inputEmail | Name = inputEmail | Поле для ввода email |
| inputPassword | Name = inputPassword | Поле для ввода пароля |
| inputPasswordConfirm | Name = inputPasswordConfirm | Поле для подтверждения пароля |
| buttonRegister | Name = buttonRegister | Выполняет функцию регистрации пользователя |
| linkToLogin | Name = linkToLogin | Переход на страницу авторизации |

Панель администратора доступна только пользователям с правами администратора, представлено на рисунке 2.3. На ней отображается список всех зарегистрированных пользователей с возможностью редактирования их данных (имя, статус администратора) и удаления. Элементы панели администратора представлены в таблице 2.12.

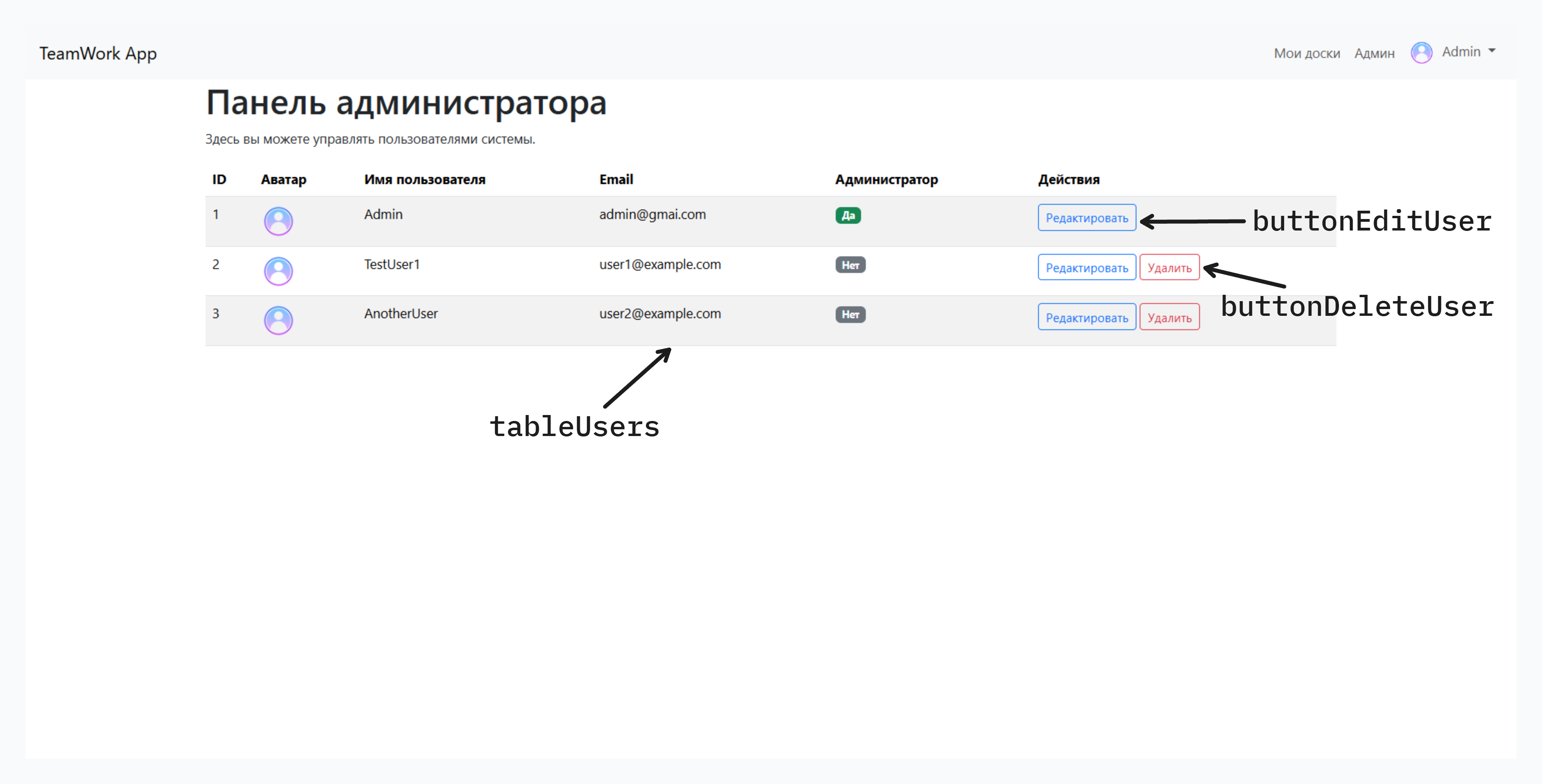


Рис.2.3. Панель администратора

Таблица 2.12 – Элементы управления панели администратора

|  |  |  |
| --- | --- | --- |
| Элементы управления | Свойства | Назначение |
| tableUsers | Name = tableUsers | Таблица для вывода списка пользователей |
| buttonEditUser | Name = buttonEditUser | Кнопка для перехода к редактированию данных пользователя |
| buttonDeleteUser | Name = buttonDeleteUser | Кнопка для удаления пользователя |

На странице моих досок пользователь видит список всех досок, где он является владельцем или участником, представлено на рисунке 2.4. Здесь можно создать новую доску, перейти к редактированию, управлению участниками или удалению доски (для владельца), а также покинуть доску (для участника). Элементы страницы моих досок представлены в таблице 2.13.

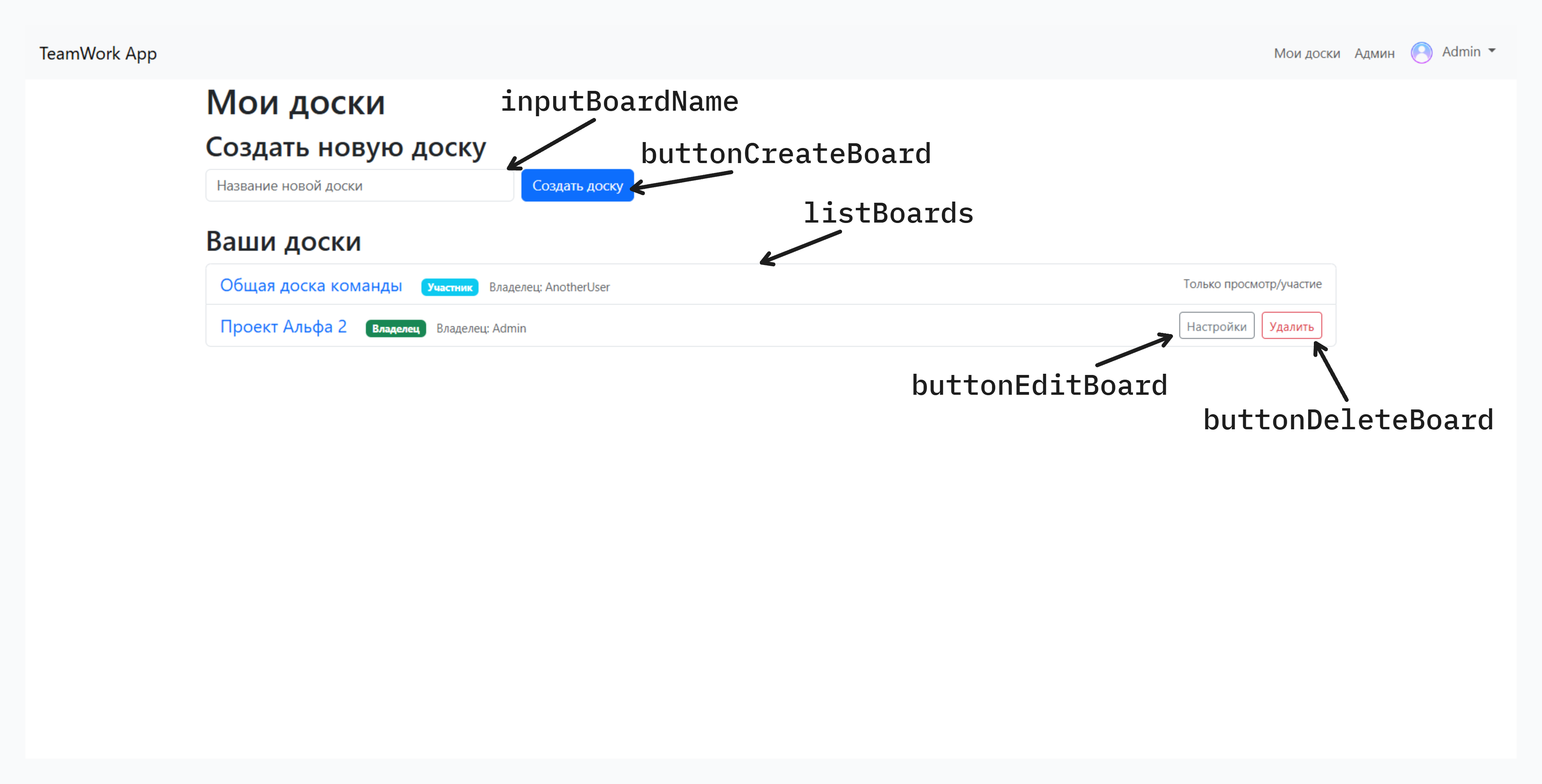


Рис.2.4. Страница моих досок

Таблица 2.13 – Элементы управления страницы моих досок

|  |  |  |
| --- | --- | --- |
| Элементы управления | Свойства | Назначение |
| listBoards | Name = listBoards | Список досок |
| buttonCreateBoard | Name = buttonCreateBoard | Кнопка для создания новой доски |
| inputBoardName | Name = inputBoardName | Поле для ввода названия новой доски |
| buttonEditBoard | Name = buttonEditBoard | Кнопка для редактирования доски |
| buttonDeleteBoard | Name = buttonDeleteBoard | Кнопка для удаления доски |

На странице профиля пользователя пользователь может просматривать свои данные (изображение пользователя, имя, email) и изменять их, представлено на рисунке 2.5. Элементы страницы профиля пользователя представлены в таблице 2.14.

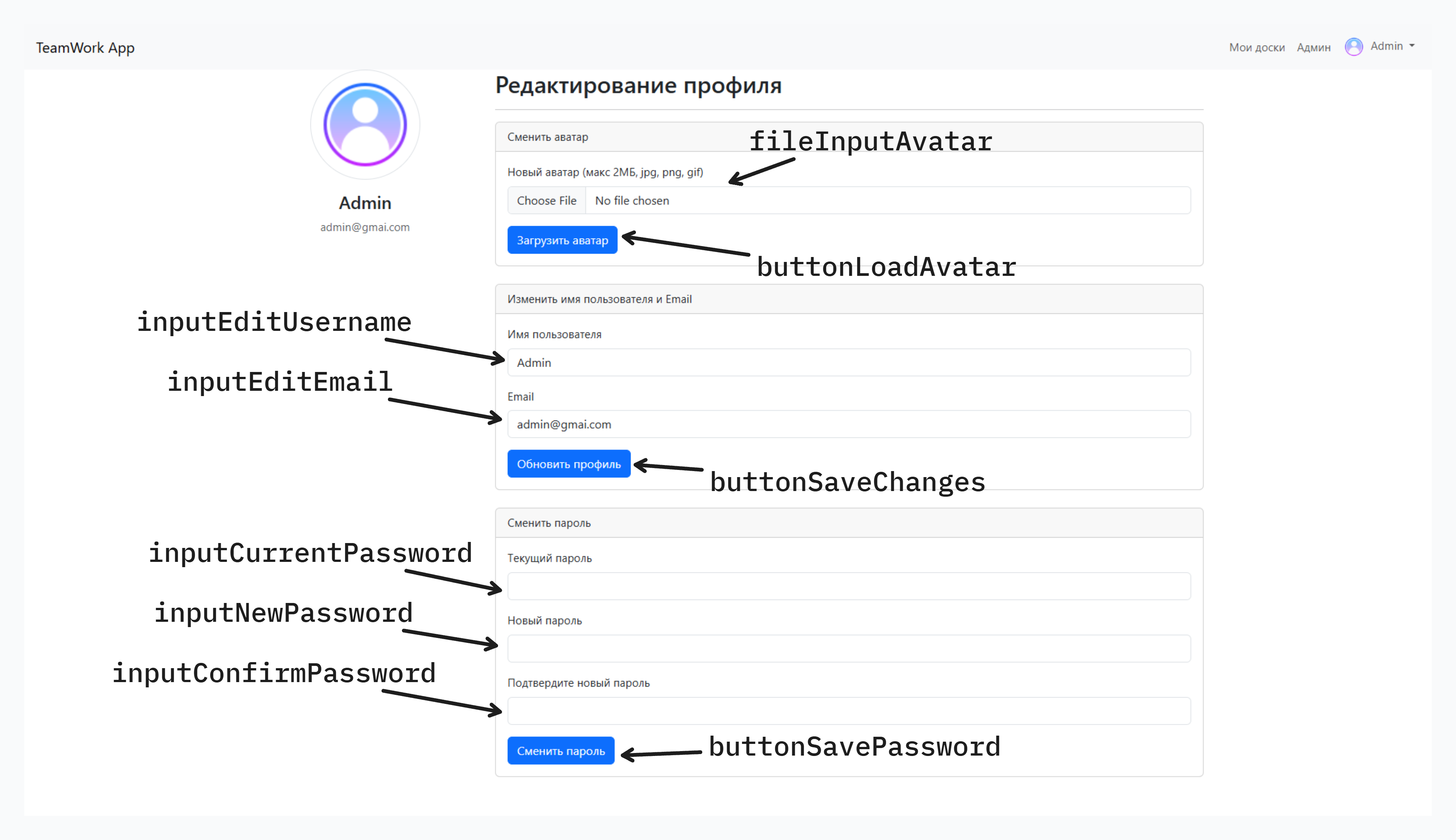
Рис.2.5. Страница профиля пользователя

Таблица 2.14 – Элементы управления страницы профиля пользователя

|  |  |  |
| --- | --- | --- |
| Элементы управления | Свойства | Назначение |
| inputEditUsername | Name = inputEditUsername | Поле для ввода нового имени пользователя |
| inputEditEmail | Name = inputEditEmail | Поле для ввода нового email |
| inputCurrentPassword | Name = inputCurrentPassword | Поле для ввода текущего пароля |
| inputNewPassword | Name = inputNewPassword | Поле для ввода нового пароля |
| inputConfirmPassword | Name = inputConfirmPassword | Поле для ввода повтора нового пароля |
| fileInputAvatar | Name = fileInputAvatar | Поле выбора изображения пользователя |
| buttonSaveChanges | Name = buttonSaveChanges | Кнопка для сохранения изменений |
| buttonLoadAvatar | Name = buttonLoadAvatar | Кнопка загрузки изображения пользователя |
| buttonSavePassword | Name = buttonSavePassword | Кнопка для сохранения нового пароля |

На странице доски отображается выбранная доска с ее колонками и карточками, представлено на рисунке 2.6. Пользователи могут создавать, редактировать и удалять колонки и карточки, перетаскивать карточки, а также использовать поиск, фильтрацию и сортировку. Элементы страницы доски представлены в таблице 2.15.

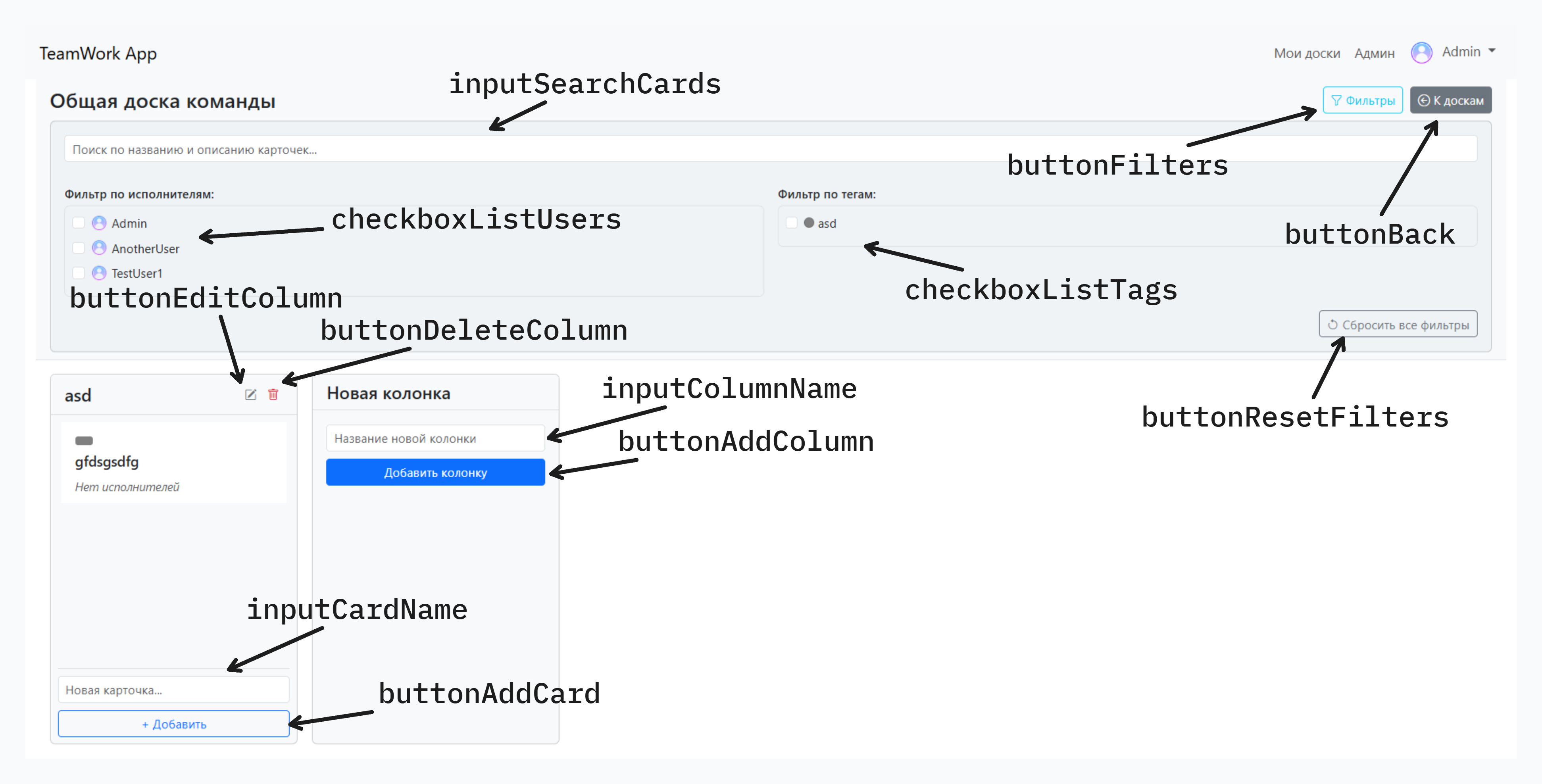


Рис.2.6. Страница доски

Таблица 2.15 – Элементы управления страницы доски

|  |  |  |
| --- | --- | --- |
| Элементы управления | Свойства | Назначение |
| inputSearchCards | Name = inputSearchCards | Поле поиска карточек |
| checkboxListAssignees | Name = checkboxListAssignees | Множественный выбор фильтрации по ответственным за задачу |
| buttonEditColumn | Name = buttonEditColumn | Кнопка для изменения названия колонки |
| buttonDeleteColumn | Name = buttonDeleteColumn | Кнопка для удаления колонки |
| inputColumnName | Name = inputColumnName | Поле для ввода названия колонки |
| buttonAddColumn | Name = buttonAddColumn | Кнопка для добавления колонки |
| inputCardName | Name = inputCardName | Поле для ввода названия карточки |
| buttonAddCard | Name = buttonAddCard | Кнопка для добавления карточки |
| buttonFilters | Name = buttonFilters | Кнопка для открытия и закрытия фильтров |

*Продолжение таблицы 2.15*

|  |  |  |
| --- | --- | --- |
| Элементы управления | Свойства | Назначение |
| checkboxListTags | Name = checkboxListTags | Множественный выбор фильтрации по тегам |
| buttonBack | Name = buttonBack | Кнопка для возврата на страницу досок |
| buttonResetFilters | Name = buttonResetFilters | Кнопка сброса фильтров |

Модальное окно карточки открывается по клику на карточку на странице доски, представлено на рисунке 2.7. В нем можно просматривать и редактировать детали карточки (заголовок, описание), управлять ответственными за задачу и тегами карточки, добавлять и просматривать комментарии, а также управлять тегами всей доски. Элементы модального окна карточки представлены в таблице 2.16.

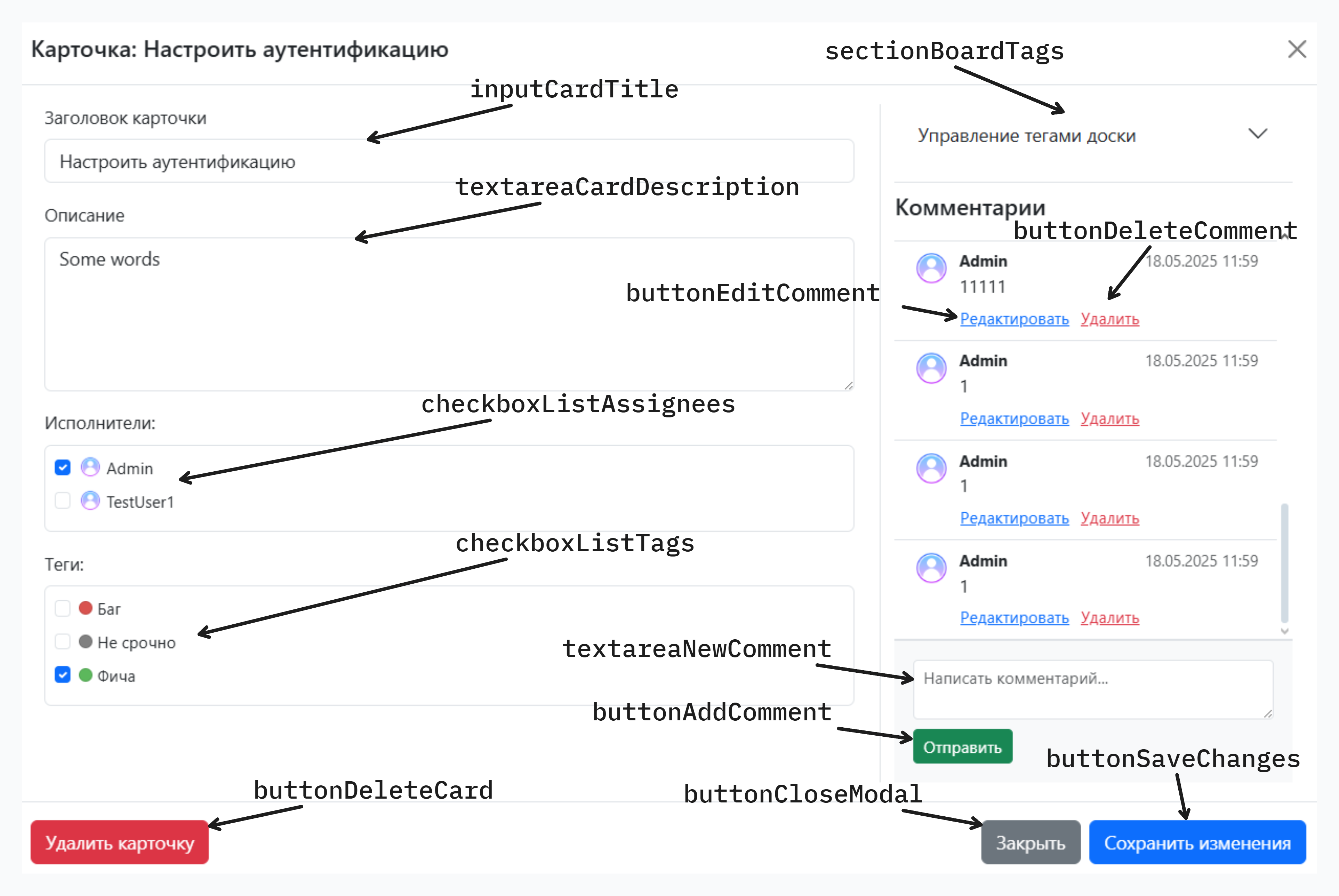


Рис.2.7. Модальное окно карточки

Таблица 2.11 – Элементы управления страницы модального окна карточки

|  |  |  |
| --- | --- | --- |
| Элементы управления | Свойства | Назначение |
| inputCardTitle | Name = inputCardTitle | Поле редактирования названия карточки |

*Продолжение таблицы 2.11*

|  |  |  |
| --- | --- | --- |
| Элементы управления | Свойства | Назначение |
| textareaCardDescription | Name = textareaCardDescription | Поле редактирования описания карточки |
| buttonEditComment | Name = buttonEditComment | Кнопка редактирования комментария |
| checkboxListAssignees | Name = checkboxListAssignees | Множественный выбор ответственных за задачу |
| checkboxListTags | Name = checkboxListTags | Множественный выбор тегов |
| buttonDeleteCard | Name = buttonDeleteCard | Кнопка удаления карточки |
| sectionBoardTags | Name = sectionBoardTags | Секция для управления тегами доски |
| buttonDeleteComment | Name = buttonDeleteComment | Поле ввода описания карточки |
| textareaNewComment | Name = textareaNewComment | Поле ввода нового комментария |
| buttonAddComment | Name = buttonAddComment | Кнопка добавления комментария |
| buttonCloseModal | Name = buttonCloseModal | Кнопка выхода из карточки |
| buttonSaveChanges | Name = buttonSaveChanges | Кнопка сохранения изменений |

# 2.5. Описание физической структуры программы

Программа состоит из набора файлов и каталогов, организованных для управления различными аспектами веб-приложения. Основная структура проекта включает следующие ключевые компоненты:

Корневой каталог содержит следующие файлы:

1. Run.py – запуск приложения и CLI-команды для управления БД.
2. Run\_app.bat – BAT-файл для быстрого запуска приложения.
3. Reinit\_db.bat – BAT-файл для переинициализации базы данных.
4. Gitignore – файл Git для исключения отслеживания определенных файлов/папок.

Каталог app содержит следующие файлы:

1. Models.py – модели базы данных SQLAlchemy (User, Board, Column, Card, Tag, Comment и др.).
2. \_\_Init\_\_.py – инициализация Flask приложения и его расширений.
3. Routes.py – маршруты приложения и основная логика обработки запросов, включая AJAX.
4. Forms.py – формы Flask-WTF для ввода данных и валидации.

Подкаталог templates хранит следующие файлы:

1. Base.html – базовый HTML-шаблон для всех страниц.
2. Index.html – шаблон главной страницы.
3. Login.html – шаблон страницы авторизации.
4. Register.html – шаблон страницы регистрации.
5. Dashboard.html – шаблон страницы мох досок.
6. Edit\_board.html – шаблон для редактирования доски и управления участниками.
7. Board.html – шаблон страницы отображения доски с колонками и карточками.
8. \_Card\_modal.html – частичный шаблон модального окна карточки.
9. Edit\_column.html – шаблон модального окна для редактирования колонки.
10. Profile.html – шаблон страницы профиля пользователя.
11. Admin\_dashboard.html – шаблон панели администратора (список пользователей).
12. Admin\_edit\_user.html – шаблон для редактирования пользователя администратором.

Подкаталог static содержит следующие файлы:

1. Style.css – пользовательские CSS стили.
2. Board\_logic.js – клиентский JavaScript для страницы доски (AJAX, SortableJS, фильтры).
3. Default\_avatar.png – изображение пользователя по умолчанию.

# 3. ТЕСТИРОВАНИЕ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определенным образом.

Проверка функциональности – проверка соответствия программного обеспечения требованиям, заявленным в спецификации. Может проводиться как полное тестирование заявленной функциональность, так и проверка только базовой функциональности.

Системное тестирование – высокоуровневая проверка функционала всей программы или системы в целом.

Тестирование производительности – тестирование, которое проводится с целью определения, как быстро работает система или её часть под определённой нагрузкой.

В результате проверки программного средства на работоспособность, были проведены следующие тестовые испытания:

1. Тест №1. Регистрация нового пользователя.
2. Тест №2. Авторизация существующего пользователя.
3. Тест №3. Создание новой доски.
4. Тест №4. Создание новой колонки на доске.
5. Тест №5. Создание новой карточки в колонке.
6. Тест №6. Редактирование данных профиля пользователя.
7. Тест №7. Добавление комментария к карточке через модальное окно.
8. Тест №8. Назначение ответственных за задачу на карточку через модальное окно.
9. Тест №9. Фильтрация карточек по тегу на странице доски.
10. Тест №10. Удаление пользователя администратором.

Тест №1 показал, что после ввода данных на странице регистрации (имя пользователя, email, пароль) и нажатия на кнопку «Зарегистрироваться» происходит успешное создание нового пользователя, и перенаправление на страницу авторизации, представлено на рисунках 3.1., 3.2.

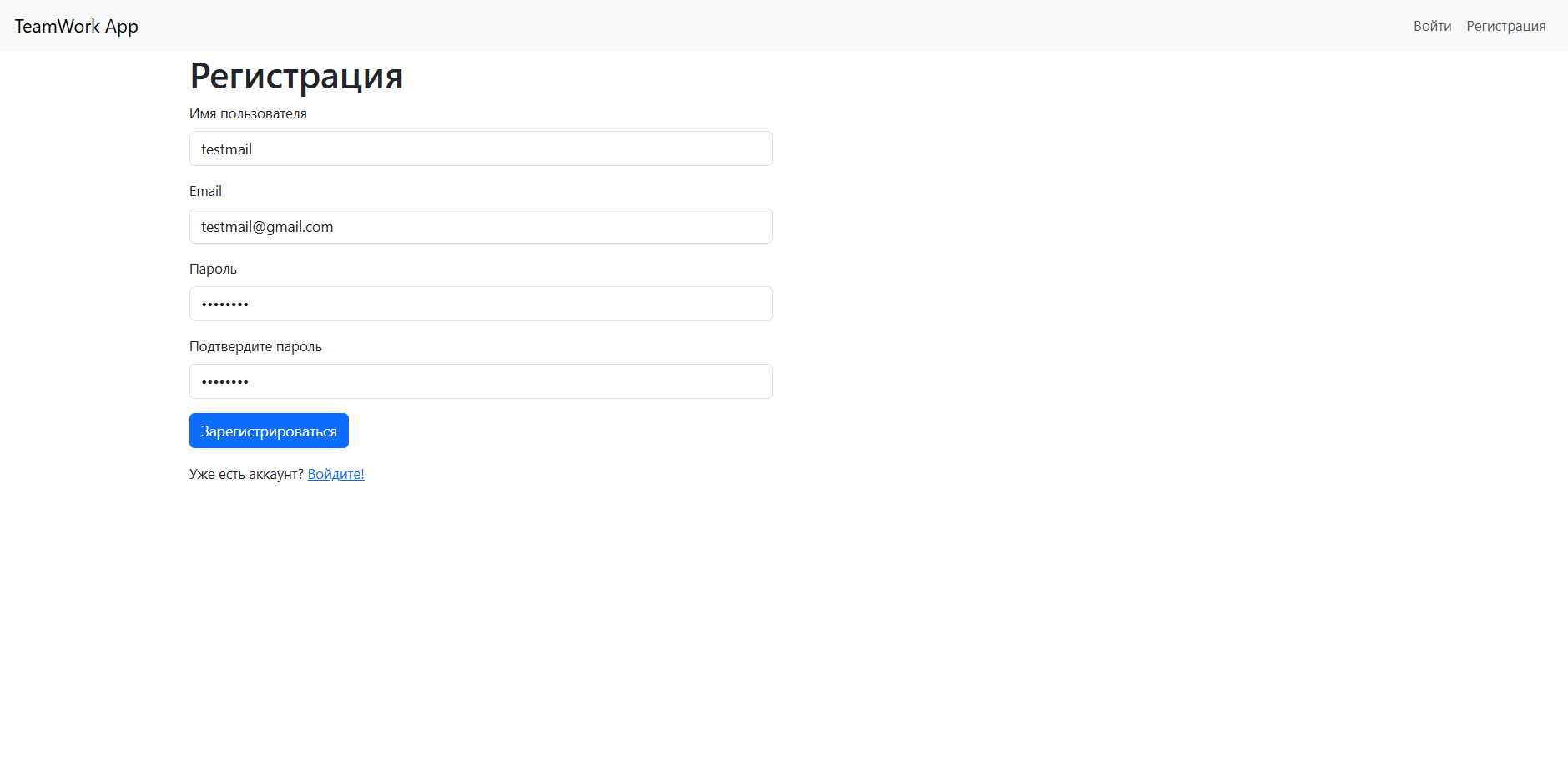


Рис. 3.1. Заполненная форма регистрации нового пользователя

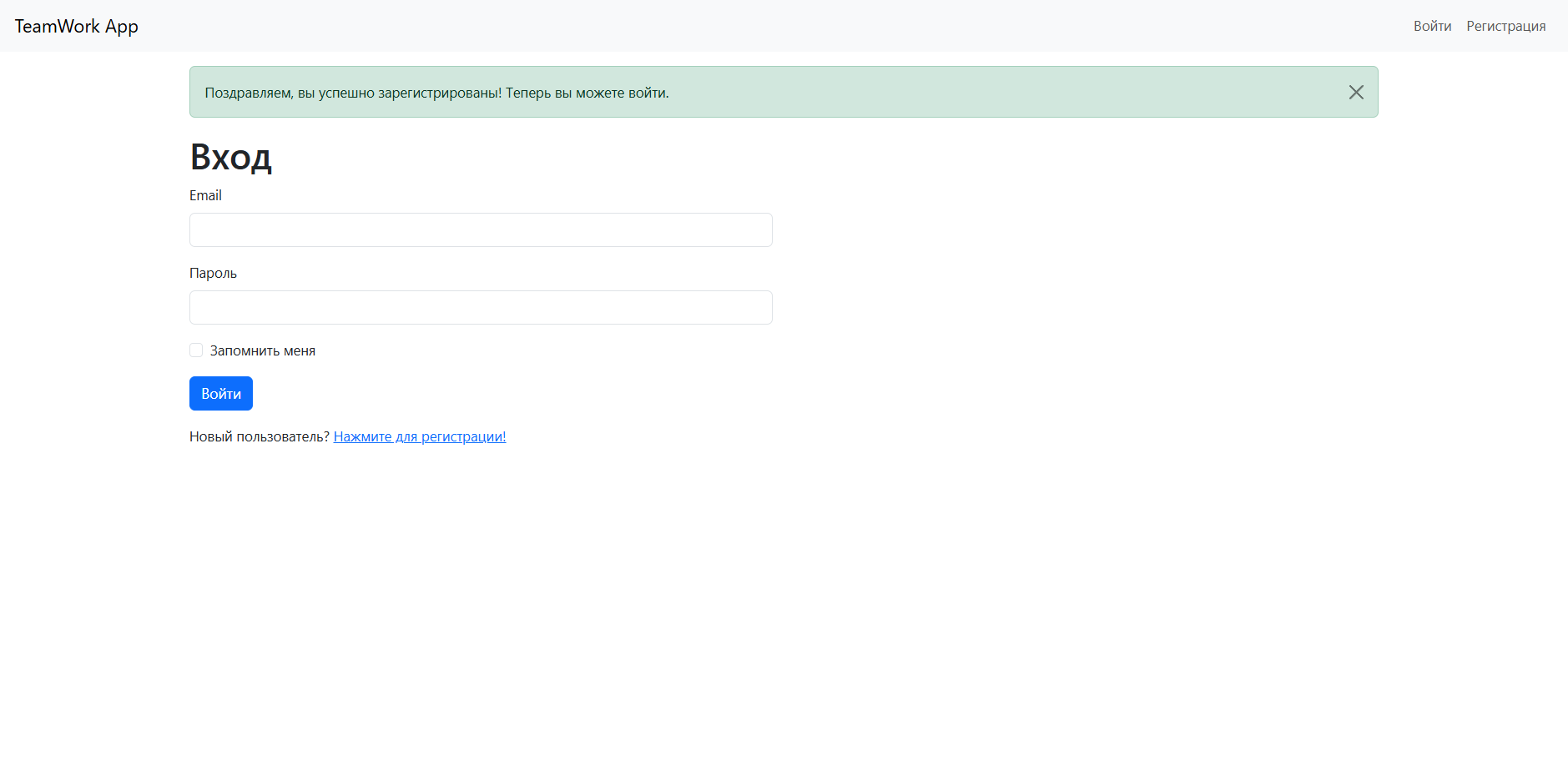


Рис. 3.2. Страница авторизации с сообщением об успешной регистрации

Тест №2 показал, что после ввода корректных email и пароля существующего пользователя на странице авторизации и нажатия на кнопку «Войти» осуществляется переход на страницу мох досок, представлено на рисунках 3.3., 3.4.

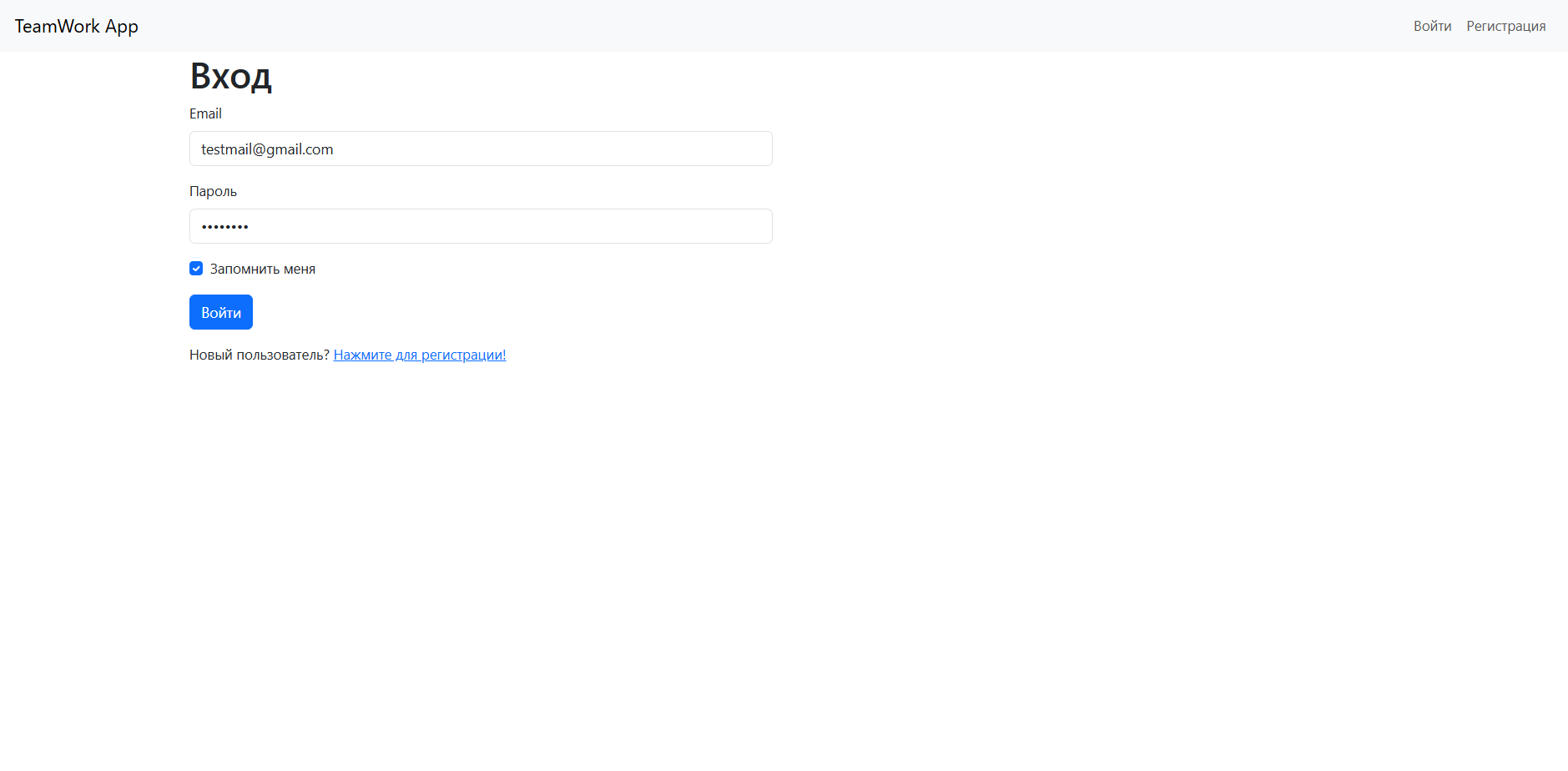


Рис. 3.3. Заполненная форма авторизации существующего пользователя

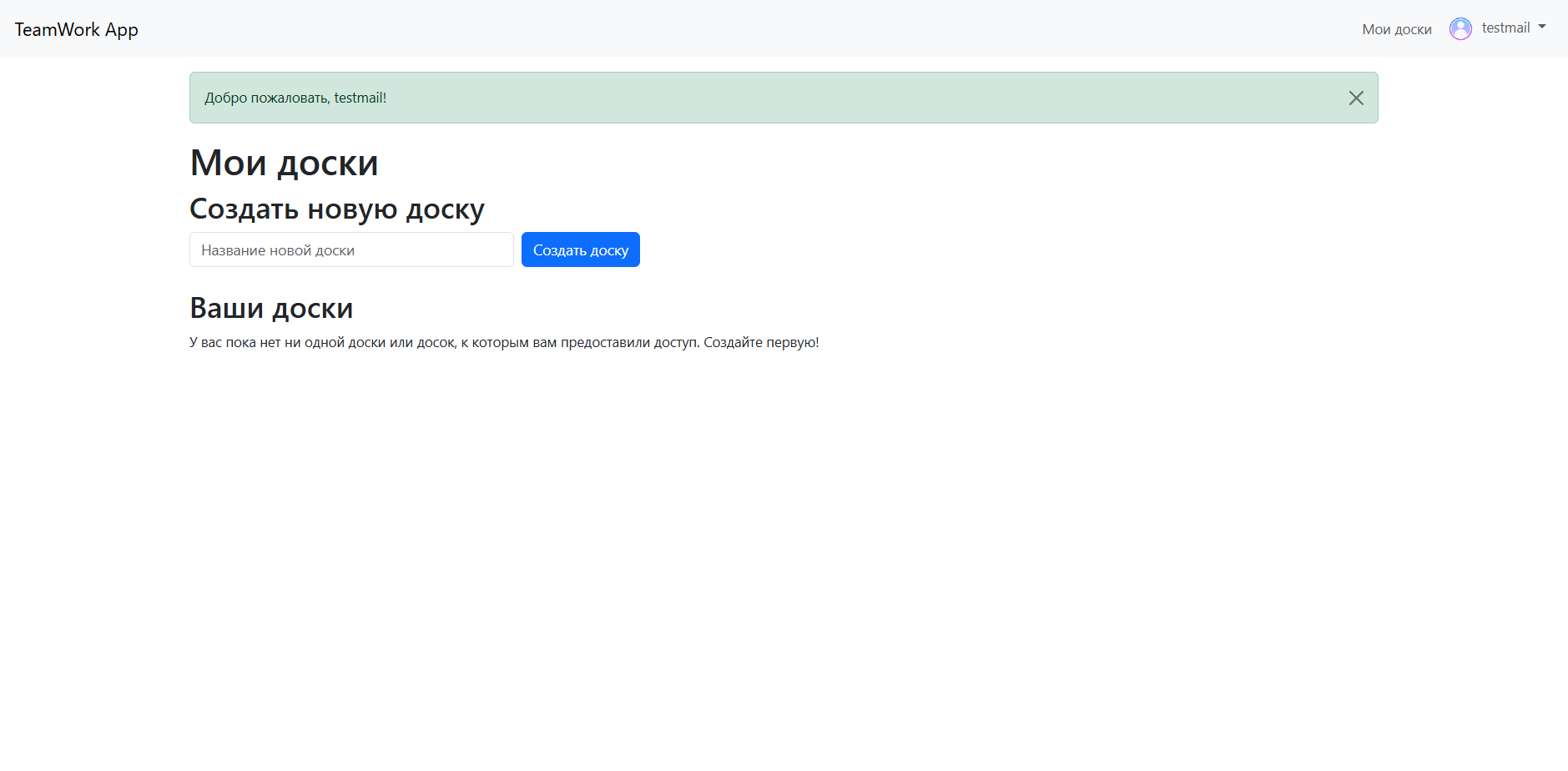


Рис. 3.4. Страница моих досок после успешной авторизации

Тест №3 показал, что авторизованный пользователь может успешно создать новую доску. После ввода названия доски и нажатия кнопки «Создать доску», новая доска появляется в списке на странице моих досок, представлено на рисунках 3.5., 3.6.

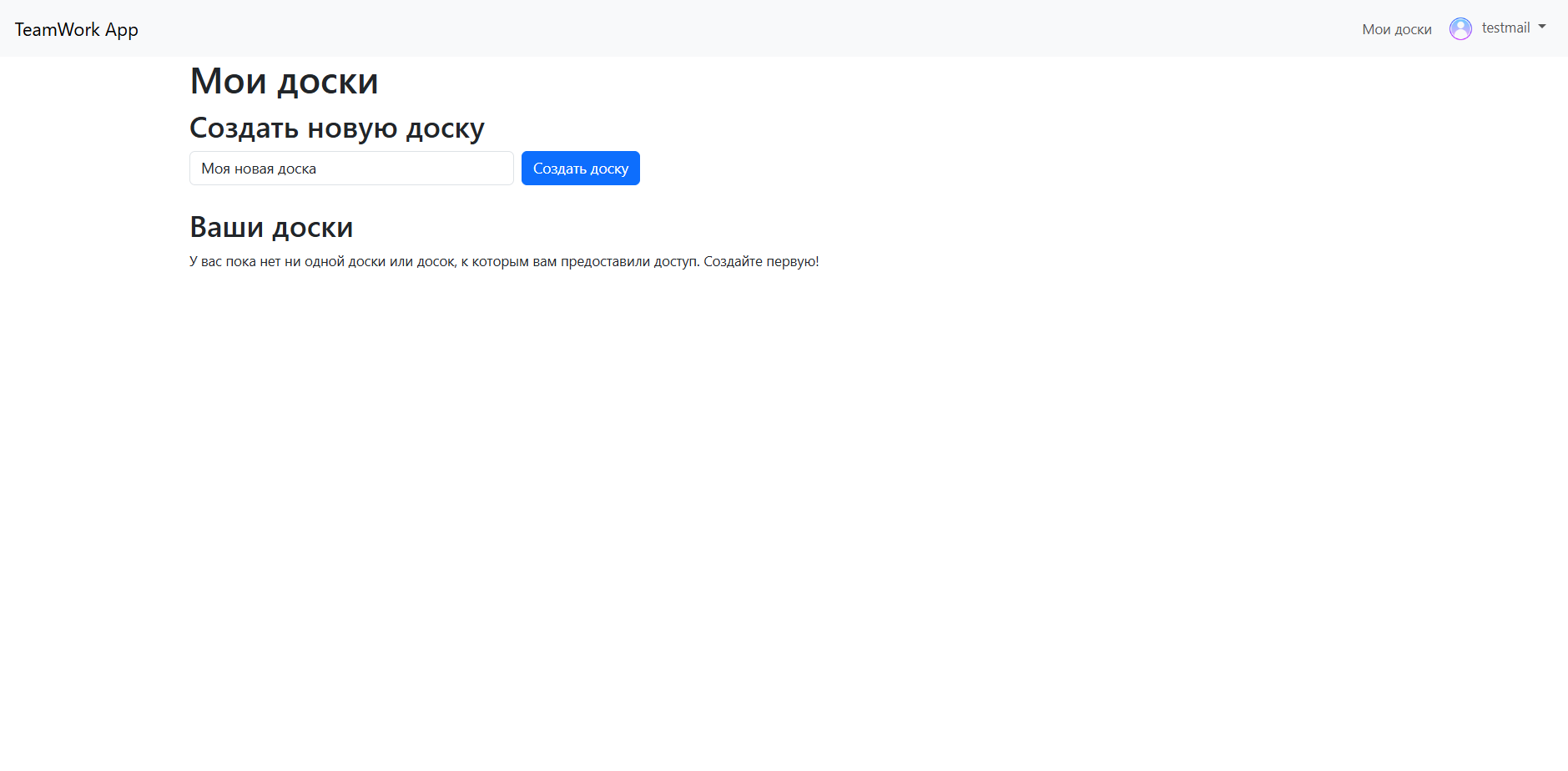


Рис. 3.5. Страница моих досок с введённым названием новой доски

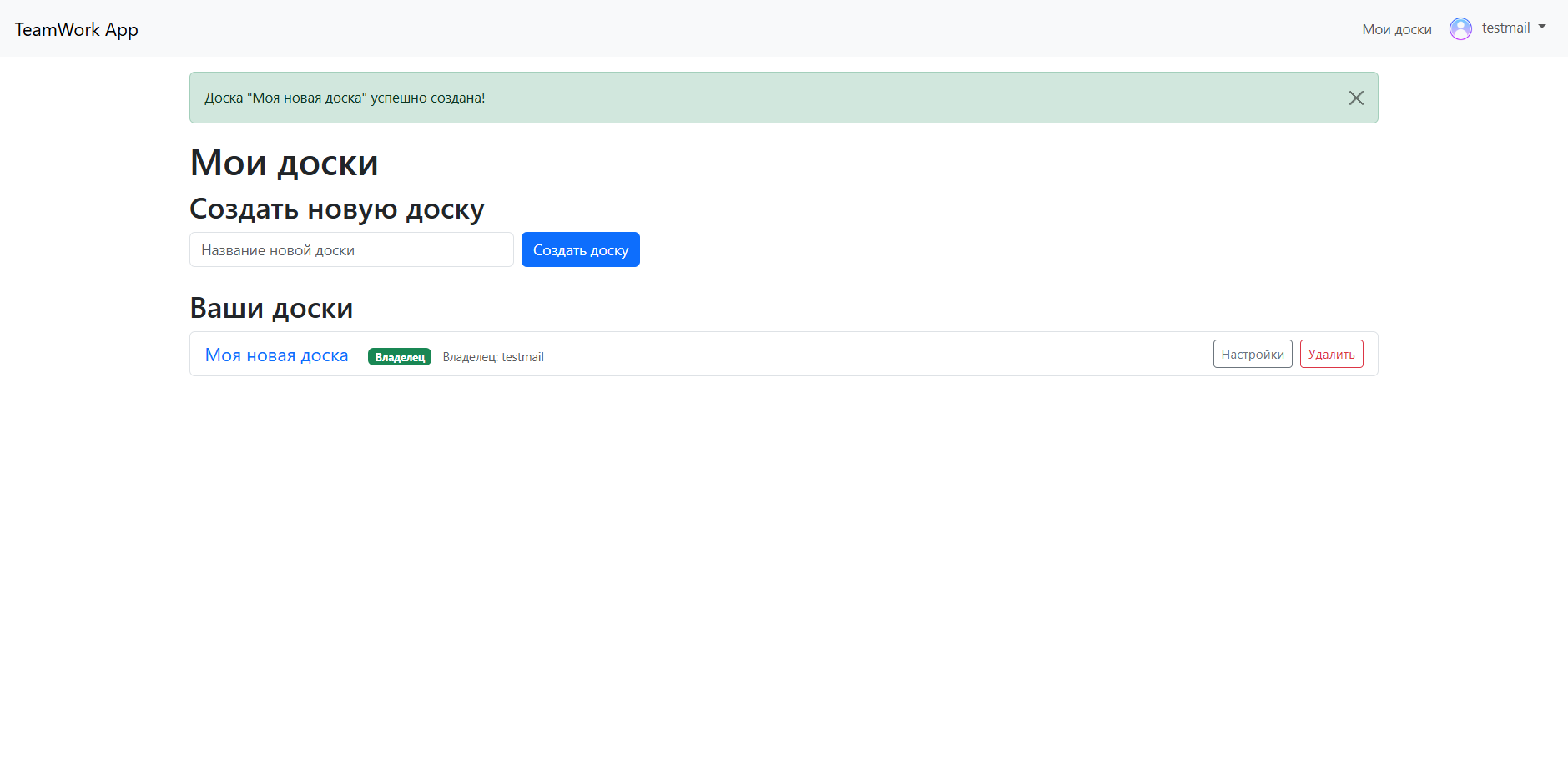


Рис. 3.6. Страница моих досок с отображением созданной доски в списке

Тест №4 показал, что на странице доски пользователь (владелец или участник с правами) может создать новую колонку. После ввода названия колонки и подтверждения, новая колонка отображается на доске, представлено на рисунках 3.7., 3.8.

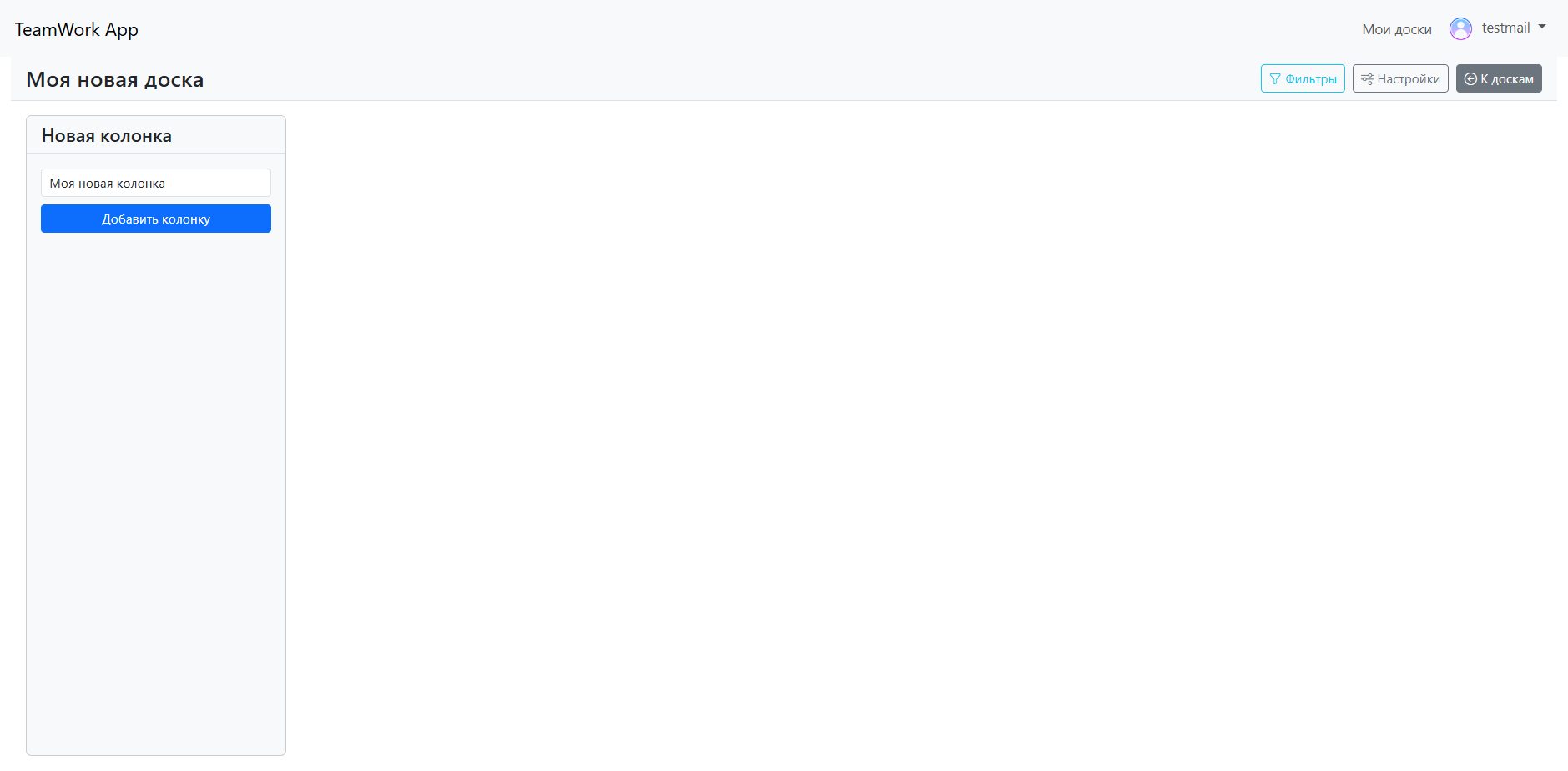


Рис. 3.7. Страница доски с введенным названием

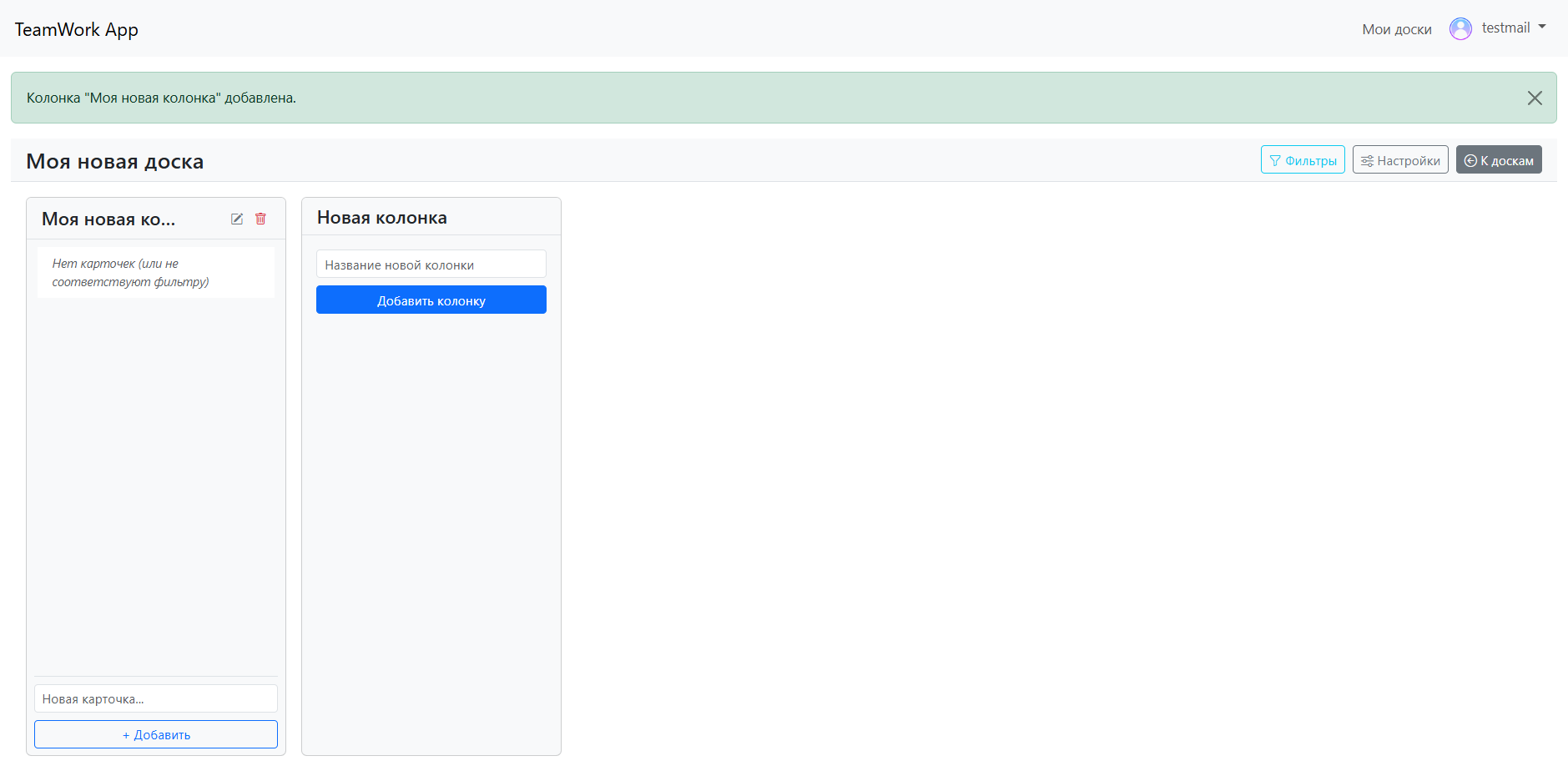


Рис. 3.8. Страница доски с отображением созданной колонки

Тест №5 показал, что пользователь может добавить новую карточку в существующую колонку. После ввода заголовка карточки и подтверждения, карточка появляется в указанной колонке, представлено на рисунках 3.9., 3.10.

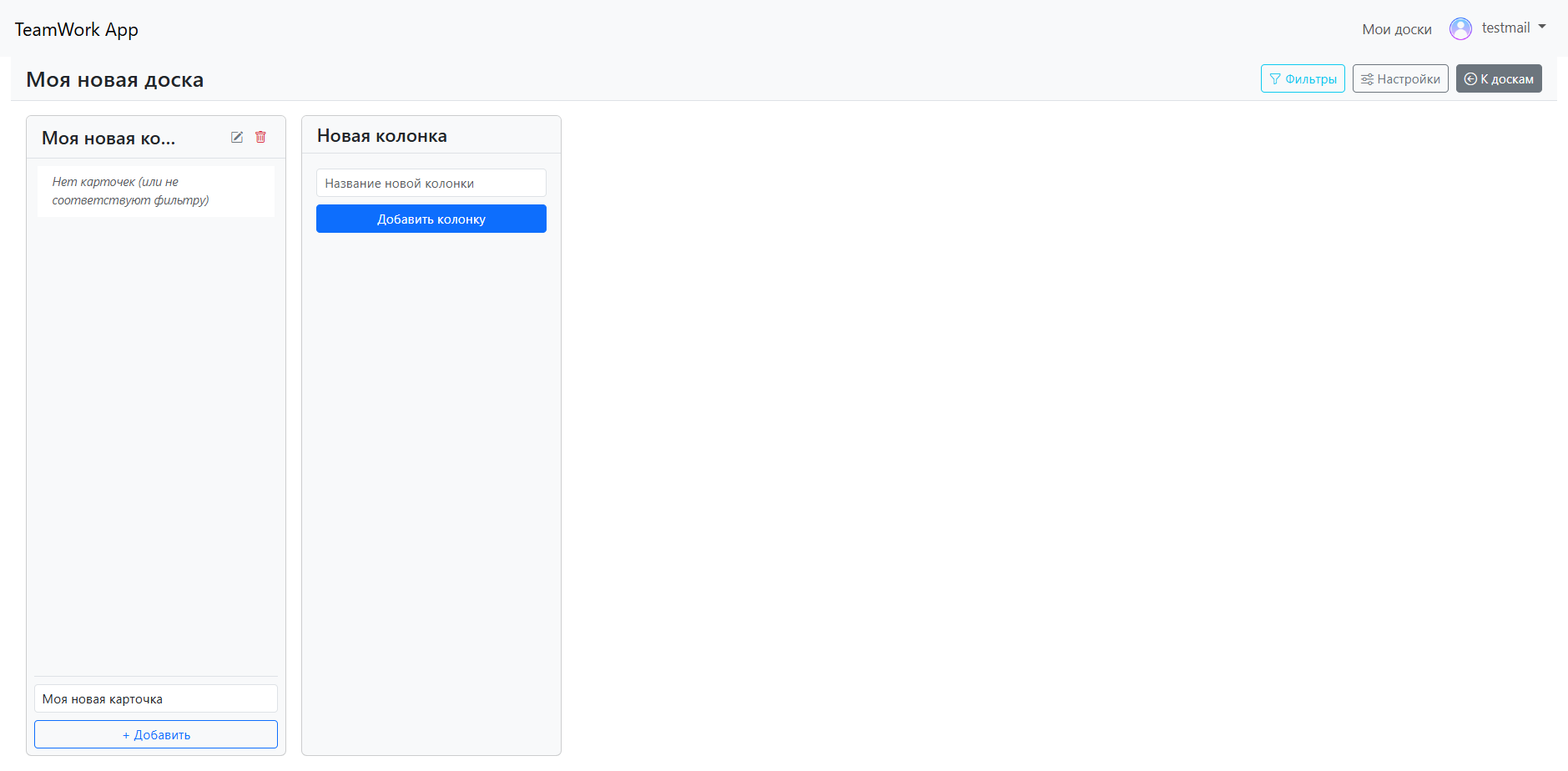


Рис. 3.9. Страница доски с введенным заголовком карточки

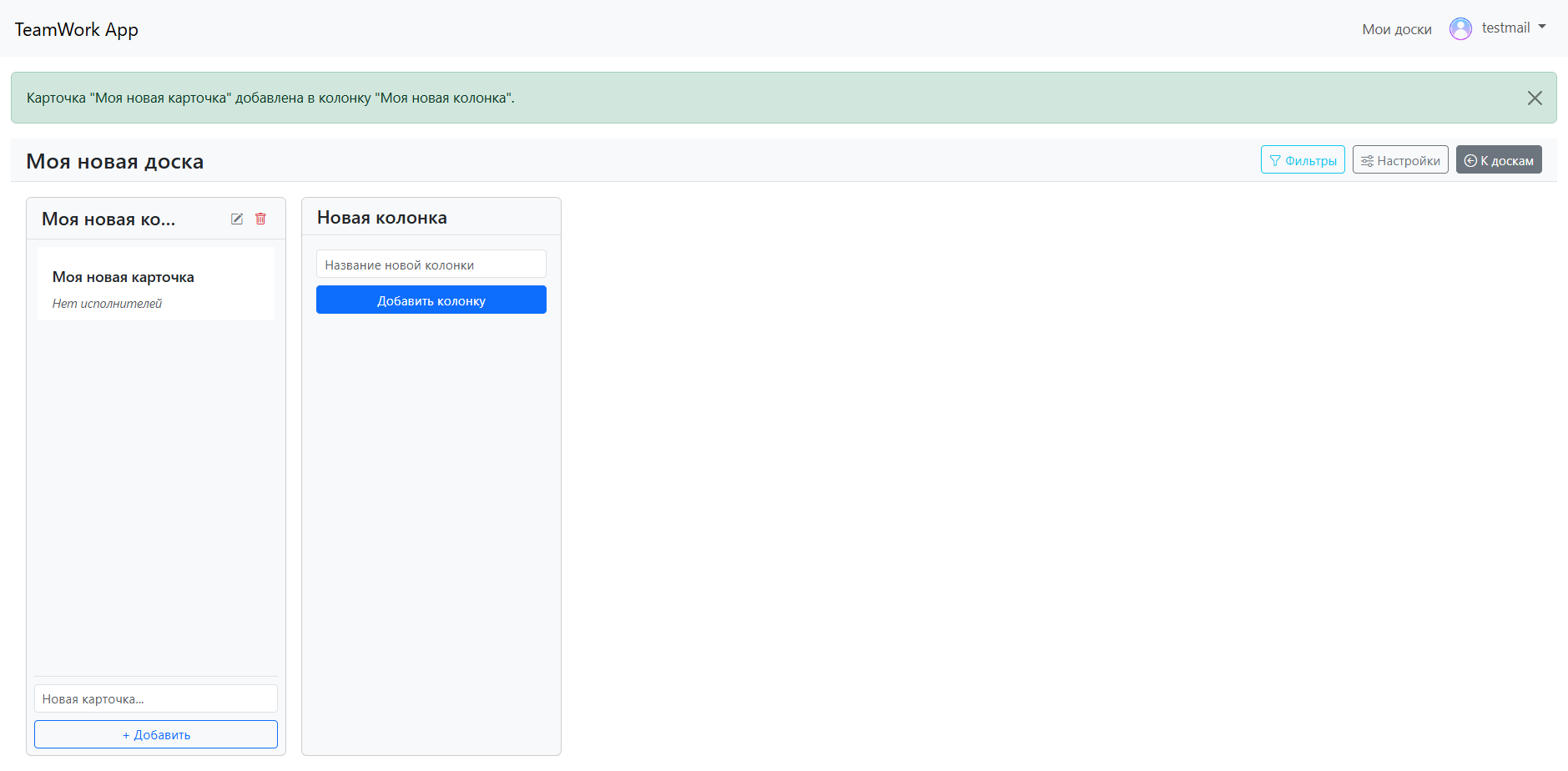


Рис. 3.10. Страница доски с отображением созданной карточки в колонке

Тест №6 показал, что пользователь может успешно редактировать данные своего профиля, включая изменение имени пользователя и загрузку нового изображения пользователя. После сохранения изменений, обновленные данные отображаются на странице профиля, представлено на рисунках 3.11., 3.12.

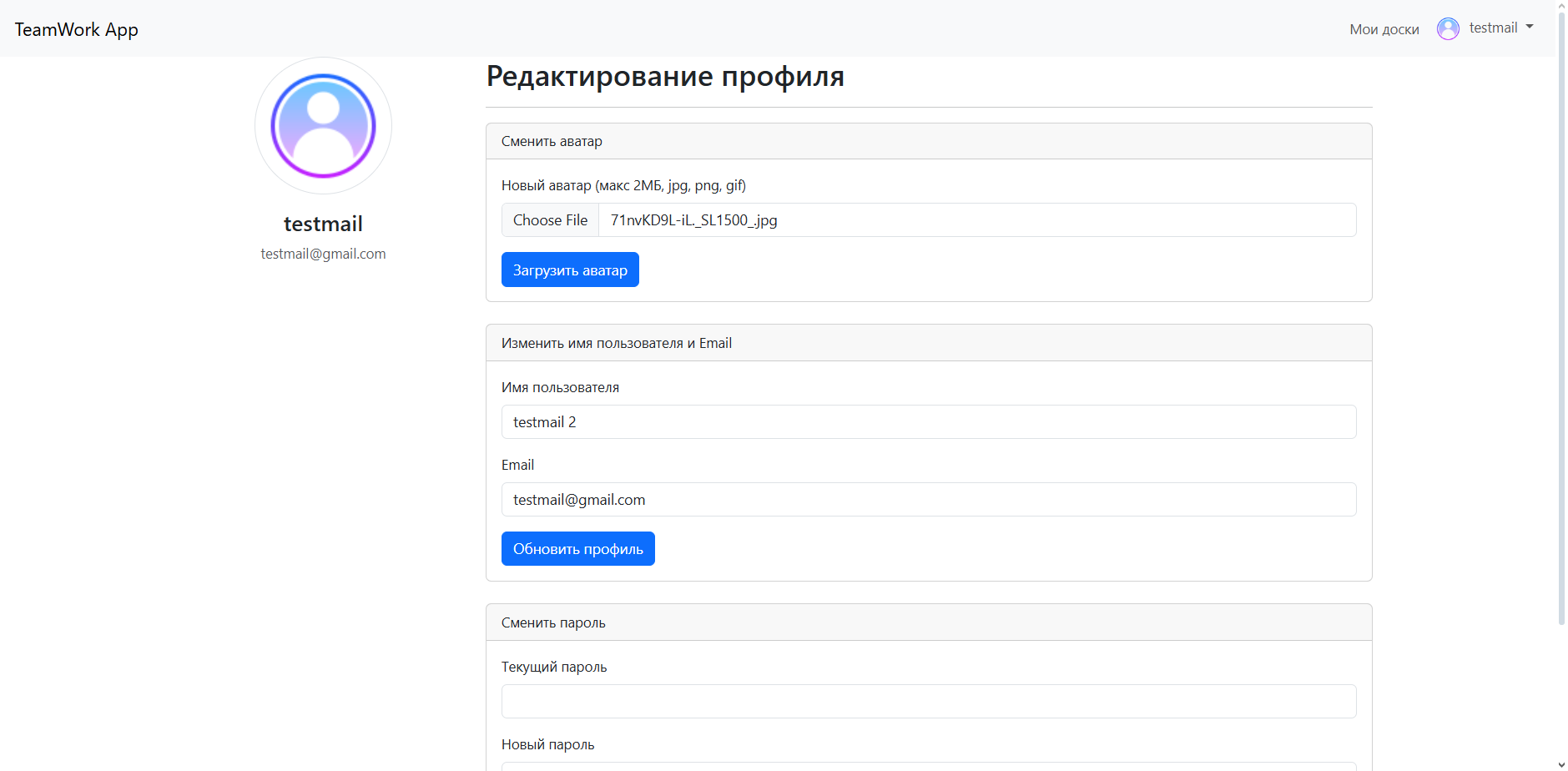


Рис. 3.11. Страница профиля с введенным новым именем пользователя и выбранным файлом нового изображения пользователя

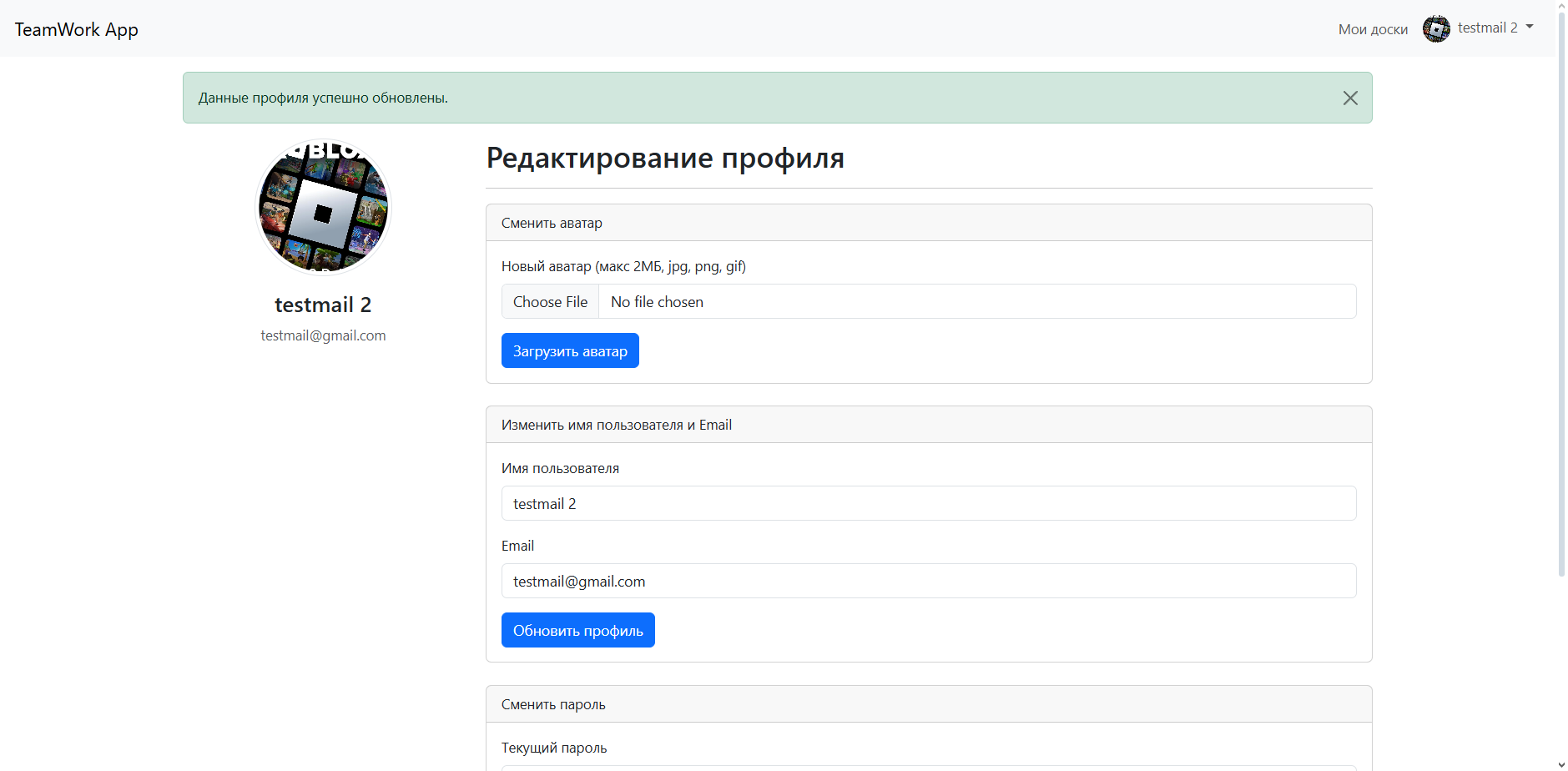


Рис. 3.12. Страница профиля с отображением обновленного имени пользователя и нового изображения пользователя

Тест №7 показал, что пользователь может добавить комментарий к карточке через модальное окно. После ввода текста комментария и отправки, комментарий отображается в списке комментариев карточки, представлено на рисунках 3.13., 3.14.

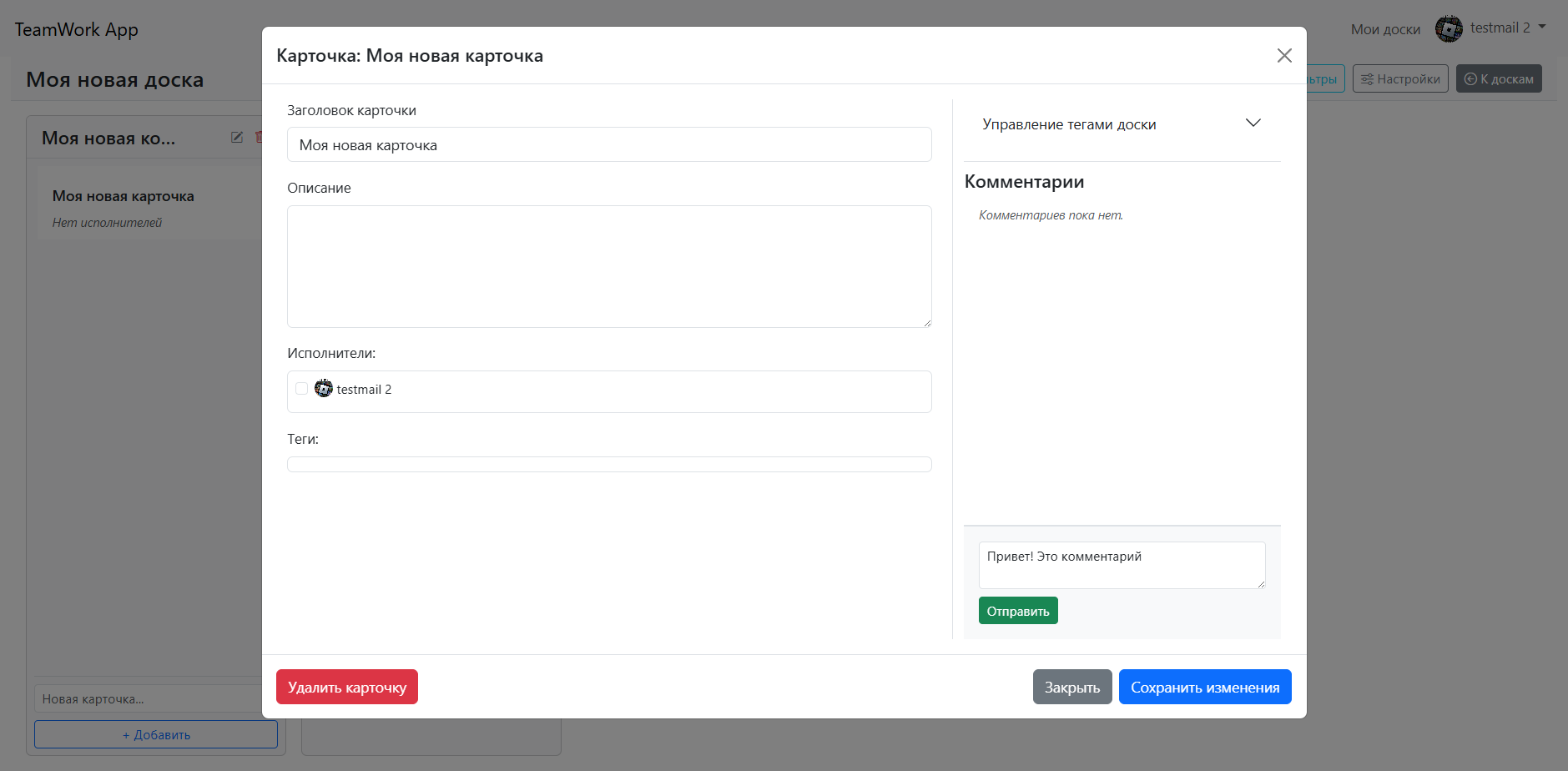


Рис. 3.13. Модальное окно карточки с введенным текстом

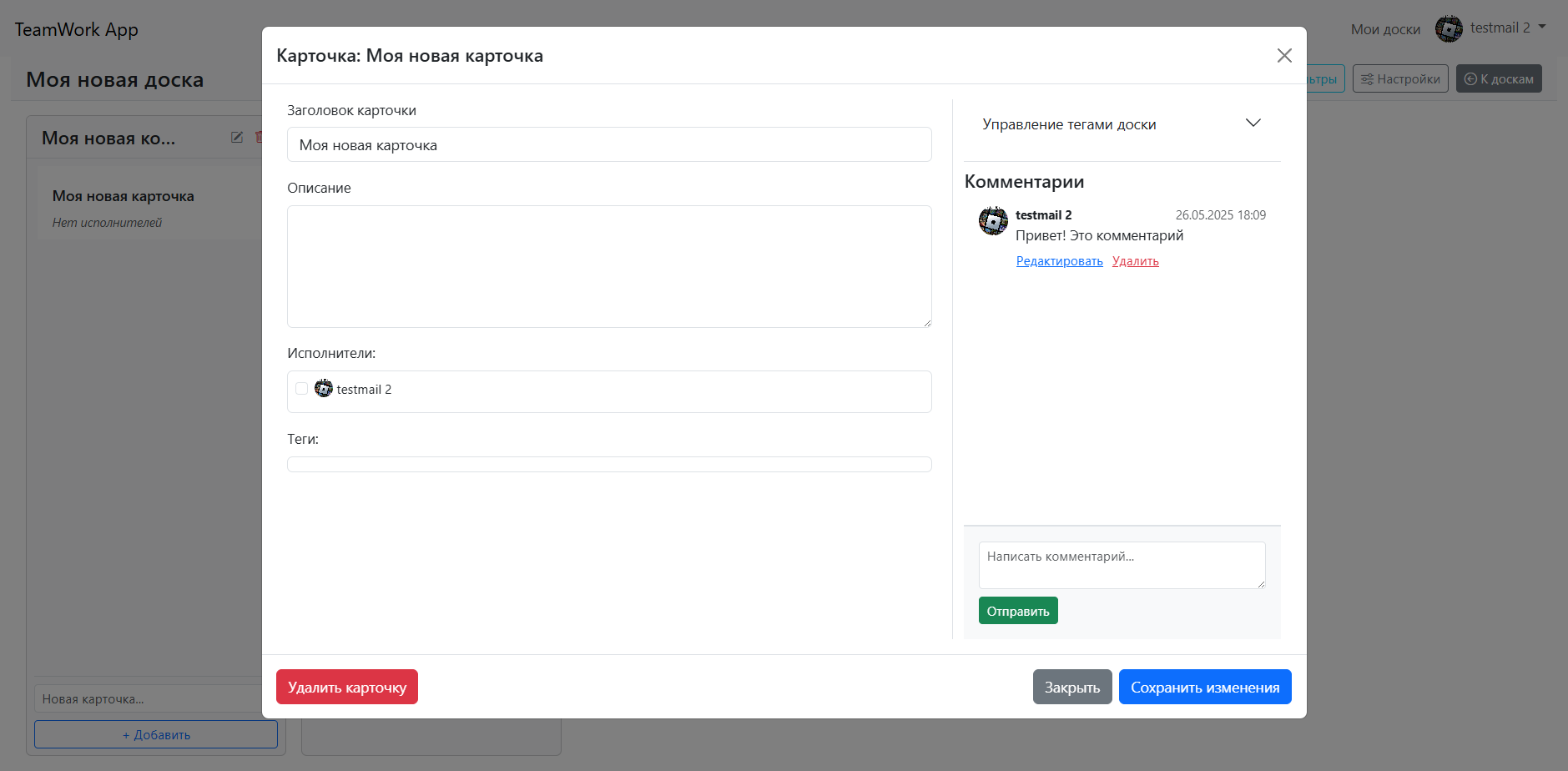


Рис. 3.14. Модальное окно карточки с отображением добавленного комментария

Тест №8 показал, что пользователь может назначить ответственного за задачу на карточку через модальное окно. После выбора пользователя из списка участников доски и сохранения, изображение пользователя ответственного за задачу отображается на карточке представлено на рисунках 3.15., 3.16.

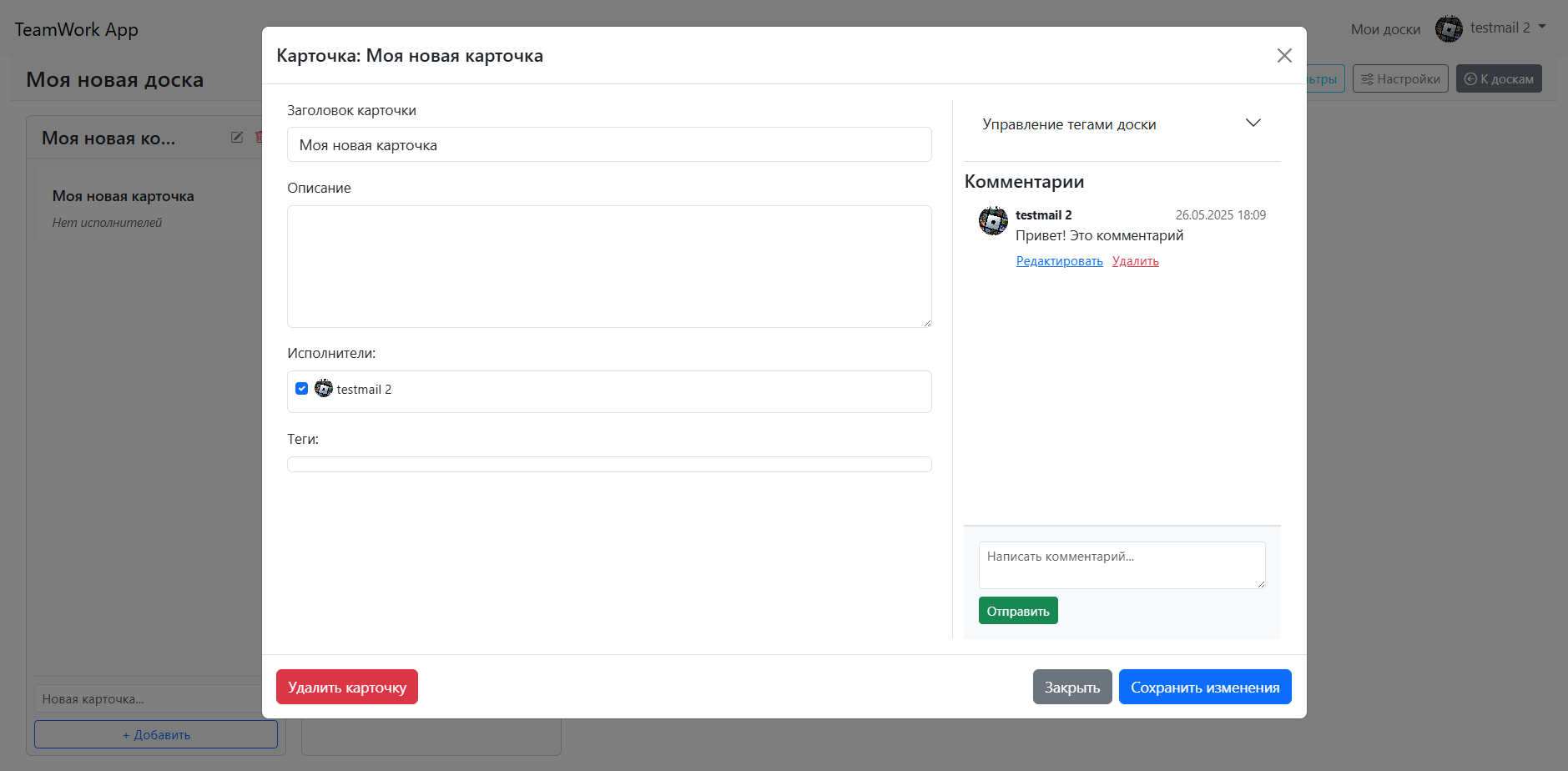


Рис. 3.15. Модальное окно карточки с выбранным ответственным за задачу

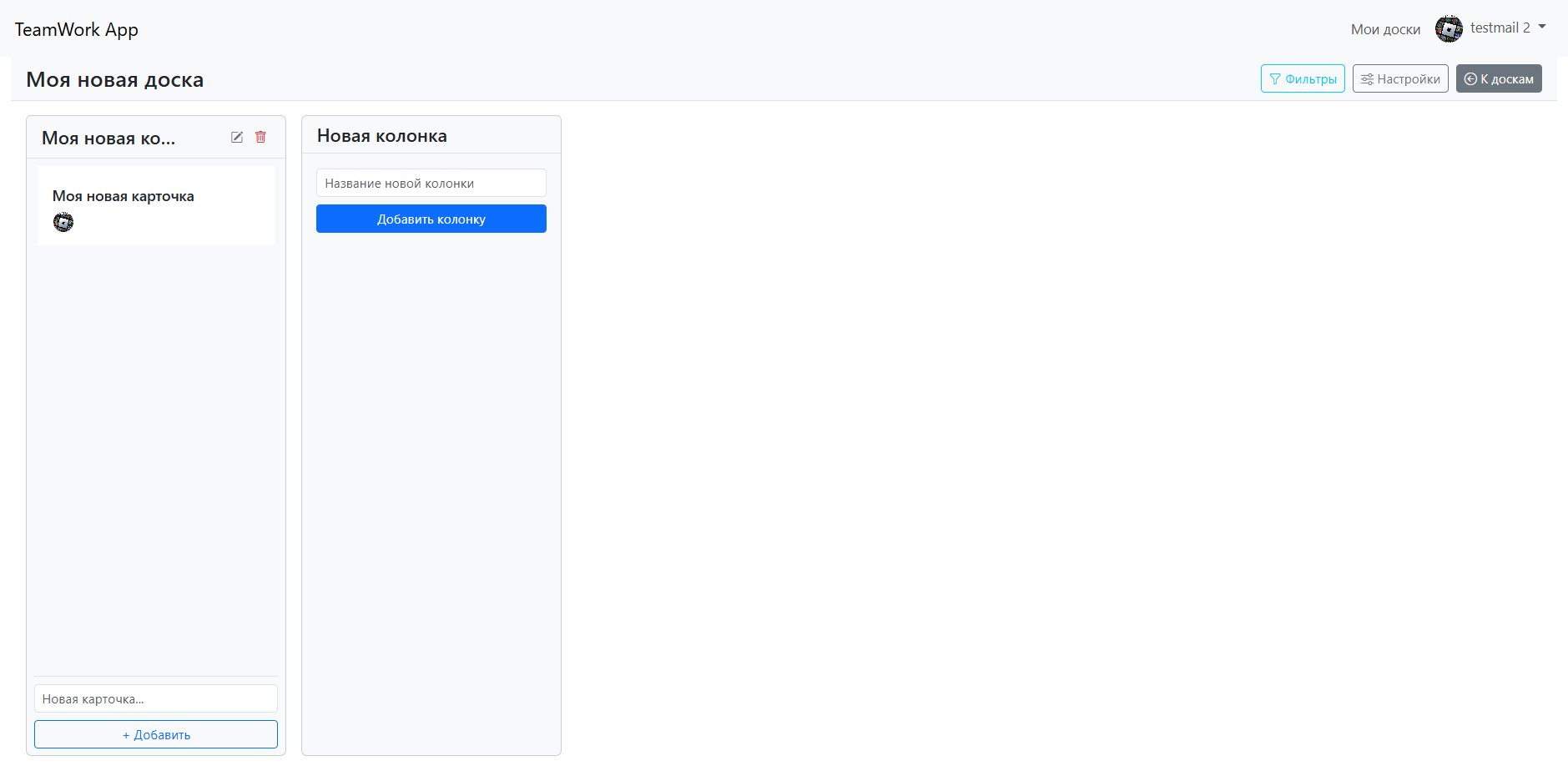


Рис. 3.16. Страница доски с назначенным ответственным за задачу

Тест №9 показал, что карточки на странице доски можно фильтровать по тегу. После выбора тега в фильтре, на доске остаются только те карточки, которым присвоен данный тег, представлено на рисунках 3.17 и 3.18.

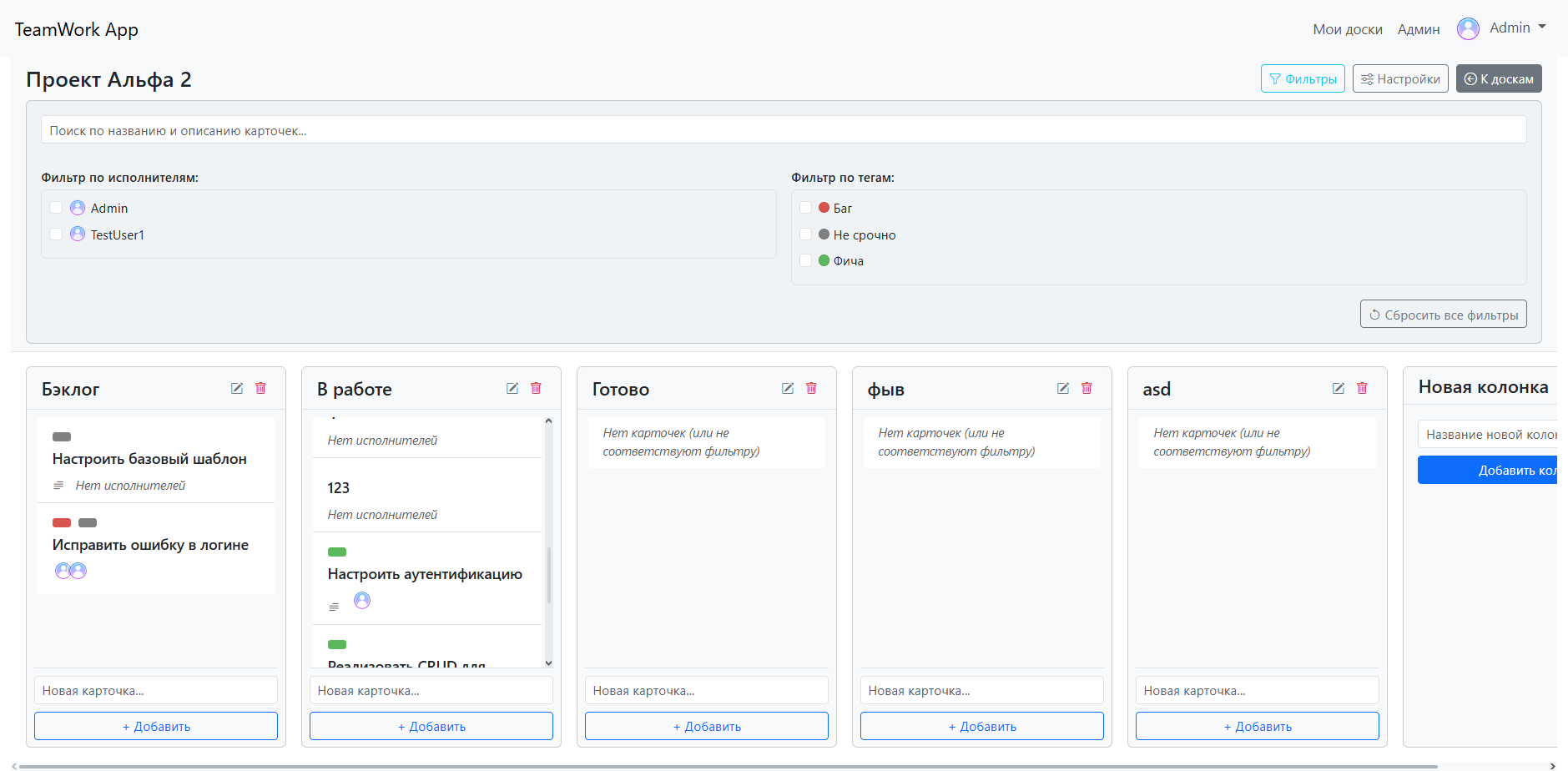


Рис. 3.17. Страница доски с карточками и открытым фильтром по тегам

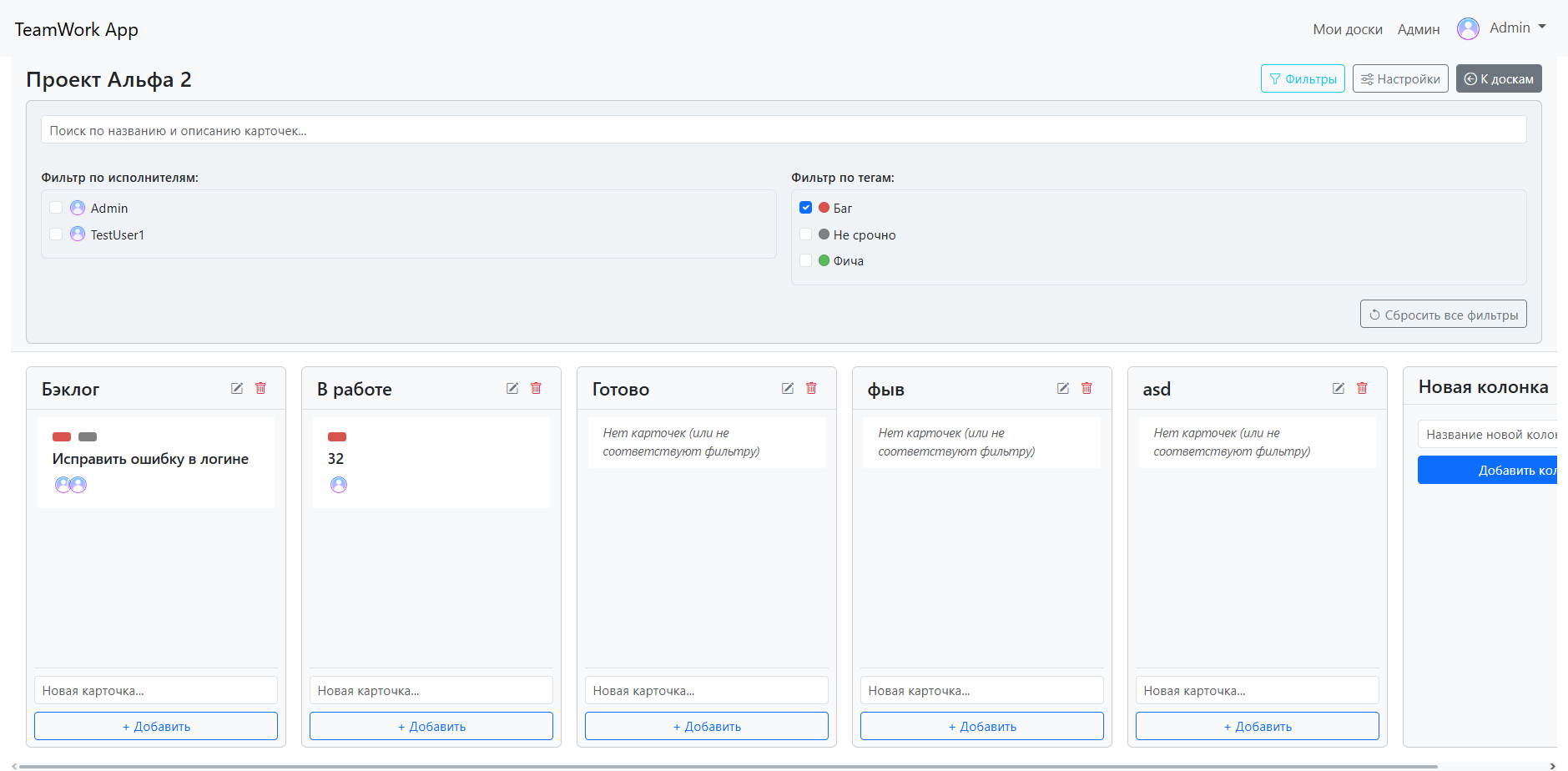


Рис. 3.18. Страница доски после применения фильтра по конкретному тегу

Тест №10 показал, что администратор может удалить пользователя. После выбора пользователя в панели администратора и удаления, пользователь исчезает из списка, представлено на рисунках 3.19 и 3.20.

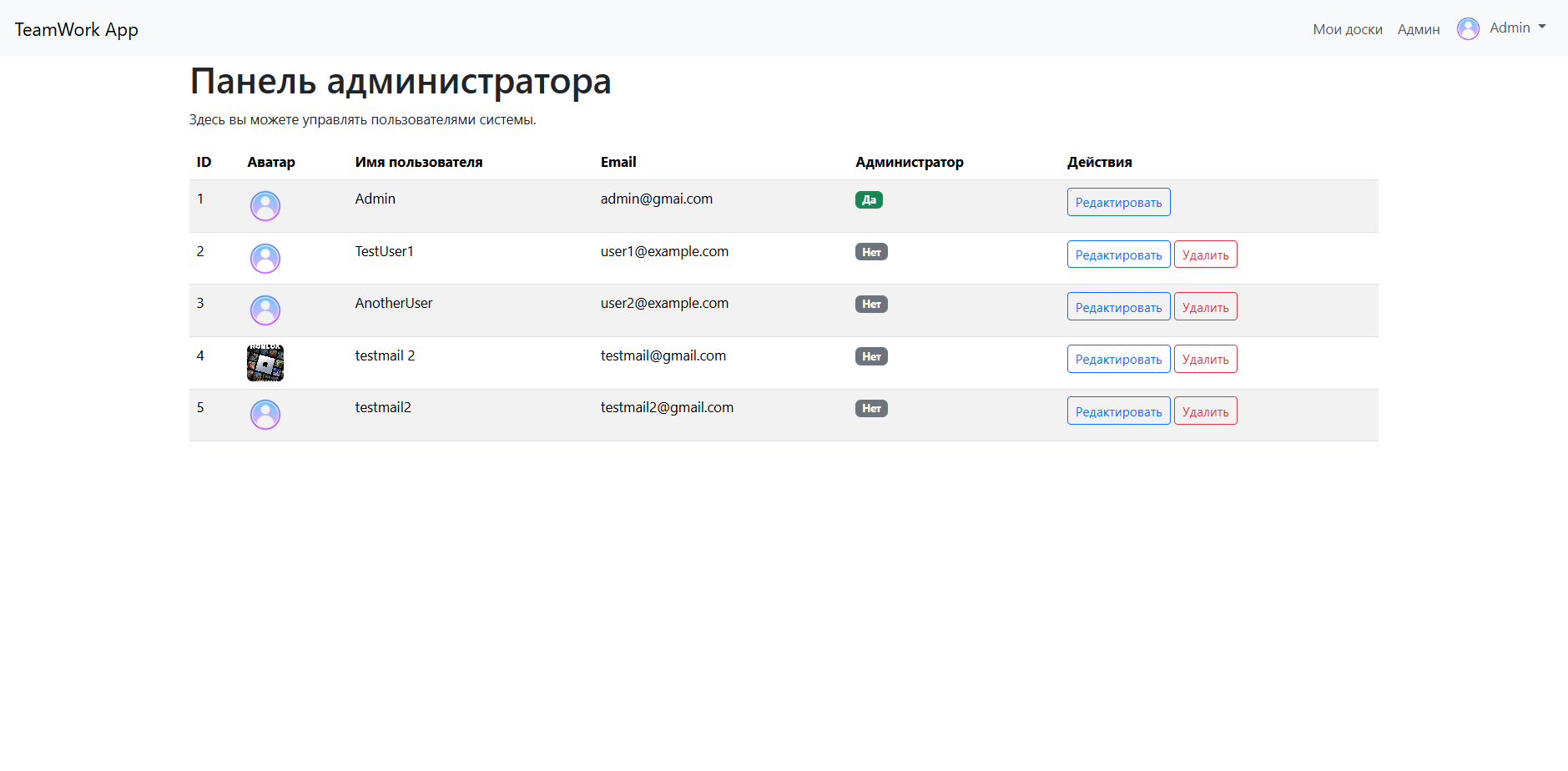


Рис. 3.19. Панель администратора со списком пользователей до удаления

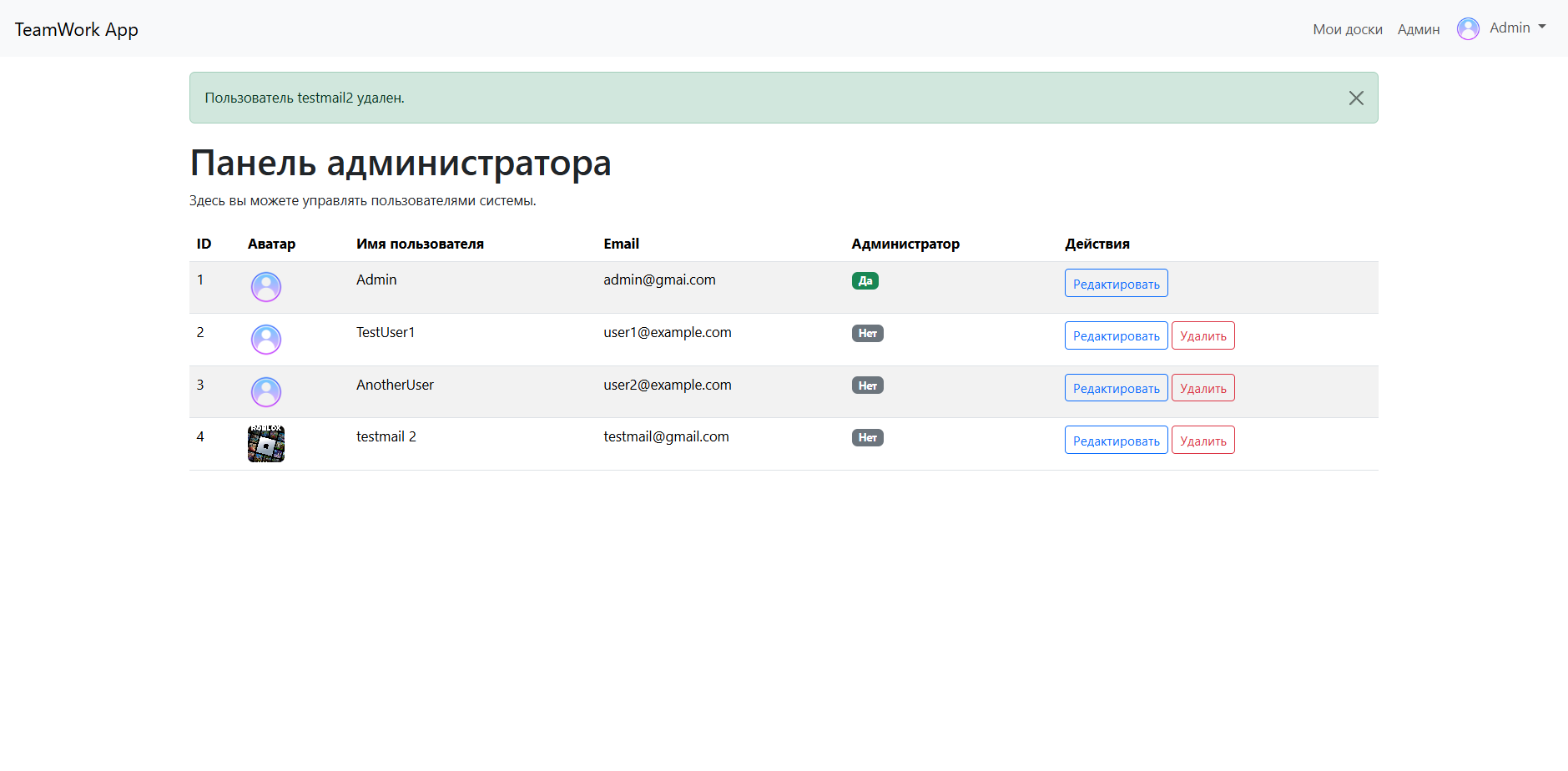


Рис. 3.20. Панель администратора со списком пользователей после удаления

# 4. РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

Для открытия веб-приложения для автоматизации командной работы необходимо запустить любой браузер на компьютере и ввести в строку поиска следующую ссылку: https://mironovroman.pythonanywhere.com, что изображено на рисунке 4.1.

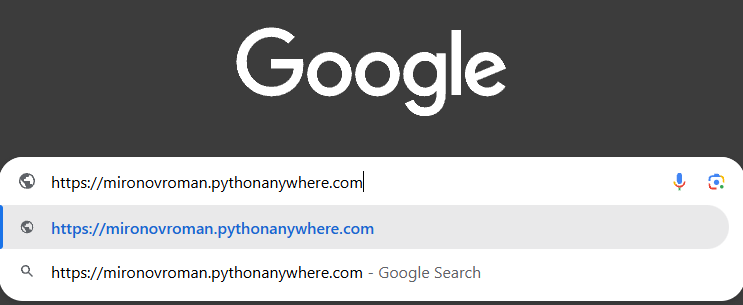


Рис.4.1. Ввод ссылки

Далее на экране отображается страница входа, изображено на рисунке 4.2.

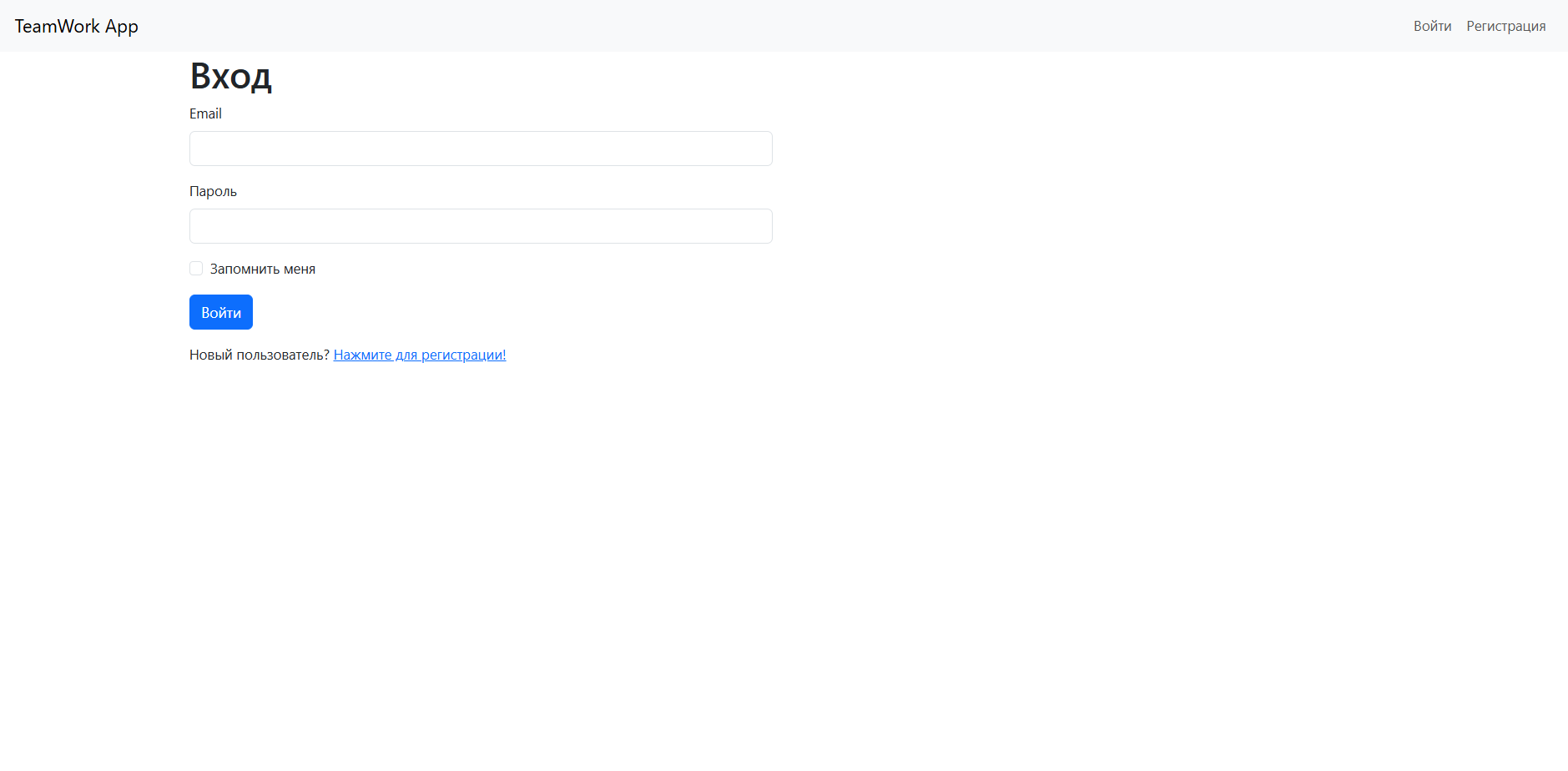


Рис. 4.2. Страница входа

При вводе своего логина, пароля и нажатия на кнопку «Войти», пользователь переходит на страницу моих досок, что изображено на рисунке 4.3.

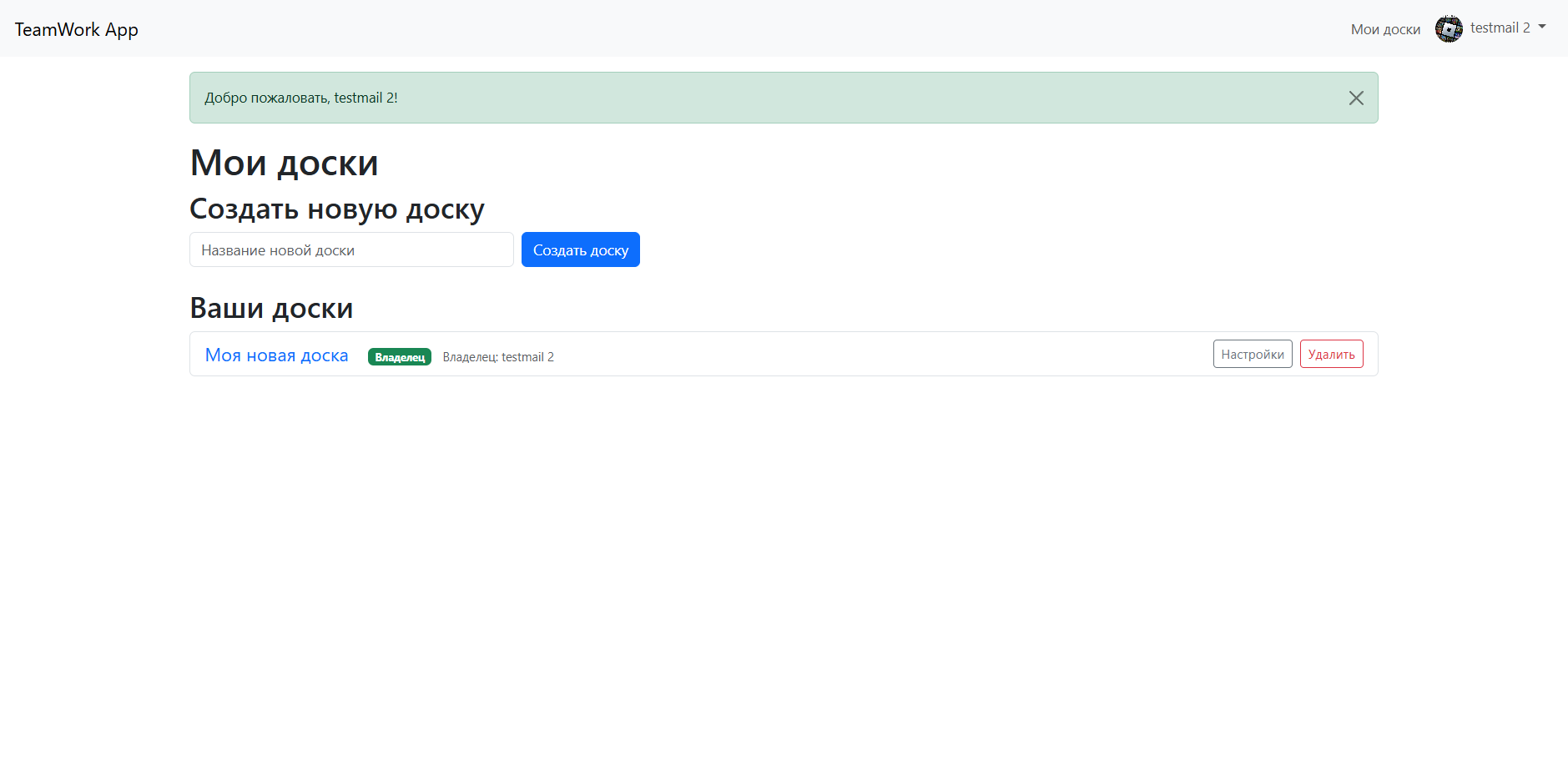


Рис. 4.3. Завершённый вход

Для того, чтобы создать доску, нужно ввести желаемое название доски и нажать «Создать доску», представлено на рисунке 4.4.

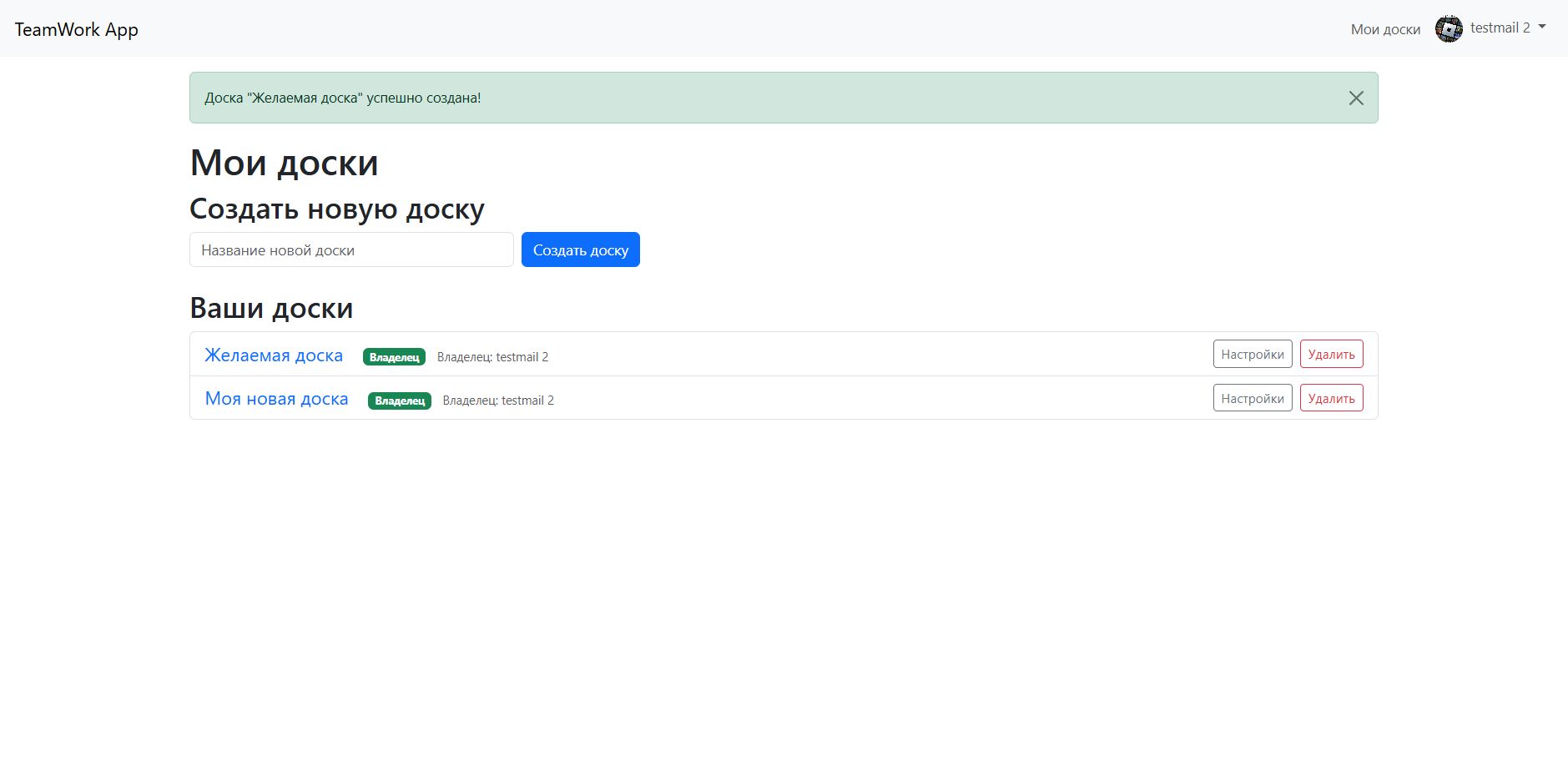


Рис. 4.4. Страница с созданной доской

При нажатии на название доски происходит переход на доску, представлено на рисунке 4.5.

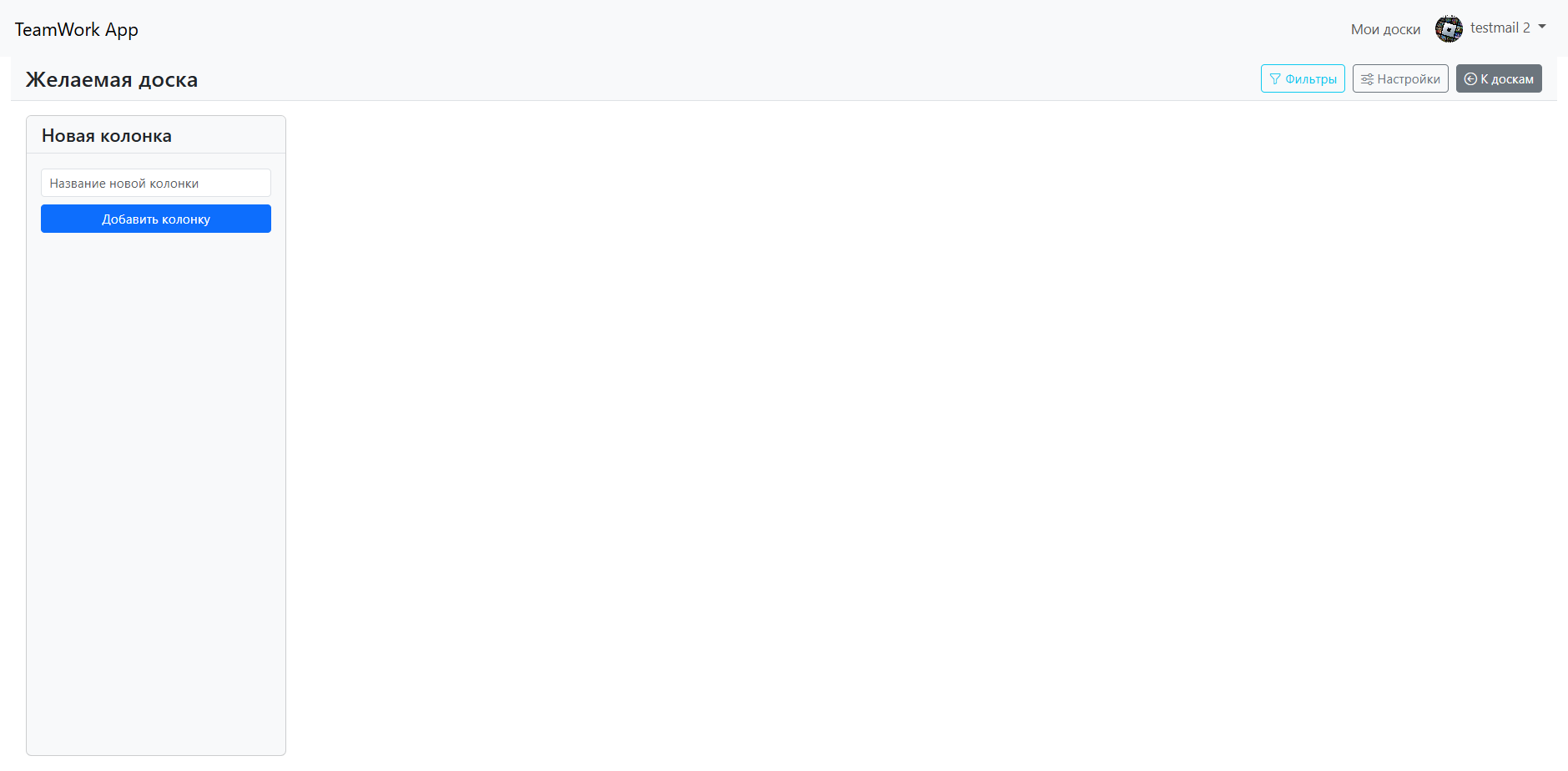


Рис. 4.5. Пустая доска

Чтобы создать колонку, нужно написать её название и нажать «Добавить колонку», представлено на рисунке 4.6.

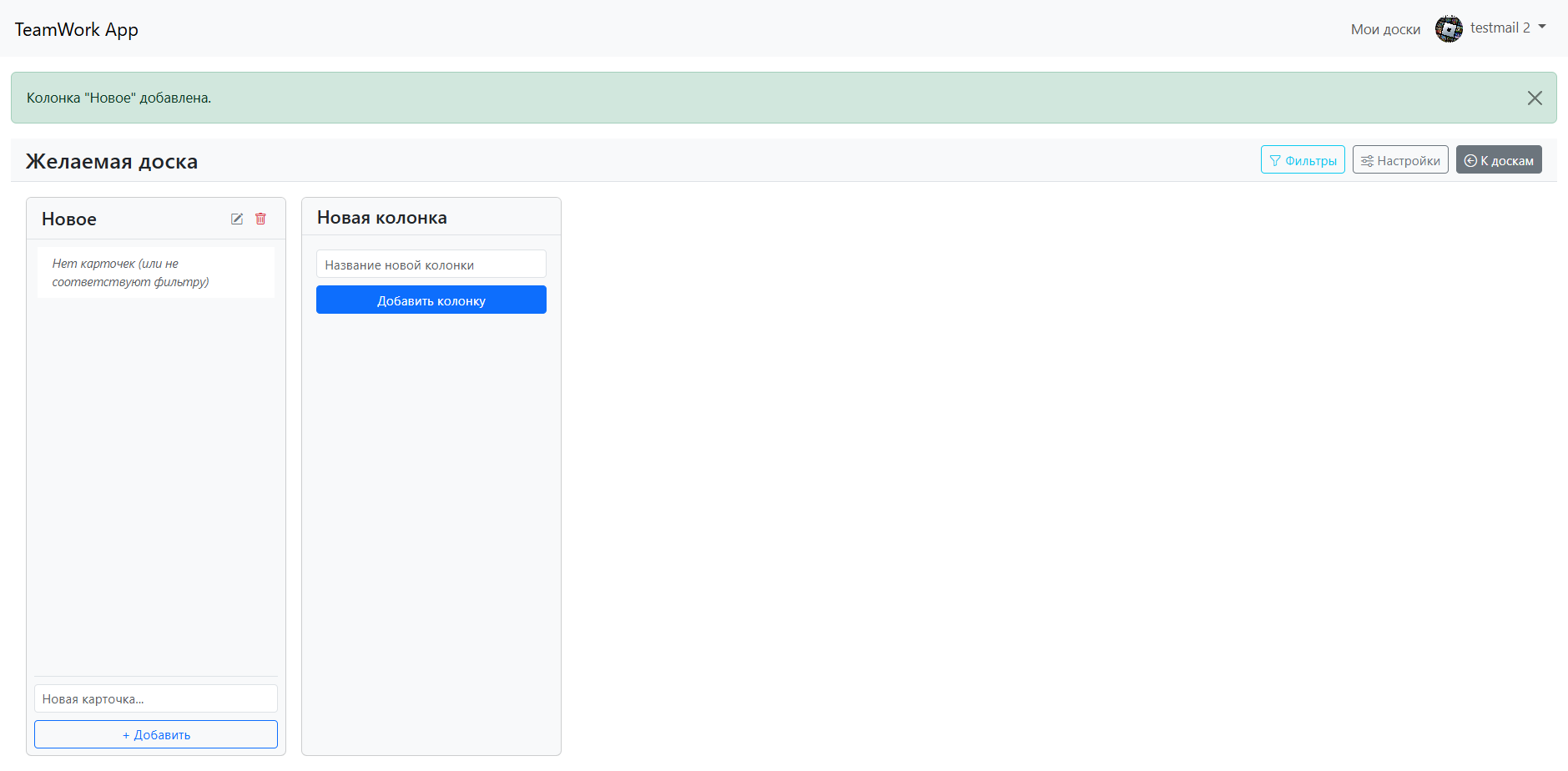


Рис. 4.6. Добавленная колонка

Чтобы добавить карточку, нужно написать её название и нажать «+ Добавить», что представлено на рисунке 4.7.

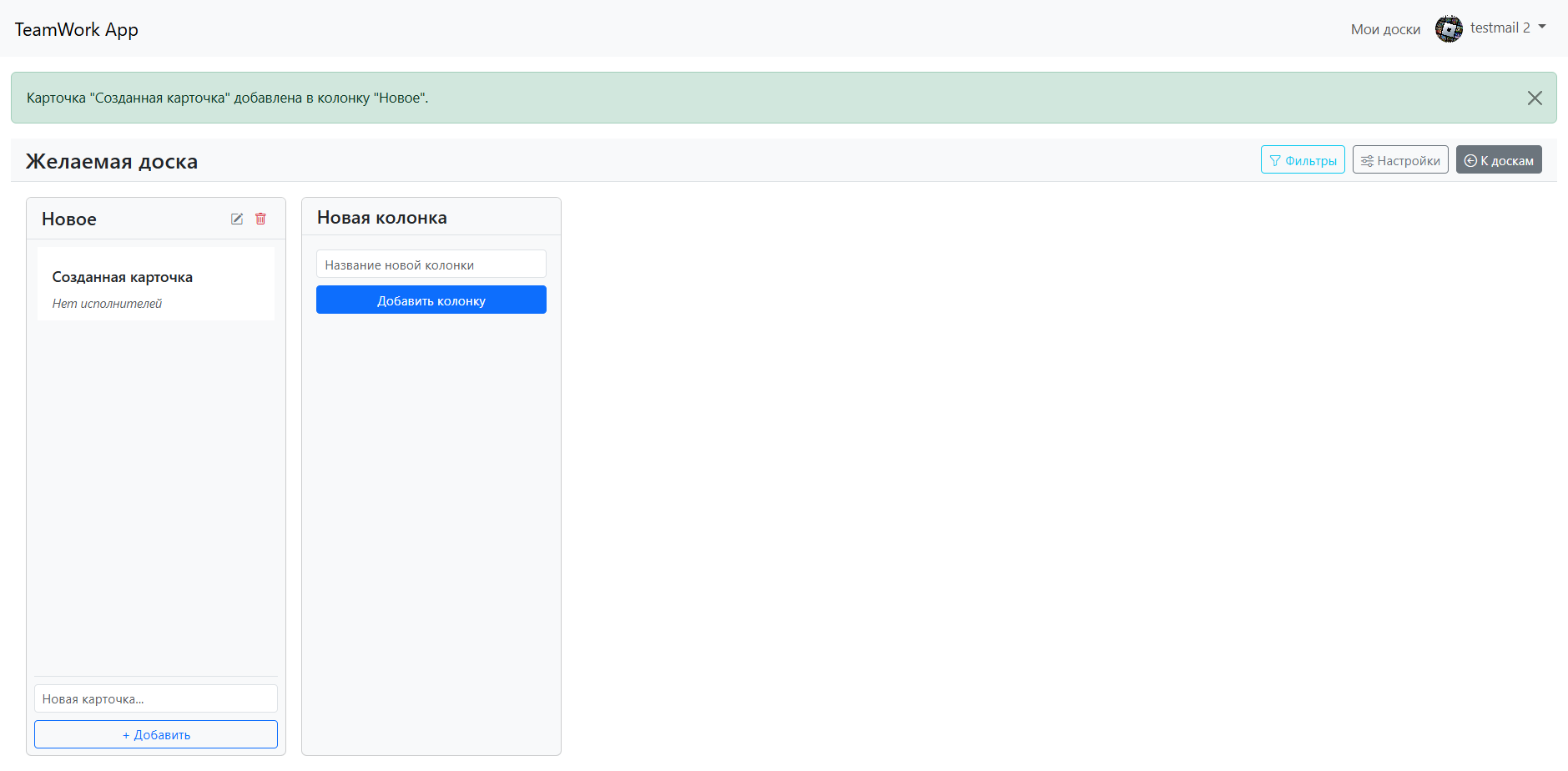


Рис. 4.7. Добавленная карточка

Чтобы изменить заголовок, описание, теги, комментарии и ответственных за выполнение задачи, нужно нажать на неё и взаимодействовать с соответствующими элементами интерфейса, после чего нажать «Сохранить изменения», изображено на рисунках 4.8 и 4.9

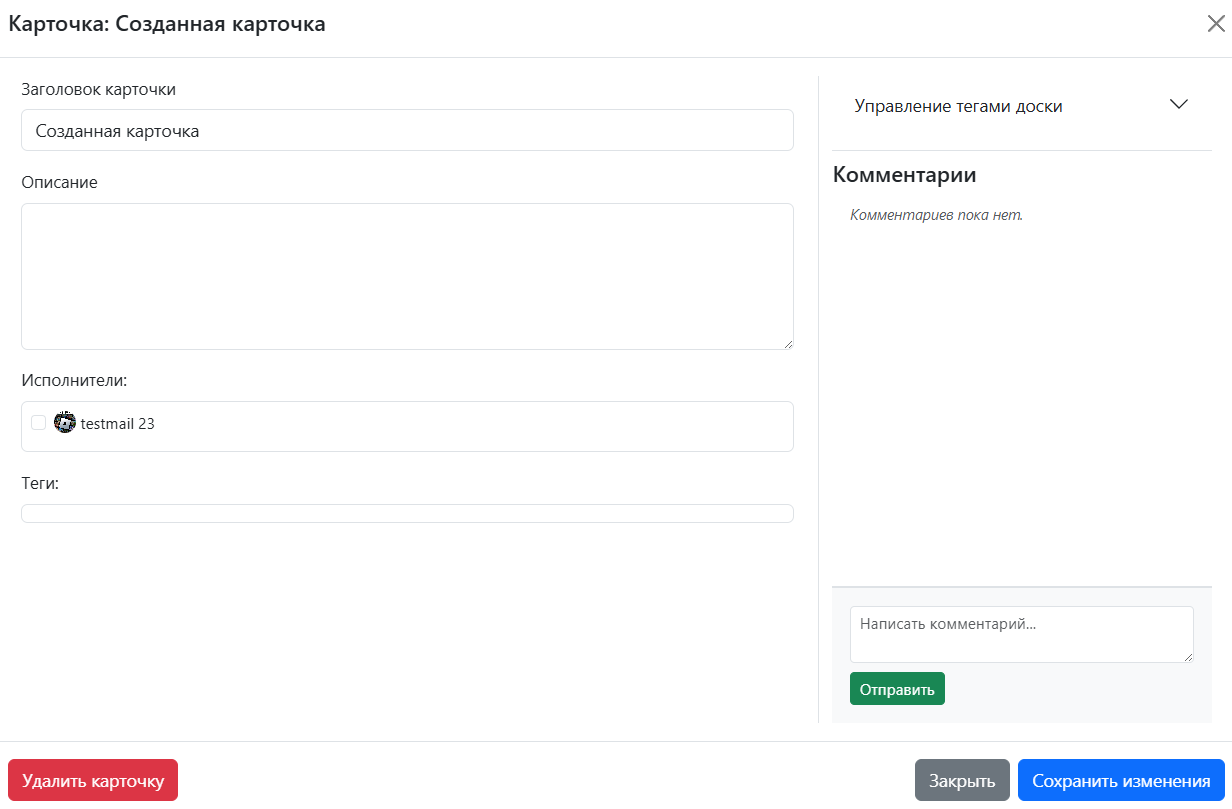


Рис. 4.8. Незаполненная карточка



Рис. 4.9. Заполненная карточка

Чтобы изменить колонку задачи, нужно задержать левую кнопку мыши на карточке и перетянуть её в желаемую колонку, изображено на рисунке 4.10.

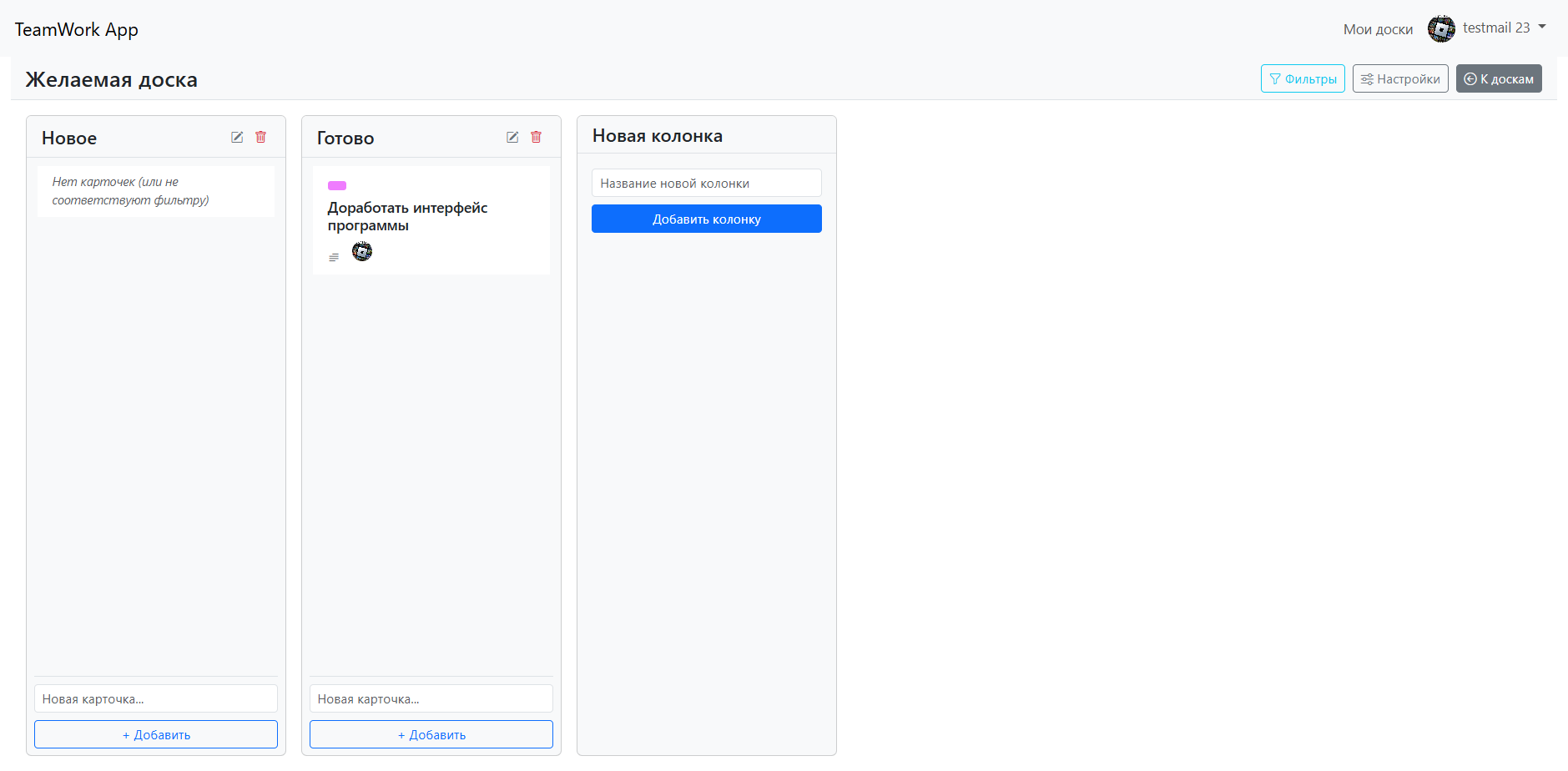


Рис. 4.10. Карточка в другой колонке

Чтобы обновить данные профиля, нужно перейти на страницу профиля, поменять данные и нажать «Загрузить аватар», «Обновить профиль» или «Сменить пароль», в зависимости от желаемых изменений, изображено на рисунках 4.11., 4.12.

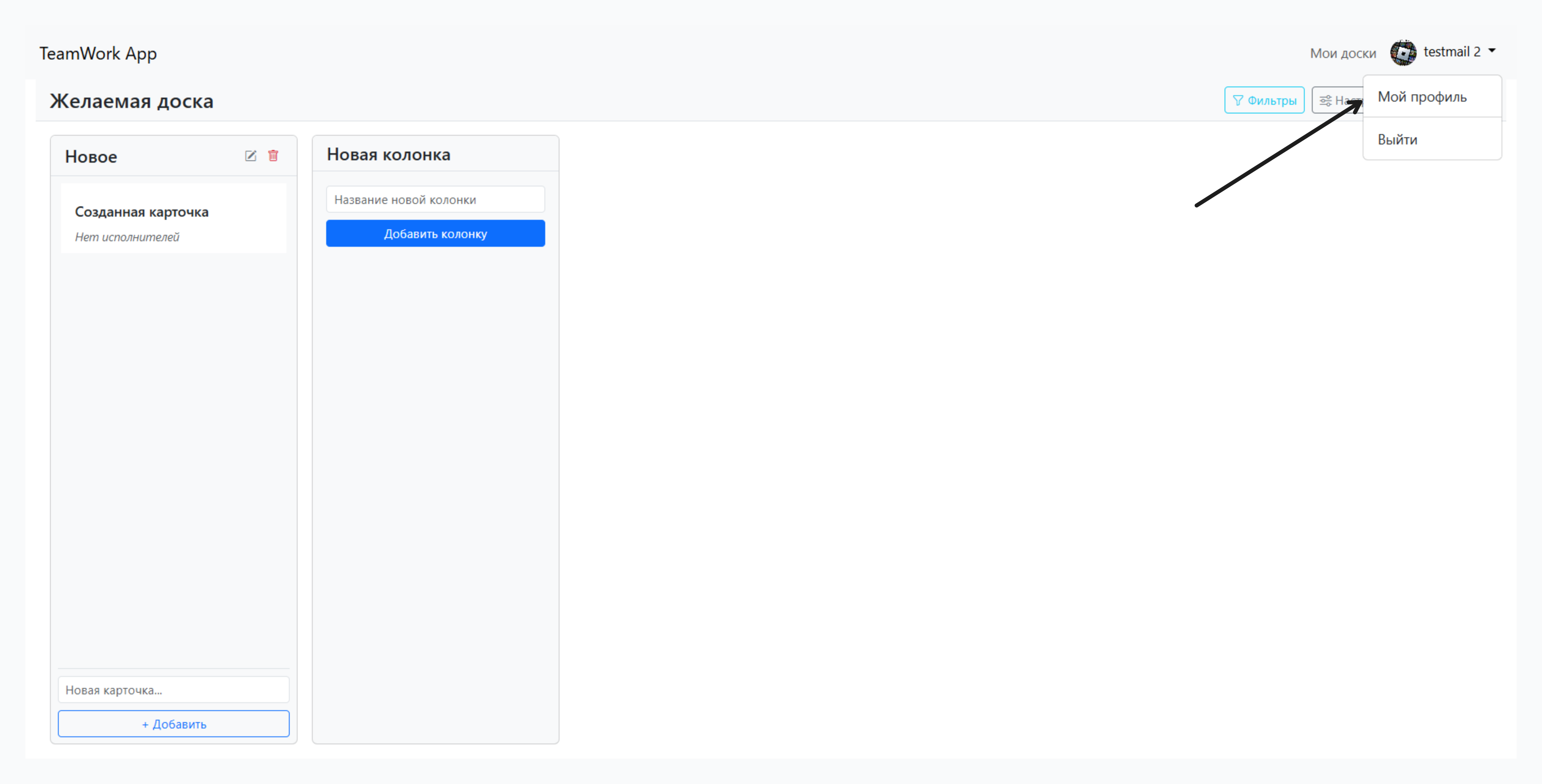


Рис. 4.11. Переход в профиль

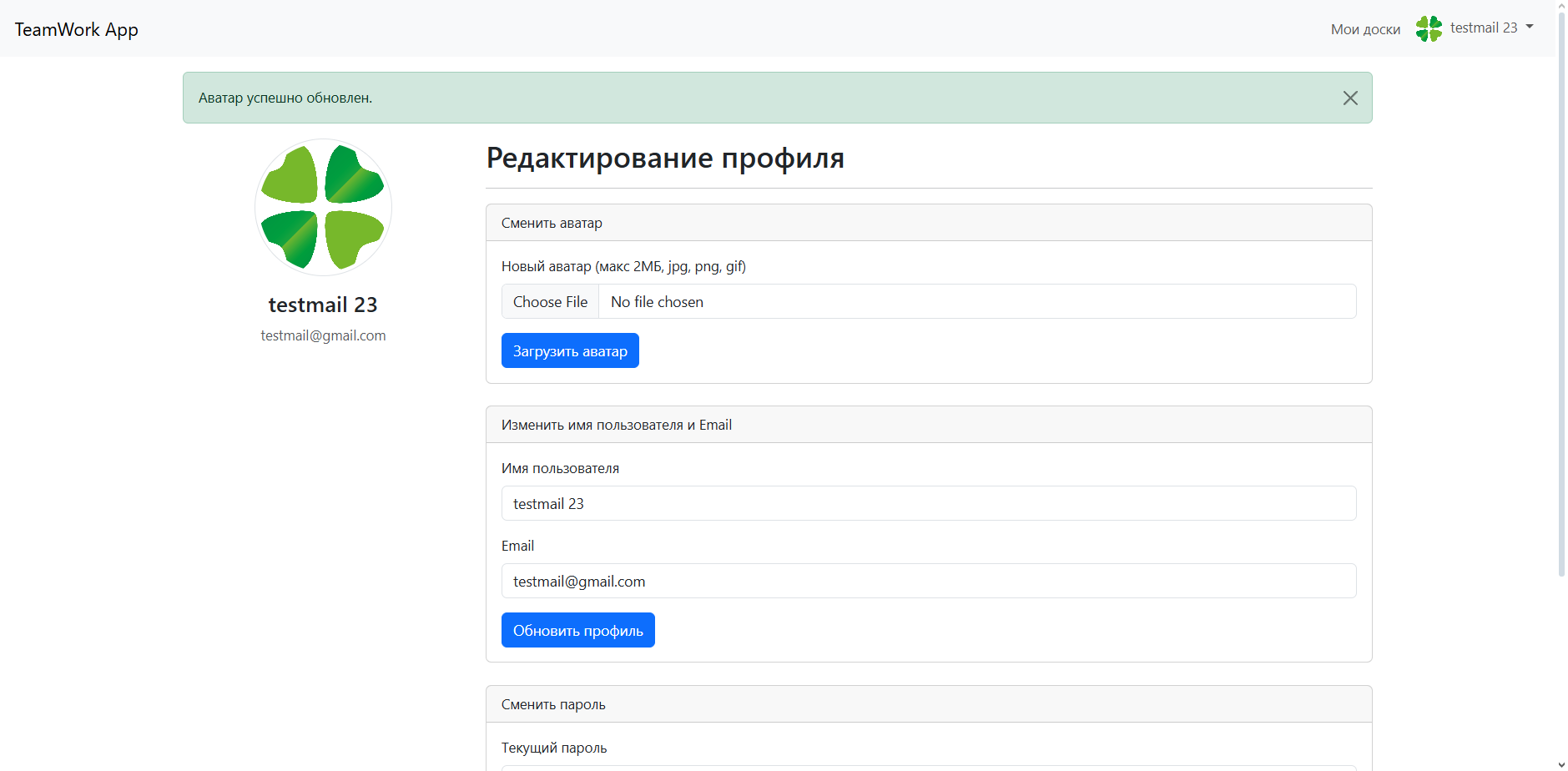


Рис. 4.12. Отредактированный профиль

Для того чтобы попасть на страницу панели администратора, нужно перейти на форму авторизации, ввести почту «admin@gmail.com» и пароль «password». После авторизации происходит переход на страницу панели администратора, изображено на рисунках 4.13., 4.14.

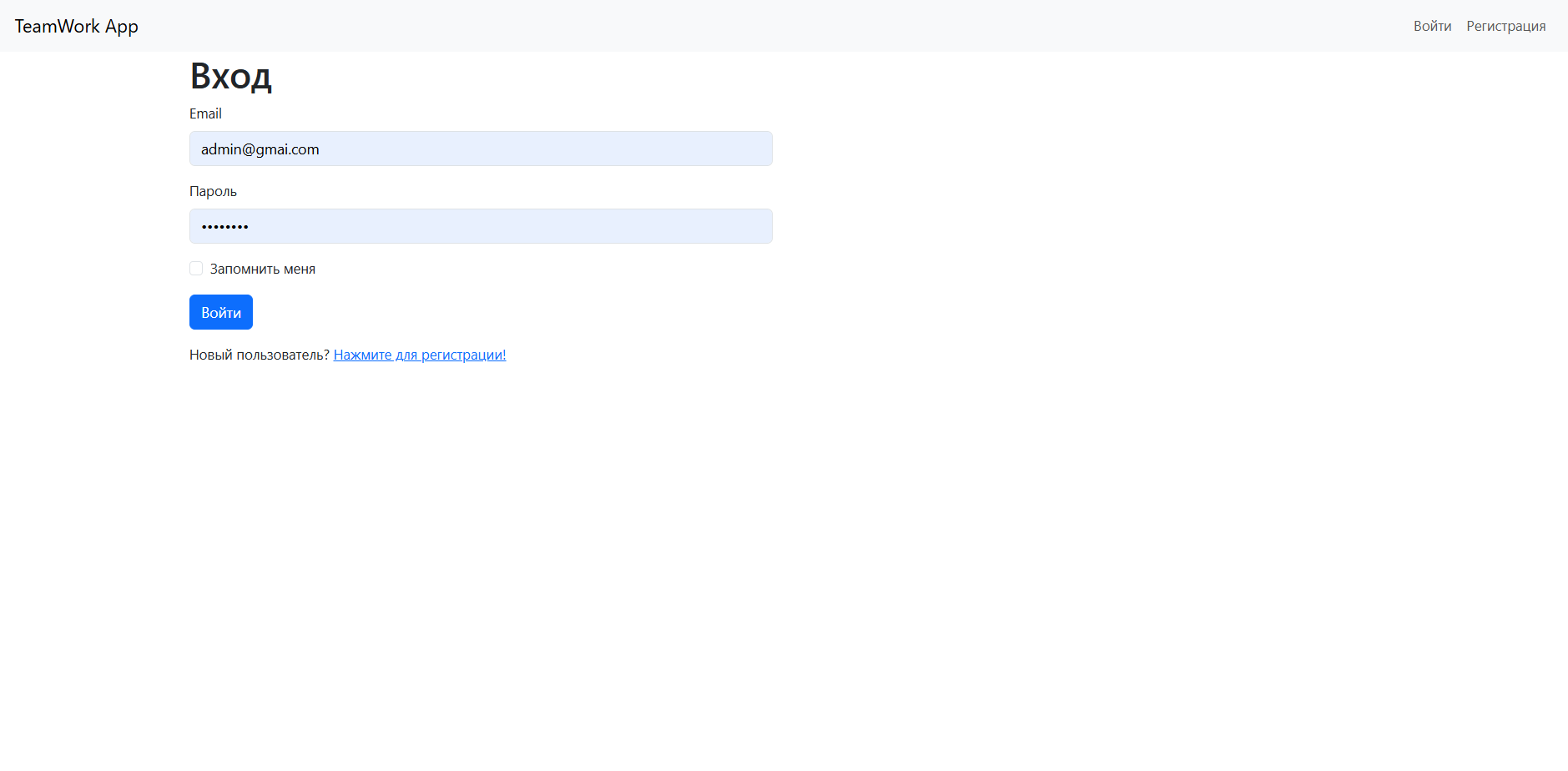


Рис. 4.13. Страница входа с введёнными данными администратора

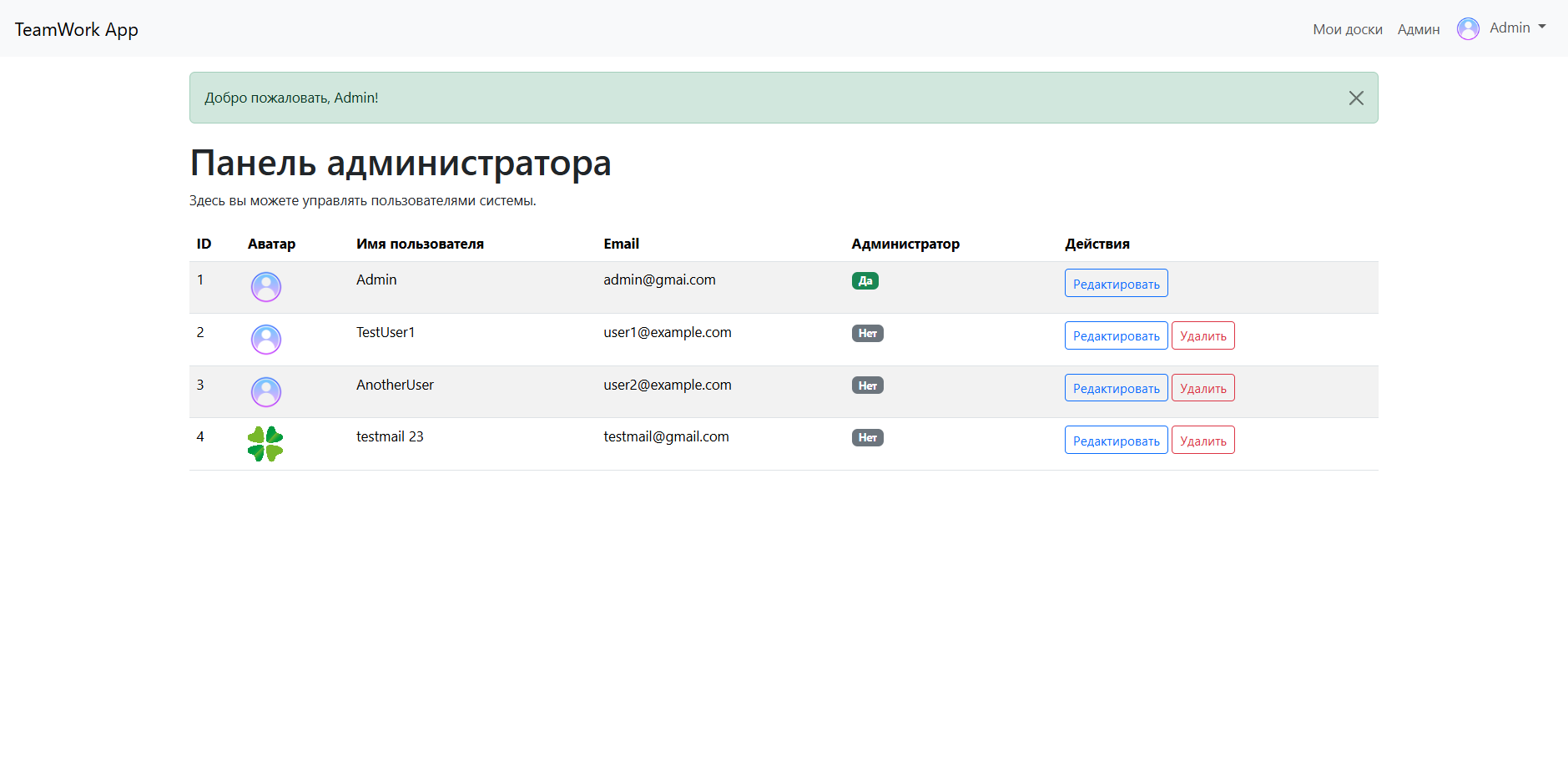


Рис. 4.14. Страница панели администратора

Чтобы отредактировать пользователя, нужно нажать на «Редактировать» на желаемом пользователе, ввести желаемую изменённую информацию, и нажать «Сохранить изменения», представлено на рисунке 4.15.

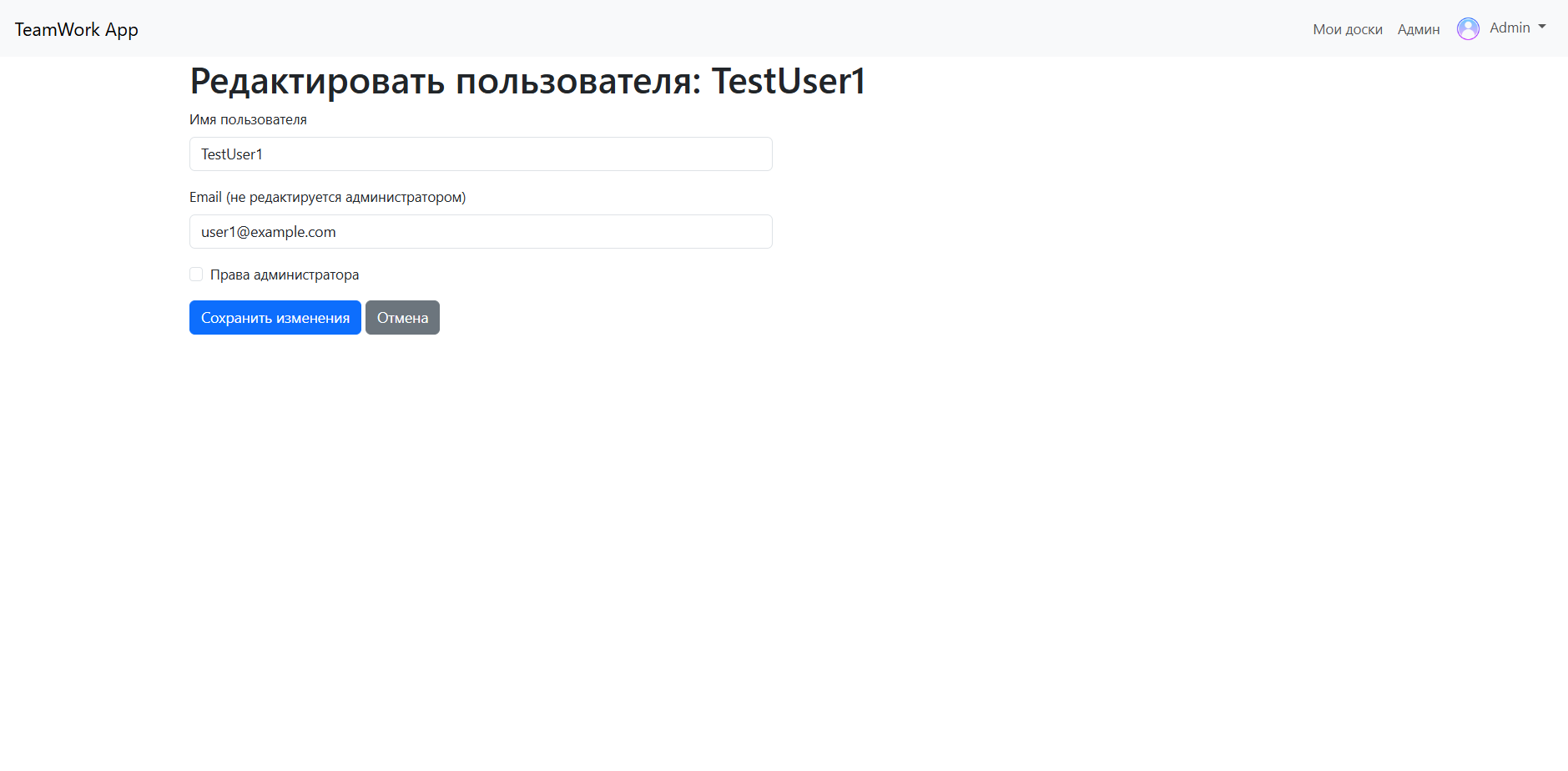


Рис. 4.15. Страница редактирования пользователя

5. ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО СРЕДСТВА

В рыночных условиях программное средство преимущественно представлено в виде продукции научно-технических организаций, представляющей собой функционально завершенные и имеющие товарный вид программные продукты, поставляемые заказчикам и продаваемые покупателям по рыночным ценам. Все завершенные разработки программного обеспечения являются научно-технической продукцией.

Стоимостная оценка программного обеспечения (ПО) и определение экономического эффекта у разработчика предполагают составление сметы затрат, которая в денежном выражении включает следующие статьи расходов:

* заработную плату исполнителей, основную (Зо) и дополнительную (Зд);
* отчисления на социальные нужды (Рсоц.);
* налоги, от фонда оплаты труда (Не);
* материалы и комплектующие (М);
* спецоборудование (Рc);
* машинное время (Рм);
* расходы на научные командировки (Рнк);
* прочие прямые расходы (Пз);
* накладные расходы (Рн).

На основе общей суммы расходов по всем статьям (Ср) и результатов маркетинговых исследований на рынке ПО определяется плановая отпускная цена (Цo) с учётом прибыли (рентабельности) и налогов, включаемых в цену.

По уточненному объёму ПО и нормативам затрат труда в расчёте на единицу объема определяются нормативная и общая трудоемкость разработки ПО.

Нормативная трудоёмкость разработки ПО (TЕн) определяется на основании принятого к расчёту объема (Vу) и категории сложности ПО (чел./дн.), указанных в таблице 1.1 приложения 1, уточняется с учётом сложности и новизны проекта, степени использования стандартных модулей при разработке. Нормативная трудоёмкость разработки ПО служит основой для определения общей трудоёмкости.

Общая трудоемкость разработки ПО (ТЕр) определяется по формуле (5.1):

TЕр = TЕн × Kс × Kн × Kт × Кур, (5.1)

где TЕн – нормативная трудоемкость разработки программного продукта, определяемая относительно общего количества строк исходного кода;

Kс – коэффициент сложности;

Kн – коэффициент, учитывающий степень новизны;

Kт – коэффициент, учитывающий использование стандартных модулей;

Кур – коэффициент, учитывающий средства разработки.

Категория сложности ПO определяется в зависимости от наличия (отсутствия) следующих характеристик:

– высокий уровень языкового интерфейса с пользователем;

– режим работы в реальном времени;

– управление удаленными объектами;

– машинная графика, многомашинные комплексы;

– существенное распараллеливание вычислений;

– нестандартная конфигурация технических средств;

– оптимизационные и особо сложные инженерные и научные расчеты;

– переносимость ПО.

Влияние фактора сложности на трудоемкость учитывается умножением нормативной трудоемкости на соответствующий коэффициент сложности.

Коэффициент сложности (Kс) учитывает дополнительные затраты труда, связанные со сложностью разрабатываемого программного продукта, при расчете которого необходимо воспользоваться дополнительными коэффициентами сложности ПО, указанными в приложении 1, таблице 1.3.

Коэффициент, учитывающий степень использования при разработке ПО стандартных модулей (Kт), определяется их удельным весом в общем объеме проектируемого продукта (Приложение 1, таблица 1.2). При определении влияния этого фактора на трудоемкость он учитывается путем умножения нормативной трудоемкости на соответствующий коэффициент.

Коэффициент новизны разрабатываемого ПО (Kн) определяется путем сравнения характеристик разрабатываемого ПО с имеющимися аналогами, позволяет определить экспертным путем степень его новизны. Влияние фактора новизны на трудоемкость учитывается путем умножения трудоемкости на соответствующий коэффициент новизны. Для расчета данного показателя используются поправочные коэффициенты, учитывающие новизну ПО (Kн) , указанные в приложении 1, таблице 1.4.

TЕp = 50 × 0,26 × 0,63 × 0,65 × 1,2 = 6 чел./дн.

Основной статьей расходов на создание ПО является заработная плата разработчиков (исполнителей) проекта, в число которых принято включать инженеров-программистов, участвующих в написании кода, руководителей проекта, системных архитекторов, дизайнеров, разрабатывающих пользовательский интерфейс, разработчиков баз данных, Web-мастеров и других специалистов, необходимых для решения специальных задач в команде. Заработная плата руководителей организации и работников вспомогательных служб (инфраструктуры) учитывается в накладных расходах.

В соответствии со статьями 1 и 63 Трудового кодекса Республики Беларусь оплата труда работника включает в себя тарифную ставку (тарифный оклад), стимулирующие (надбавки, премии, бонусы и др.) и компенсирующие выплаты. При этом конкретные формы, системы и размеры оплаты труда работников устанавливаются нанимателем на основании коллективного договора, соглашения, иных локальных правовых актов и трудового договора. В рамках данного дипломного проекта применяется стандартизированная методика калькуляции трудовых затрат, основанная на следующих принципах:

1. В состав расходов на оплату труда включены: основная заработная плата, начисляемая по тарифной системе; стимулирующие выплаты (премиальная составляющая); оплата фактически отработанного времени.

2. Нормативная база расчета: единицей измерения принят человеко-день; расчетные нормы установлены для одного программного средства; базовый режим труда: 5-дневная рабочая неделя, 8-часовая продолжительность рабочего дня.

Данная методика соответствует принципам нормирования труда в IT-отрасли и позволяет осуществить корректную оценку трудозатрат на разработку программного средства.

Основная заработная плата исполнителей на конкретное программное обеспечение рассчитывается по формуле (5.2):

ЗПосн = ТСч × ТЕр × Кпр × 8, (5.2)

где ТСч – часовая тарифная ставка соответствующего разряда, руб.;

ТЕр – трудоемкость разработки, часов;

Кпр – коэффициент премирования.

ЗПосн = 3,12 × 6 × 1,3 × 8 = 194,69 руб.

Расчет дополнительной заработной платы исполнителей за конкретное ПО (ЗПдоп) включает выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей), и определяется по нормативу в процентах к основной заработной плате, согласно формуле (5.3):

, (5.3)

где ЗПосн – основная заработная плата исполнителя, руб.;

%ЗПдоп – процент дополнительной заработной платы, %.

.

Обязательным условием является определение отчислений на социальные нужды (Рсоц) согласно формуле (5.4):

, (5.4)

где ЗПосн – основная заработная плата исполнителей, руб.;

ЗПдоп – дополнительная заработная плата исполнителей, руб.;

– норматив отчислений в ФСЗН и Белгосстрах (в соответствии с действующим законодательством)

Расходы на материалы и комплектующие определяются на основании сметы затрат, разрабатываемой на программное обеспечение с учетом действующих нормативов. В данные затраты включают расходы на магнитные носители, бумагу, красящие ленты и другие материалы, необходимые для разработки программного обеспечения. Нормы расхода материалов в суммарном выражении определяются в расчете на 100 строк исходного кода или по нормативу в процентах основной заработной платы разработчиков, который устанавливается организацией.

Сумма затрат на расходные материалы рассчитывается по формуле (5.5):

(5.5)

где ЗПосн – основная заработная плата исполнителей, руб.;

Нм – норматив расхода на материалы, %.

Расходы по статье «Машинное время» включают оплату машинного времени, необходимого для разработки и отладки ПО, которое определяется по нормативам (в машино-часах) на 100 строк исходного кода машинного времени в зависимости от характера решаемых задач и типа ПК (0,6 машино-часов (5.6):

(5.6)

где Цм – цена одного машино-часа, руб.;

Vстрок – общий объем ПО (количество строк исходного кода);

Нмв – норматив расхода машинного времени на отладку 100 строк исходного кода (машино-часов).

Общехозяйственные расходы связанны с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды, и относятся на конкретное программное обеспечение по нормативу в процентном отношении к основной заработной плате исполнителей. Норматив устанавливается в целом по организации (30-80 %) и рассчитывается по формуле (5.7):

, (5.7)

где % ОХР – норматив общехозяйственных расходов, %.

Прочие прямые затраты – это затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Они определяются по нормативу, разрабатываемому в целом по организации, в процентах к основной заработной плате и рассчитывается по формуле (5.8):

(5.8)

где Нп.з. – норматив прочих прямых затрат, %.

Результаты расчета затрат внесем в таблицу 5.1.

Таблица 5.1 – Расчет полной себестоимости программного обеспечения

|  |  |
| --- | --- |
| Наименование статей затрат | Сумма, руб. |
| Материальные затраты | 9,73 |
| Основная заработная плата | 194,69 |
| Дополнительная заработная плата | 38,94 |
| Отчисления в ФСЗН и Белгосстрах | 80,84 |
| Машинное время | 23,79 |
| Общехозяйственные расходы | 68,04 |
| Прочие прямые затраты | 19,47 |
| Полная себестоимость | 435,50 |

Для определения цены программного обеспечения необходимо рассчитать плановую прибыль.

Рентабельность и прибыль создаваемого программного продукта определяется исходя из результатов анализа рыночных условий переговоров с заказчиком и согласования с ним отпускной цены.

Прибыль рассчитывается по формуле (5.9):

(5.9)

где с/сполн – полная себестоимость ПО, руб;

Ппл – плановая прибыль от реализации ПО, руб.;

Р – уровень рентабельности ПО, %.

Отпускная цена без НДС определяется по формуле (5.10):

ЦОТП(без НДС) = с/сполн. + Ппл, (5.10)

ЦОТП(без НДС)= 435,50+ = 522,60 руб.

Сумма НДС определяется по формуле (5.11):

, (5.11)

где % НДС – ставка НДС, %.

Отпускная цена ПО с НДС(ЦОТП(с НДС))рассчитывается по формуле (5.12):

ЦОТП(с НДС**)** = ЦОТП **(**без НДС**)** + НДС, (5.12)

где НДС – сумма налога на добавленную стоимость, руб.

ЦОТП(с НДС**)** = 522,60 + 104,52 = 627,12 руб.

Полученные результаты расчетов внесем в таблицу 5.2.

Таблица 5.2 – Расчет отпускной цены программного обеспечения

|  |  |
| --- | --- |
| Показатели | Сумма, руб. |
| Полная себестоимость | 435,50 |
| Прибыль | 87,10 |
| Цена без НДС | 522,60 |
| НДС | 104,52 |
| Отпускная цена с НДС | 627,12 |

Экономическим эффектом от создания программного обеспечения для разработчика является чистая прибыль. Она определяется после уплаты налога на прибыль, формула (5.13):

, (5.13)

где %Нпр – ставка налога на прибыль, %.

.

Затраты на разработку веб-приложения для автоматизации командной работы составили 435,50 руб. Отпускная прогнозируемая цена составляет 627,12 руб. Прибыль разработчика составит руб. Рассчитанная цена разработанного программного обеспечения позволит полностью покрыть все затраты.

# 6. ОХРАНА ТРУДА



## Правовые и организационные вопросы охраны труда

Данный дипломный проект может быть использован в деятельности как на предприятиях с большой численностью работников, так и индивидуальными предпринимателями. Для безопасной реализации дипломного проекта необходимо обратить внимание на вопросы охраны труда, то есть системе обеспечения безопасности жизни и здоровья работников в процессе трудовой деятельности, включающие правовые, социально-экономические, организационно- технические, санитарно-гигиенические, лечебно-профилактические, реабилитационные и иные мероприятия и средства.

Общий контроль и руководство по соблюдению правил охраны труда в ООО «Бухгалтерская компания «КлеверКонсалт»» осуществляет инженер по охране труда Сорока Екатерина Геннадьевна. Она координирует работу по проведению всех видов инструктажей, контролю за состоянием охраны труда в ООО «Бухгалтерская компания «КлеверКонсалт»», планирует и проводит мероприятиях, направленные на профилактику травматизма и профессиональных заболеваний.

Перед тем как вновь принятые сотрудники или практиканты в ООО «Бухгалтерская компания «КлеверКонсалт»» прибудут на свое рабочее место, с ними проводится вводный инструктаж. При проведении инструктажа рассматриваются вопросы внутреннего трудового распорядка, прав и обязанностей работника и работодателя. Также сотрудники ООО «Бухгалтерская компания «КлеверКонсалт»» получают общую информации об организации и описание опасных и вредных факторов по предприятию в целом.

Со всеми вновь принятыми сотрудниками или практикантами в ООО «Бухгалтерская компания «КлеверКонсалт»» проводится первичный инструктаж, выделяется персональное рабочее место с установленными техническими средствами.

При проведении первичного инструктажа специалистом по охране трудаособое внимание уделялось самостоятельному подключению приборов к электросети. Было дополнительно подчёркнуто, что при пользовании средствами вычислительной техники и периферийным оборудованием каждый работник должен внимательно и осторожно обращаться с электропроводкой, приборами и аппаратами и всегда помнить, что пренебрежение правилами безопасности угрожает и здоровью, и жизни человека.

В инструкции, с которой я был ознакомлен в первый рабочий день, указывалось о необходимости постоянного контроля на своем рабочем месте за исправным состоянием электропроводки, выключателей, штепсельных розеток, при помощи которых оборудование включается в сеть, и заземления. Делался упор на то, что при обнаружении неисправности немедленно обесточить электрооборудование, оповестить администрацию. Продолжение работы возможно только после устранения неисправности.

В общие правила, соблюдение которых является обязательным для всех работников организации ООО «Бухгалтерская компания «КлеверКонсалт»», включены и те, которые касаются профилактики повреждения изоляции проводов и возникновения коротких замыканий, а именно не разрешается:

* вешать что-либо на провода;
* закрашивать и белить шнуры и провода;
* закладывать провода и шнуры за газовые и водопроводные трубы, за батареи отопительной системы;
* выдергивать штепсельную вилку из розетки за шнур, усилие должно быть приложено к корпусу вилки.

Среди основных правил, направленных на исключение поражения электрическим током, можно назвать следующие:

* запрещается часто включать и выключать компьютер без необходимости;
* запрещается прикасаться к экрану и к тыльной стороне блоков компьютера;
* не разрешается работать на средствах вычислительной техники и периферийном оборудовании мокрыми руками;
* не разрешается работать на средствах вычислительной техники и периферийном оборудовании, имеющих нарушения целостности корпуса, нарушения изоляции проводов, неисправную индикацию включения питания, с признаками электрического напряжения на корпусе;
* запрещается класть на средства вычислительной техники и периферийном оборудовании посторонние предметы.

В действующих инструкциях ООО «Бухгалтерская компания «КлеверКонсалт»» сотруднику при работе за монитором рекомендуется раз в 45 минут делать зарядку для глаз.

При проверке соблюдения правил охраны труда указывалось на необходимость поддержания своего рабочего места и инвентаря в чистоте, что является одной из задач работника.

## Производственная санитария и гигиена труда

В ООО «Бухгалтерская компания «КлеверКонсалт»» большинство сотрудников работают дистанционно. Лишь небольшое количество персонала находятся на рабочих местах в офисе. У каждого работника есть свое рабочее место, оборудованное техническими средствами, а именно: монитор, системный блок, клавиатура, мышь. Площадь помещения, выделенная одному работнику – 4,9 кв. м. Расстояние между столами двух сотрудников составляет около двух метров. Так же каждый стол оснащен дополнительными преградами от проводов. У каждого стола размещен удобный стул с регуляторами высоты и длины сиденья.

В действующих инструкциях ООО «Бухгалтерская компания «КлеверКонсалт»» указано, что работник обязан содержать в чистоте своё рабочее место, не допускать приёма пищи за рабочим столом. Тщательно мыть руки. Следить за порядком и препятствовать скоплению мусора.

Работа программиста в организациях и на предприятиях связана с вычислительными машинами, обработкой различных документов и информации, и подразумевает работу без физического напряжения, но под влиянием большого количества вредных производственных факторов.

При работе с персональным компьютером на программиста действуют следующие вредные факторы:

* ультрафиолетовое и инфракрасное излучения;
* электромагнитные излучения;
* рентгеновское излучение;
* статистическое электричество;
* блики и мерцание экрана;
* акустические шумы работающих принтеров, электродвигателей приводов винчестера, компакт – дисков, вентилятора процессора.

Кроме того, работа программиста за компьютером сопряжена с повышенными физическими и психологическими нагрузками, обусловленными следующими причинами:

* необходимость длительного напряжения, сохранение неподвижной позы во время работы. Длительная неподвижность приводит к усталости, болям в позвоночнике, шее, плечевых суставах;

− работа с клавиатурой вызывает боль в локтевых суставах, предплечьях, запястьях, кистях и пальцах рук;

* перегрузка зрительного аппарата.

Основная нагрузка на глаза обусловлена тем, что большая часть информации воспринимается визуально. Излишняя нагрузка на глаза может быть ослаблена правильным выбором монитора и его настройкой.

Поэтому в ООО «Бухгалтерская компания «КлеверКонсалт»» проводится комплекс мероприятий, направленных на минимизацию воздействия вредных производственных факторов. Чтобы решить проблему с микроклиматом и шумом, в ООО «Бухгалтерская компания «КлеверКонсалт»» серьезно подходят к оборудованию рабочих мест работников.

В рабочих кабинетах поддерживаются следующие показатели микроклимата:

− температура воздуха от плюс 19 до плюс 25°С;

− относительная влажность от 40 до 50%;

− скорость движения воздуха до 0,1 м/с;

− запыленность до 10 мг/м3.

Для борьбы с шумом применяются звукопоглощающие материалы.

Обязательно проводится проветривание помещений, так как в офисе становится очень душно, что может плохо сказаться на работоспособности, проветривание осуществляется три или четыре раза в течение рабочего дня.

Для создания благоприятных условий труда к организации рабочего места программиста клавиатура должны быть размещена на столе на высоте по отношению к полу 700 мм. Дисплей должен быть помещен на столе так, чтобы его экран располагался в вертикальной плоскости под углом менее 15 градусов. Во избежание перенапряжения органов зрения должно соблюдаться расстояние до монитора 40-50 см. Данные требования соблюдались мной на рабочем месте в ООО «Бухгалтерская компания «КлеверКонсалт»».

В помещении, где находилось моё рабочее место имелось естественное и искусственное освещение, что помогает избежать проблемы со зрением. Освещенность на поверхности стола в зоне размещения рабочего документа составляла 300 – 500 люкс. Освещение не создавало бликов на поверхности экрана.

Для предотвращения утомления и повышения работоспособности был установлен рациональный режим труда и отдыха. В ООО «Бухгалтерская компания «КлеверКонсалт»» установлен регламентированный режим труда и отдыха. Предприятие начинает работу с 9:00 утра и заканчивает в 18:00. Обеденное время начинается с 13:00 и заканчивается в 14:00. Помимо обеденного перерыва сотрудник имеет право взять технологический перерыв: через каждый час работы введён перерыв на 5-10 минут, а через два часа – на 15 минут. Рабочая неделя является пятидневной, начало недели в понедельник и окончание в пятницу.

## Техника безопасности (электробезопасность)

Место проведения преддипломной практики относилось к помещению без повышенной опасности. Вся оргтехника ООО «Бухгалтерская компания «КлеверКонсалт»» оборудована бесперебойными источниками питания. При отключении основного электропитания источники бесперебойного питания позволяют избежать потери информации.

Периодически проводится комплексный осмотр оборудования на внешние повреждения.

А также проводится перечень мероприятий для обеспечения электробезопасности:

1. Отключают напряжение и принимают меры, исключающие его ошибочную подачу к месту работы.
2. Вывешивают предупредительные плакаты на коммутационной аппаратуре, на постоянных и временных ограждениях.
3. Проверяют, есть ли напряжение на отключенной для работы части установки и накладывают на токоведущие части установки переносное заземление.

В ООО «Бухгалтерская компания «КлеверКонсалт»» оптимально рассчитана нагрузка на сеть, предусмотрены запасные электрогенераторы со своей электрической сетью, что исключает возможность перегрузки основной электрической сети, а при возникновении экстренной ситуации предоставляет возможность экстренно перейти на дополнительную сеть:

* немедленно прекращается работа, выключается аппаратура, о всех недостатках сообщается должностному лицу;
* после окончания работы приостанавливается функционирование всех использованных приложений и программ.

При проведении первичного инструктажа особое внимание уделялось самостоятельному подключению приборов к электросети. Было дополнительно подчёркнуто, что при пользовании средствами вычислительной техники и периферийным оборудованием каждый работник должен внимательно и осторожно обращаться с электропроводкой, приборами и аппаратами и всегда помнить, что пренебрежение правилами безопасности угрожает и здоровью, и жизни человека.

Сорокой Екатериной Геннадьевной, инженером по охране труда, неоднократно указывалось на то, что необходимо постоянно следить на своем рабочем месте за исправным состоянием электропроводки, выключателей, штепсельных розеток, при помощи которых оборудование включается в сеть, и заземления. Делался упор на то, что при обнаружении неисправности немедленно обесточить электрооборудование, оповестить администрацию. Продолжение работы возможно только после устранения неисправности;

Во избежание повреждения изоляции проводов и возникновения коротких замыканий не разрешается:

* вешать что-либо на провода;
* закрашивать и белить шнуры и провода;
* закладывать провода и шнуры за газовые и водопроводные трубы, за батареи отопительной системы;
* выдергивать штепсельную вилку из розетки за шнур, усилие должно быть приложено к корпусу вилки.

Для исключения поражения электрическим током запрещается:

* часто включать и выключать компьютер без необходимости;
* прикасаться к экрану и к тыльной стороне блоков компьютера;
* работать на средствах вычислительной техники и периферийном оборудовании мокрыми руками;
* работать на средствах вычислительной техники и периферийном оборудовании, имеющих нарушения целостности корпуса, нарушения изоляции проводов, неисправную индикацию включения питания, с признаками электрического напряжения на корпусе.

## Пожарная безопасность

Среди возможных причин возникновения пожара при эксплуатации ПЭВМ (возникновение аварийных ситуаций) можно назвать следующие:

* короткие замыкания;
* перегрузки;
* повышение переходных сопротивлений в электрических контактах;
* перенапряжение;
* возникновение токов утечки.

В ООО «Бухгалтерская компания «КлеверКонсалт»» принимаются меры, соблюдение которых помогает исключить с большой вероятностью возможность возникновения пожара:

1) Для понижения воспламеняемости и способности распространять пламя кабели покрывают огнезащитным покрытием;

2) При ремонтно-профилактических работах строго соблюдаются правила пожарной безопасности;

3) Каждое из помещений, где производится эксплуатация устройств ПЭВМ, оборудовано первичными средствами пожаротушения и обеспечено инструкциями по их применению. В качестве средств пожаротушения в ООО «Бухгалтерская компания «КлеверКонсалт»» разрешается использование углекислотного огнетушителя типа ОУ-2, ОУ-5, а также порошковый тип. Применение пенных огнетушителей не допускается, так как жидкость пропускает ток;

4) Устройства ПЭВМ установлены вдали отопительных и нагревательных приборов (расстояние не менее 1 м и в местах, где не затруднена их вентиляция и нет прямых солнечных лучей);

5) Разработаны организационные меры по обучению персонала навыкам ликвидации пожара имеющимися в наличии средствами тушения пожара до прибытия пожарного подразделения.

К средствам тушения пожара, предназначенных для локализации небольших загораний в помещениях с ПЭВМ относятся:

* пожарные стволы;
* внутренние пожарные водопроводы;
* огнетушители;
* сухой песок;
* асбестовые одеяла.

Для профилактики действий при пожаре разработан и неукоснительно применяется специальный план эвакуации из помещения.

В каждом кабинете ООО «Бухгалтерская компания «КлеверКонсалт»» расположены плакаты с памятками о мероприятиях по соблюдению правил охраны труда и техники безопасности. На этих стендах можно получить основную информацию о мероприятиях по охране труда, указаны ответственные за соблюдение пожарной безопасности и соблюдение правил охраны труда.

В ООО «Бухгалтерская компания «КлеверКонсалт»» проводятся меры по устранению причин пожаров, осуществляется постоянная уборка мусора, который может вызвать возгорание, электроинструмент хранится в специальных условиях.

В ООО «Бухгалтерская компания «КлеверКонсалт»» осуществляется ряд мероприятий, направленных на предотвращение огня, на устройство путей эвакуации людей и материальных ценностей, на использование средств и условий для быстрой ликвидации пожара. Присутствует план эвакуации людей при возникновении пожара из помещения, а также плакаты первой медицинской помощи.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта было разработано веб-приложение для автоматизации командной работы, которое позволяет пользователям регистрироваться, создавать доски, управлять на них колонками и карточками, а также эффективно взаимодействовать в команде. Проект демонстрирует значительный потенциал в области улучшения организации рабочих процессов и повышения продуктивности командной работы, предоставляя пользователям интуитивно понятный инструмент для визуализации задач и отслеживания прогресса.

Разработанное веб-приложение успешно сочетает в себе удобство и функциональность, предлагая пользователям простой интерфейс и логичную структуру для управления проектами в стиле Kanban. Интеграция таких элементов, как комментарии, теги, управление участниками и назначение исполнителей, способствует эффективной совместной работе и делает веб-приложение для автоматизации командной работы не просто инструментом для управления задачами, но и платформой для командного взаимодействия.

Программное средство разработано в среде Visual Studio Code с использованием языка программирования Python 3 и веб-фреймворка Flask. Для работы с базой данных использовалась SQLite с ORM Flask-SQLAlchemy. Пользовательский интерфейс реализован с помощью HTML5, CSS3 и JavaScript.

Тестирование веб-приложения подтвердило его корректную работу и соответствие заявленным требованиям, а предполагаемое руководство по использованию обеспечит легкость освоения функционала. Рассмотрение аспектов охраны труда и анализ трудозатрат в рамках соответствующих разделов подчеркивают завершенность проекта как дипломной работы.

Затраты на разработку веб-приложения для автоматизации командной работы составили 435,50 руб. Отпускная прогнозируемая цена составляет 627,12 руб. Прибыль разработчика составит 69,68 руб. Рассчитанная цена разработанного программного обеспечения позволит полностью покрыть все затраты.

В заключение, веб-приложение для автоматизации командной работы является успешным примером практической реализации системы управления проектами. Он отвечает современным требованиям к инструментам для организации командной работы, предлагая пользователям не только удобный инструмент для управления задачами и проектами, но и возможность эффективного взаимодействия внутри команды.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. CSS-Tricks [Электронный ресурс] / A Complete Guide to Flexbox. Режим доступа: css-tricks.com – Дата доступа: 08.05.2025.
2. Flask Documentation [Электронный ресурс] / Quickstart. Режим доступа: https://flask.palletsprojects.com – Дата доступа: 10.04.2025.
3. Flask-Login Documentation [Электронный ресурс] / Usage. Режим доступа: https://flask-login.readthedocs.io – Дата доступа: 02.05.2025.
4. Flask-SQLAlchemy Documentation [Электронный ресурс] / Basic Application Setup. Режим доступа: https://flask-sqlalchemy.palletsprojects.com – Дата доступа: 18.04.2025.
5. Flask-WTF Documentation [Электронный ресурс] / Quickstart. Режим доступа: https://flask-wtf.readthedocs.io – Дата доступа: 25.04.2025.
6. FreeCodeCamp [Электронный ресурс] / JavaScript DOM Manipulation. Режим доступа: https://freecodecamp.org – Дата доступа: 12.05.2025.
7. MDN Web Docs: JavaScript [Электронный ресурс] / JavaScript. Режим доступа: https://developer.mozilla.org – Дата доступа: 09.04.2025.
8. MDN Web Docs: Using the Fetch API [Электронный ресурс] / Using Fetch. Режим доступа: https://developer.mozilla.org – Дата доступа: 19.05.2025.
9. Python Official Documentation [Электронный ресурс] / The Python Tutorial. Режим доступа: https://docs.python.org – Дата доступа: 11.05.2025.
10. Real Python [Электронный ресурс] / Flask by Example – Building a Task List App. Режим доступа: https://realpython.com – Дата доступа: 05.05.2025.
11. SQLite Official Documentation [Электронный ресурс] / SQL Syntax. Режим доступа: https://sqlite.org – Дата доступа: 17.05.2025.
12. Libeldoc.bsuir.by [Электронный ресурс] / Технико-экономическое обоснование дипломных проектов. Режим доступа: https://libeldoc.bsuir.by/bitstream/123456789/360/2/Palicin\_ch4.pdf – Дата доступа: 19.05.2025.
13. News-newsby-org.narod.ru [Электронный ресурс] / Методические указания к выполнению экономической части дипломного проекта. Режим доступа: https://news-newsby-org.narod.ru/doc-razn/pos05/postn05236/index.htm – Дата доступа: 13.05.2025
14. Fsc.bsu.by [Электронный ресурс] / Безопасность жизнедеятельности в дипломных проектах. Режим доступа: https://fsc.bsu.by/wp-content/uploads/2017/01/tema-Obespechenie-ohrany-truda-v-rb.pdf – Дата доступа: 21.05.2025

# ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

ВП – веб-приложение

БД – база данных

ПО – программное обеспечение

ДП – дипломный проект

СУБД – система управления базами данных

ПРИЛОЖЕНИЕ 1 (ОБЯЗАТЕЛЬНОЕ) – УКРУПНЕННЫЕ НОРМЫ ЗАТРАТ ТРУДА НА РАЗРАБОТКУ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Таблица 1.1 – Нормативная трудоемко сть на разработку по (ТН) в зависимости от уточненного объема по (Vy) и категории сложности по (чел./дн.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Объем ПО  (строки исходного кода  (LOC)), Vy, до | Категория сложности ПО | | | Номер  нормы |
| 1-я | 2-я | 3-я |
| 400 | 30 | 25 | 21 | 1 |
| 420 | 31 | 26 | 22 | 2 |
| 440 | 32 | 28 | 23 | 3 |
| 470 | 34 | 29 | 25 | 4 |
| 490 | 36 | 30 | 26 | 5 |
| 520 | 38 | 32 | 27 | 6 |
| 550 | 40 | 34 | 29 | 7 |
| 580 | 42 | 35 | 30 | 8 |
| 610 | 44 | 37 | 32 | 9 |
| 640 | 46 | 39 | 33 | 10 |
| 680 | 48 | 41 | 35 | 11 |
| 710 | 50 | 43 | 36 | 12 |
| 750 | 53 | 45 | 38 | 13 |
| 790 | 56 | 47 | 40 | 14 |
| 840 | 59 | 50 | 42 | 15 |
| 880 | 61 | 52 | 44 | 16 |
| 930 | 65 | 55 | 46 | 17 |
| 980 | 68 | 57 | 49 | 18 |
| 1040 | 72 | 61 | 51 | 19 |
| 1090 | 75 | 63 | 53 | 20 |
| 1150 | 79 | 66 | 56 | 21 |
| 1210 | 82 | 69 | 59 | 22 |
| 1280 | 87 | 73 | 62 | 23 |
| 1350 | 91 | 77 | 65 | 24 |
| 1420 | 95 | 80 | 68 | 25 |
| 1500 | 100 | 85 | 71 | 26 |
| 1580 | 105 | 89 | 75 | 27 |
| 1670 | 111 | 93 | 79 | 28 |
| 1760 | 116 | 98 | 83 | 29 |
| 1850 | 122 | 102 | 86 | 30 |
| 1960 | 128 | 108 | 91 | 31 |
| 2060 | 134 | 113 | 95 | 32 |
| 2170 | 141 | 119 | 100 | 33 |
| 2290 | 148 | 125 | 105 | 34 |

*Продолжение таблицы 1.1*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Объем ПО  (строки исходного кода  (LOC)), Vy, до | Категория сложности ПО | | | | Номер  нормы | |
| 2420 | 156 | 131 | 110 | 35 | |
| 2550 | 163 | 137 | 116 | 36 | |
| 2690 | 172 | 144 | 122 | 37 | |
| 2800 | 178 | 150 | 126 | 38 | |
| 2990 | 189 | 159 | 134 | 39 | |
| 3150 | 198 | 167 | 140 | 40 | |

Таблица 1.2 – Перечень и объем функций программного модуля

|  |  |  |  |
| --- | --- | --- | --- |
| N функции | Наименование (содержание) | Объем функции (LОС) | |
|  |  | по каталогу Vi | уточненный Vyi |
| 101 | Организация ввода информации | 150 | 150 |
| 203 | Формирование баз данных | 2180 | 2180 |
| 204 | Обработка наборов и записей баз  данных | 2670 | 2670 |
| 208 | Организация поиска и поиск в базе  данных | 5480 | 3836 |
| 210 | Загрузка базы данных | 2780 | 2780 |
| 703 | Расчет показателей | 460 | 460 |
| 705 | Формирование и вывод на внешние  носители | 3500 | 2450 |
| 706 | Предварительная обработка и печать файлов | 470 | 470 |
|  | Итого | 17 690 | 14 996 |

Таблица 1.3 – Дополнительные коэффициенты сложности ПО

|  |  |
| --- | --- |
| Характеристика ПО | Значения Kс |
| 1. Функционирование ПО в расширенной операционной среде (связь с другими ПО). | 0,08 |
| 1. Интерактивный доступ. | 0,06 |
| 1. Обеспечение хранения, ведения и поиска данных в сложных структурах. | 0,07 |

*Продолжение таблицы 1.3*

|  |  |
| --- | --- |
| Характеристика ПО | Значения Kс |
| 1. Наличие у ПО одновременно нескольких характеристик по данным из табл., отражающей «Характеристики категорий сложности ПО»:    1. 2 характеристики.    2. 3 характеристики.    3. Свыше 3-х характеристик. | 0,12  0,18  0,26 |

Таблица 1.4 – Поправочные коэффициенты, учитывающие новизну ПО (Kн)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория новизны ПО | Степень новизны | Использование | | Значение Kн |
| На основе нового типа ПК | В среде  новой ОС |
| А | Принципиально новые ПО, не имеющие доступных аналогов | +  -  +  - | +  +  -  - | 1,58  1,44  1,10  1,0 |
| Б | ПО, являющиеся развитием определенного параметрического ряда ПО | +  -  + | +  -  - | 1,0  0,81  0,72 |
| В | ПО, являющиеся развитием определенного параметрического ряда ПО, разработанных для ранее освоенных типов конфигурации ПК и ОС | - | - | 0,63 |

ПРИЛОЖЕНИЕ 2 (ОБЯЗАТЕЛЬНОЕ) – ЛИСТИНГ ПРОГРАММЫ

# app/routes.py

from flask import render\_template, flash, redirect, url\_for, request, jsonify, abort

from flask\_login import login\_user, logout\_user, current\_user, login\_required

from werkzeug.security import generate\_password\_hash

from werkzeug.utils import secure\_filename

from app import app, db

from app.forms import (

LoginForm, RegistrationForm, BoardForm, ColumnForm, CardForm, CommentForm,

InviteUserForm, UpdateAccountForm, ChangePasswordForm, UpdateAvatarForm, AdminEditUserForm,

TagForm

)

from app.models import User, Board, Column, Card, Comment, Tag

from sqlalchemy import or\_, exc

import os

from functools import wraps

from wtforms import SelectField, SelectMultipleField

from datetime import datetime

def admin\_required(f):

@wraps(f)

def decorated\_function(\*args, \*\*kwargs):

if not current\_user.is\_authenticated or not current\_user.is\_admin:

flash('Доступ запрещен. У вас нет прав администратора.', 'danger')

return redirect(url\_for('index'))

return f(\*args, \*\*kwargs)

return decorated\_function

def \_populate\_assignee\_choices(form, board):

eligible\_users = board.get\_eligible\_assignees()

choices\_for\_multiple = [(user.id, user.username) for user in eligible\_users]

if hasattr(form, 'assignees') and isinstance(form.assignees, SelectMultipleField):

form.assignees.choices = choices\_for\_multiple

def \_populate\_tag\_choices(form, board):

if hasattr(form, 'tags') and isinstance(form.tags, SelectMultipleField):

board\_tags = board.tags.order\_by(Tag.name).all()

form.tags.choices = [(tag.id, tag.name) for tag in board\_tags]

@app.route('/')

@app.route('/index')

def index():

if current\_user.is\_authenticated:

return redirect(url\_for('dashboard'))

return render\_template('index.html', title='Главная')

@app.route('/dashboard', methods=['GET', 'POST'])

@login\_required

def dashboard():

form = BoardForm()

if form.validate\_on\_submit():

new\_board = Board(name=form.name.data, owner=current\_user)

db.session.add(new\_board)

db.session.commit()

flash(f'Доска "{new\_board.name}" успешно создана!', 'success')

return redirect(url\_for('dashboard'))

owned\_boards = current\_user.owned\_boards.order\_by(Board.id.desc()).all()

shared\_boards\_list = current\_user.shared\_boards.order\_by(Board.id.desc()).all()

all\_boards\_dict = {board.id: board for board in owned\_boards}

for board in shared\_boards\_list:

if board.id not in all\_boards\_dict:

all\_boards\_dict[board.id] = board

all\_user\_boards = sorted(all\_boards\_dict.values(), key=lambda b: b.id, reverse=True)

return render\_template('dashboard.html', title='Мои доски', form=form, boards=all\_user\_boards)

@app.route('/login', methods=['GET', 'POST'])

def login():

if current\_user.is\_authenticated:

return redirect(url\_for('index'))

form = LoginForm()

if form.validate\_on\_submit():

user = User.query.filter\_by(email=form.email.data).first()

if user is None or not user.check\_password(form.password.data):

flash('Неверный email или пароль.', 'danger')

return redirect(url\_for('login'))

login\_user(user, remember=form.remember\_me.data)

flash(f'Добро пожаловать, {user.username}!', 'success')

next\_page = request.args.get('next')

if not next\_page or not next\_page.startswith('/'):

if current\_user.is\_admin:

next\_page = url\_for('admin\_dashboard')

else:

next\_page = url\_for('dashboard')

return redirect(next\_page)

return render\_template('login.html', title='Вход', form=form)

@app.route('/logout')

def logout():

logout\_user()

flash('Вы успешно вышли из системы.', 'info')

return redirect(url\_for('login'))

@app.route('/register', methods=['GET', 'POST'])

def register():

if current\_user.is\_authenticated:

return redirect(url\_for('index'))

form = RegistrationForm()

if form.validate\_on\_submit():

user = User(username=form.username.data, email=form.email.data)

user.set\_password(form.password.data)

db.session.add(user)

db.session.commit()

flash('Поздравляем, вы успешно зарегистрированы! Теперь вы можете войти.', 'success')

return redirect(url\_for('login'))

return render\_template('register.html', title='Регистрация', form=form)

@app.route('/profile', methods=['GET'])

@login\_required

def profile():

update\_account\_form = UpdateAccountForm(obj=current\_user)

if request.method == 'GET':

update\_account\_form.username.data = current\_user.username

update\_account\_form.email.data = current\_user.email

change\_password\_form = ChangePasswordForm()

update\_avatar\_form = UpdateAvatarForm()

return render\_template('profile.html', title='Мой профиль',

update\_account\_form=update\_account\_form,

change\_password\_form=change\_password\_form,

update\_avatar\_form=update\_avatar\_form)

@app.route('/profile/edit\_account', methods=['POST'])

@login\_required

def edit\_account():

form = UpdateAccountForm()

if form.validate\_on\_submit():

current\_user.username = form.username.data

current\_user.email = form.email.data

db.session.commit()

flash('Данные профиля успешно обновлены.', 'success')

else:

for field, errors in form.errors.items():

for error in errors:

flash(f"Ошибка обновления профиля ({getattr(form, field).label.text}): {error}", 'danger')

return redirect(url\_for('profile'))

@app.route('/profile/change\_password', methods=['POST'])

@login\_required

def change\_password():

form = ChangePasswordForm()

if form.validate\_on\_submit():

current\_user.set\_password(form.new\_password.data)

db.session.commit()

flash('Пароль успешно изменен.', 'success')

else:

for field, errors in form.errors.items():

for error in errors:

if field == 'current\_password' and form.current\_password.errors:

flash(form.current\_password.errors[0], 'danger')

else:

flash(f"Ошибка смены пароля ({getattr(form, field).label.text}): {error}", 'danger')

return redirect(url\_for('profile'))

@app.route('/profile/update\_avatar', methods=['POST'])

@login\_required

def update\_avatar():

form = UpdateAvatarForm()

if form.validate\_on\_submit():

file = form.avatar.data

filename = secure\_filename(file.filename)

\_, f\_ext = os.path.splitext(filename)

if not f\_ext:

flash('Файл должен иметь расширение (например, .jpg, .png).', 'warning')

return redirect(url\_for('profile'))

avatar\_fn = f"user\_{current\_user.id}{f\_ext.lower()}"

if current\_user.avatar\_url and current\_user.avatar\_url != 'default\_avatar.png':

try:

old\_avatar\_path = os.path.join(app.config['UPLOADED\_AVATARS\_DEST'], current\_user.avatar\_url)

if os.path.exists(old\_avatar\_path) and os.path.isfile(old\_avatar\_path):

os.remove(old\_avatar\_path)

except Exception as e:

app.logger.error(f"Error deleting old avatar: {e}")

file\_path = os.path.join(app.config['UPLOADED\_AVATARS\_DEST'], avatar\_fn)

try:

file.save(file\_path)

current\_user.avatar\_url = avatar\_fn

db.session.commit()

flash('Аватар успешно обновлен.', 'success')

except Exception as e:

app.logger.error(f"Error saving avatar: {e}")

flash('Не удалось сохранить аватар. Попробуйте еще раз.', 'danger')

else:

for field, errors in form.errors.items():

for error in errors:

flash(f"Ошибка загрузки аватара: {error}", 'danger')

return redirect(url\_for('profile'))

@app.route('/admin')

@login\_required

@admin\_required

def admin\_dashboard():

users = User.query.order\_by(User.id).all()

return render\_template('admin\_dashboard.html', title='Панель администратора', users=users)

@app.route('/admin/user/<int:user\_id>/edit', methods=['GET', 'POST'])

@login\_required

@admin\_required

def admin\_edit\_user(user\_id):

user\_to\_edit = User.query.get\_or\_404(user\_id)

form = AdminEditUserForm(original\_username=user\_to\_edit.username, obj=user\_to\_edit if request.method == 'GET' else None)

if form.validate\_on\_submit():

if user\_to\_edit == current\_user and not form.is\_admin.data:

admins\_count = User.query.filter\_by(is\_admin=True).count()

if admins\_count <= 1:

flash('Вы не можете снять с себя права администратора, если вы единственный администратор.', 'warning')

return redirect(url\_for('admin\_edit\_user', user\_id=user\_id))

user\_to\_edit.username = form.username.data

user\_to\_edit.is\_admin = form.is\_admin.data

db.session.commit()

flash(f'Данные пользователя {user\_to\_edit.username} обновлены.', 'success')

return redirect(url\_for('admin\_dashboard'))

if request.method == 'GET':

form.username.data = user\_to\_edit.username

form.is\_admin.data = user\_to\_edit.is\_admin

form.email.data = user\_to\_edit.email

return render\_template('admin\_edit\_user.html', title=f'Редактировать пользователя: {user\_to\_edit.username}',

form=form, user\_to\_edit=user\_to\_edit)

@app.route('/admin/user/<int:user\_id>/delete', methods=['POST'])

@login\_required

@admin\_required

def admin\_delete\_user(user\_id):

user\_to\_delete = User.query.get\_or\_404(user\_id)

if user\_to\_delete == current\_user:

flash('Вы не можете удалить свой собственный аккаунт из панели администратора.', 'danger')

return redirect(url\_for('admin\_dashboard'))

if user\_to\_delete.owned\_boards.count() > 0:

flash(f'Нельзя удалить пользователя {user\_to\_delete.username}, так как он владеет досками. Сначала удалите или переназначьте его доски.', 'warning')

return redirect(url\_for('admin\_dashboard'))

username = user\_to\_delete.username

db.session.delete(user\_to\_delete)

db.session.commit()

flash(f'Пользователь {username} удален.', 'success')

return redirect(url\_for('admin\_dashboard'))

@app.route('/boards/<int:board\_id>/edit', methods=['GET', 'POST'])

@login\_required

def edit\_board(board\_id):

board = Board.query.get\_or\_404(board\_id)

if not current\_user.can\_delete\_board(board):

flash('У вас нет прав для редактирования этой доски.', 'danger')

return redirect(url\_for('view\_board', board\_id=board.id))

form = BoardForm(obj=board if request.method == 'GET' else None)

invite\_form = InviteUserForm()

if form.validate\_on\_submit() and 'submit\_board\_name' in request.form:

board.name = form.name.data

db.session.commit()

flash('Название доски успешно обновлено!', 'success')

return redirect(url\_for('edit\_board', board\_id=board.id))

if request.method == 'GET':

form.name.data = board.name

board\_members\_list = board.members.all()

is\_owner = (current\_user == board.owner)

return render\_template('edit\_board.html', title=f'Настройки доски: {board.name}',

form=form, invite\_form=invite\_form, board=board,

board\_members=board\_members\_list, is\_owner=is\_owner)

@app.route('/boards/<int:board\_id>/delete', methods=['POST'])

@login\_required

def delete\_board(board\_id):

board\_to\_delete = Board.query.get\_or\_404(board\_id)

if not current\_user.can\_delete\_board(board\_to\_delete):

flash('У вас нет прав для удаления этой доски.', 'danger')

return redirect(url\_for('dashboard'))

board\_name = board\_to\_delete.name

db.session.delete(board\_to\_delete)

db.session.commit()

flash(f'Доска "{board\_name}" удалена.', 'success')

return redirect(url\_for('dashboard'))

@app.route('/boards/<int:board\_id>', methods=['GET', 'POST'])

@app.route('/boards/<int:board\_id>/cards/<int:card\_id\_in\_url>', methods=['GET'])

@login\_required

def view\_board(board\_id, card\_id\_in\_url=None):

board = Board.query.get\_or\_404(board\_id)

if not current\_user.can\_edit\_board(board):

flash('У вас нет доступа к этой доске.', 'danger')

return redirect(url\_for('dashboard'))

column\_form = ColumnForm()

card\_form = CardForm()

comment\_form = CommentForm()

tag\_form = TagForm()

\_populate\_assignee\_choices(card\_form, board)

\_populate\_tag\_choices(card\_form, board)

if request.method == 'POST':

if column\_form.validate\_on\_submit() and 'submit\_column' in request.form:

last\_column = Column.query.filter\_by(board\_id=board.id).order\_by(Column.position.desc()).first()

new\_position = (last\_column.position + 1) if last\_column else 0

new\_column = Column(name=column\_form.name.data, board\_id=board.id, position=new\_position)

db.session.add(new\_column)

db.session.commit()

flash(f'Колонка "{new\_column.name}" добавлена.', 'success')

redirect\_url = url\_for('view\_board', board\_id=board.id, card\_id\_in\_url=card\_id\_in\_url) if card\_id\_in\_url else url\_for('view\_board', board\_id=board.id)

return redirect(redirect\_url)

columns = board.columns

card\_to\_open = None

if card\_id\_in\_url:

card\_to\_open\_query = Card.query.join(Column).filter(Card.id == card\_id\_in\_url, Column.board\_id == board\_id)

card\_to\_open = card\_to\_open\_query.first()

if not card\_to\_open:

flash(f'Карточка с ID {card\_id\_in\_url} не найдена на этой доске.', 'warning')

return redirect(url\_for('view\_board', board\_id=board\_id))

board\_all\_users = board.get\_eligible\_assignees()

board\_all\_tags = board.tags.order\_by(Tag.name).all()

return render\_template('board.html', title=f"{board.name}", board=board,

columns=columns, column\_form=column\_form, card\_form=card\_form,

comment\_form=comment\_form, tag\_form=tag\_form,

card\_id\_to\_open\_on\_load=card\_to\_open.id if card\_to\_open else None,

board\_all\_users=board\_all\_users, board\_all\_tags=board\_all\_tags)

@app.route('/boards/<int:board\_id>/invite', methods=['POST'])

@login\_required

def invite\_to\_board(board\_id):

board = Board.query.get\_or\_404(board\_id)

if not current\_user.can\_delete\_board(board):

flash('Только владелец доски может приглашать участников.', 'danger')

return redirect(url\_for('edit\_board', board\_id=board.id, \_anchor='members-management'))

invite\_form = InviteUserForm()

if invite\_form.validate\_on\_submit():

identifier = invite\_form.email\_or\_username.data

user\_to\_invite = User.query.filter(or\_(User.email == identifier, User.username == identifier)).first()

if not user\_to\_invite:

flash(f'Пользователь с email/именем "{identifier}" не найден.', 'warning')

elif user\_to\_invite == board.owner:

flash('Владелец уже имеет полный доступ к доске.', 'info')

elif user\_to\_invite in board.members.all():

flash(f'Пользователь "{user\_to\_invite.username}" уже является участником этой доски.', 'info')

else:

board.members.append(user\_to\_invite)

db.session.commit()

flash(f'Пользователь "{user\_to\_invite.username}" успешно приглашен на доску "{board.name}".', 'success')

else:

for field, errors in invite\_form.errors.items():

for error in errors:

flash(f"Ошибка в поле приглашения: {error}", 'danger')

return redirect(url\_for('edit\_board', board\_id=board.id, \_anchor='members-management'))

@app.route('/boards/<int:board\_id>/members/<int:user\_id>/remove', methods=['POST'])

@login\_required

def remove\_from\_board(board\_id, user\_id):

board = Board.query.get\_or\_404(board\_id)

user\_to\_remove = User.query.get\_or\_404(user\_id)

can\_remove = False

if current\_user.can\_delete\_board(board):

if user\_to\_remove != current\_user:

can\_remove = True

else:

flash('Владелец не может удалить себя из участников через эту форму. Передайте права или удалите доску.', 'warning')

elif current\_user == user\_to\_remove:

if user\_to\_remove != board.owner:

can\_remove = True

else:

flash('У вас нет прав для удаления этого участника.', 'danger')

return redirect(url\_for('edit\_board', board\_id=board.id, \_anchor='members-management'))

if can\_remove:

if user\_to\_remove in board.members:

board.members.remove(user\_to\_remove)

for column in board.columns:

for card in column.cards:

if user\_to\_remove in card.assignees:

card.assignees.remove(user\_to\_remove)

db.session.commit()

flash(f'Пользователь "{user\_to\_remove.username}" удален с доски "{board.name}".', 'success')

elif user\_to\_remove == board.owner:

flash(f'Пользователь "{user\_to\_remove.username}" является владельцем и не может быть удален как участник.', 'info')

else:

flash(f'Пользователь "{user\_to\_remove.username}" не найден среди участников этой доски.', 'info')

return redirect(url\_for('edit\_board', board\_id=board.id, \_anchor='members-management'))

@app.route('/columns/<int:column\_id>/edit', methods=['GET', 'POST'])

@login\_required

def edit\_column(column\_id):

column = Column.query.get\_or\_404(column\_id)

board = column.board

if not current\_user.can\_edit\_board(board):

flash('Нет прав для редактирования элементов этой доски.', 'danger')

return redirect(url\_for('view\_board', board\_id=board.id))

form = ColumnForm(obj=column)

if form.validate\_on\_submit():

column.name = form.name.data

db.session.commit()

flash('Название колонки обновлено.', 'success')

return redirect(url\_for('view\_board', board\_id=board.id))

current\_name = column.name

return render\_template('edit\_column.html', title='Редактировать колонку', form=form, column\_id=column\_id, board\_id=board.id, current\_name=current\_name)

@app.route('/columns/<int:column\_id>/delete', methods=['POST'])

@login\_required

def delete\_column(column\_id):

column\_to\_delete = Column.query.get\_or\_404(column\_id)

board = column\_to\_delete.board

if not current\_user.can\_edit\_board(board):

flash('У вас нет прав для удаления элементов этой доски.', 'danger')

return redirect(url\_for('view\_board', board\_id=board.id))

column\_name = column\_to\_delete.name

db.session.delete(column\_to\_delete)

db.session.commit()

flash(f'Колонка "{column\_name}" удалена.', 'success')

return redirect(url\_for('view\_board', board\_id=board.id))

@app.route('/columns/<int:column\_id>/cards/create', methods=['POST'])

@login\_required

def create\_card(column\_id):

column = Column.query.get\_or\_404(column\_id)

board = column.board

if not current\_user.can\_edit\_board(board):

flash('У вас нет прав для добавления карточек на эту доску.', 'danger')

return redirect(url\_for('view\_board', board\_id=board.id, \_anchor=f'column-{column.id}'))

card\_form = CardForm(request.form)

\_populate\_assignee\_choices(card\_form, board)

\_populate\_tag\_choices(card\_form, board)

if card\_form.validate\_on\_submit():

new\_card = Card(

title=card\_form.title.data,

description=card\_form.description.data,

column\_id=column.id

)

selected\_assignee\_ids = card\_form.assignees.data

if selected\_assignee\_ids:

assignees\_to\_add = User.query.filter(User.id.in\_(selected\_assignee\_ids)).all()

for user in assignees\_to\_add:

new\_card.assignees.append(user)

selected\_tag\_ids = card\_form.tags.data

if selected\_tag\_ids:

tags\_to\_add = Tag.query.filter(Tag.id.in\_(selected\_tag\_ids), Tag.board\_id == board.id).all()

for tag in tags\_to\_add:

new\_card.tags.append(tag)

db.session.add(new\_card)

db.session.commit()

flash(f'Карточка "{new\_card.title}" добавлена в колонку "{column.name}".', 'success')

else:

for field\_name, errors in card\_form.errors.items():

field\_label = field\_name

try:

field\_label = getattr(card\_form, field\_name).label.text

except AttributeError:

pass

for error in errors:

flash(f"Ошибка в поле '{field\_label}': {error} (для колонки {column.name})", 'danger')

return redirect(url\_for('view\_board', board\_id=board.id, \_anchor=f'column-{column.id}'))

@app.route('/cards/<int:card\_id>/edit', methods=['GET', 'POST'])

@login\_required

def edit\_card(card\_id):

card = Card.query.get\_or\_404(card\_id)

column = card.column

board = column.board

if not current\_user.can\_edit\_board(board):

app.logger.warning(f"User {current\_user.id} tried to edit card {card\_id} without permission.")

if request.headers.get('X-Requested-With') == 'XMLHttpRequest':

return jsonify(success=False, error='Нет прав для редактирования этой карточки.'), 403

flash('Нет прав для редактирования карточек на этой доске.', 'danger')

return redirect(url\_for('view\_board', board\_id=board.id))

form = CardForm(request.form if request.method == 'POST' else None)

\_populate\_assignee\_choices(form, board) # Populates form.assignees.choices

\_populate\_tag\_choices(form, board) # Populates form.tags.choices

if request.method == 'GET':

form.title.data = card.title

form.description.data = card.description

form.assignees.data = [assignee.id for assignee in card.assignees.all()]

form.tags.data = [tag.id for tag in card.tags.all()]

if request.headers.get('X-Requested-With') == 'XMLHttpRequest':

current\_card\_assignees\_data = [{'id': u.id, 'username': u.username, 'avatar\_url': u.get\_avatar()} for u in card.assignees.all()]

current\_card\_tags\_data = [{'id': t.id, 'name': t.name, 'color': t.color} for t in card.tags.all()]

all\_board\_tags\_data = [{'id': t.id, 'name': t.name, 'color': t.color} for t in board.tags.order\_by(Tag.name).all()]

# Prepare full list of eligible assignees for the board (for checkboxes in modal)

all\_board\_assignees\_for\_modal = []

for user\_id, username in form.assignees.choices:

user\_obj = User.query.get(user\_id)

if user\_obj:

all\_board\_assignees\_for\_modal.append({

'id': user\_obj.id,

'username': user\_obj.username,

'avatar\_url': user\_obj.get\_avatar()

})

return jsonify(

success=True,

card={

'id': card.id,

'title': card.title,

'description': card.description or "",

'assignees': current\_card\_assignees\_data, # Current assignees of THIS card

'assignee\_ids': [u.id for u in card.assignees.all()],

'tags': current\_card\_tags\_data, # Current tags of THIS card

'tag\_ids': [t.id for t in card.tags.all()]

},

all\_board\_assignees=all\_board\_assignees\_for\_modal, # All eligible assignees for the board

all\_board\_tags=all\_board\_tags\_data # All tags for the board

)

else:

flash('Для редактирования карточек используется модальное окно.', 'info')

return redirect(url\_for('view\_board', board\_id=board.id, card\_id\_in\_url=card.id))

if request.method == 'POST':

# Form data for assignees and tags will be submitted based on the hidden SelectMultipleFields

# which should be updated by JS from the checkboxes.

if form.validate\_on\_submit():

try:

card.title = form.title.data

card.description = form.description.data

new\_assignee\_ids = set(form.assignees.data if form.assignees.data else [])

current\_assignee\_ids\_on\_card = {user.id for user in card.assignees}

ids\_to\_remove\_from\_card = current\_assignee\_ids\_on\_card - new\_assignee\_ids

if ids\_to\_remove\_from\_card:

users\_to\_remove = User.query.filter(User.id.in\_(ids\_to\_remove\_from\_card)).all()

for user in users\_to\_remove:

if user in card.assignees:

card.assignees.remove(user)

ids\_to\_add\_to\_card = new\_assignee\_ids - current\_assignee\_ids\_on\_card

if ids\_to\_add\_to\_card:

users\_to\_add = User.query.filter(User.id.in\_(ids\_to\_add\_to\_card)).all()

for user in users\_to\_add:

if user not in card.assignees:

card.assignees.append(user)

new\_tag\_ids = set(form.tags.data if form.tags.data else [])

current\_tag\_ids\_on\_card = {tag.id for tag in card.tags}

tag\_ids\_to\_remove = current\_tag\_ids\_on\_card - new\_tag\_ids

if tag\_ids\_to\_remove:

tags\_to\_remove\_objs = Tag.query.filter(Tag.id.in\_(tag\_ids\_to\_remove)).all()

for tag\_obj in tags\_to\_remove\_objs:

if tag\_obj in card.tags:

card.tags.remove(tag\_obj)

tag\_ids\_to\_add = new\_tag\_ids - current\_tag\_ids\_on\_card

if tag\_ids\_to\_add:

tags\_to\_add\_objs = Tag.query.filter(Tag.id.in\_(tag\_ids\_to\_add), Tag.board\_id == board.id).all()

for tag\_obj in tags\_to\_add\_objs:

if tag\_obj not in card.tags:

card.tags.append(tag\_obj)

db.session.commit()

app.logger.info(f"Card {card.id} updated successfully by user {current\_user.id}.")

if request.headers.get('X-Requested-With') == 'XMLHttpRequest':

assignees\_data = [{'id': u.id, 'username': u.username, 'avatar\_url': u.get\_avatar()} for u in card.assignees.all()]

tags\_data = [{'id': t.id, 'name': t.name, 'color': t.color} for t in card.tags.all()]

return jsonify(success=True, card={

'id': card.id,

'title': card.title,

'description': card.description or "",

'assignees': assignees\_data,

'assignee\_ids': [u.id for u in card.assignees.all()],

'tags': tags\_data,

'tag\_ids': [t.id for t in card.tags.all()]

})

else:

flash('Карточка обновлена.', 'success')

return redirect(url\_for('view\_board', board\_id=board.id))

except Exception as e:

db.session.rollback()

app.logger.error(f"Error updating card {card.id}: {e}", exc\_info=True)

if request.headers.get('X-Requested-With') == 'XMLHttpRequest':

return jsonify(success=False, error="Внутренняя ошибка сервера при обновлении карточки."), 500

else:

flash('Ошибка при обновлении карточки. Пожалуйста, попробуйте снова.', 'danger')

return redirect(url\_for('view\_board', board\_id=board.id, card\_id\_in\_url=card.id))

else:

app.logger.warning(f"Card {card.id} update form validation failed: {form.errors}")

if request.headers.get('X-Requested-With') == 'XMLHttpRequest':

errors = {field: error[0] for field, error in form.errors.items()}

return jsonify(success=False, errors=errors), 400

else:

flash('Пожалуйста, исправьте ошибки в форме.', 'danger')

return redirect(url\_for('view\_board', board\_id=board.id, card\_id\_in\_url=card.id))

app.logger.error(f"Unexpected state in edit\_card for card {card\_id}, method: {request.method}")

return redirect(url\_for('view\_board', board\_id=board.id))

@app.route('/cards/<int:card\_id>/delete', methods=['POST'])

@login\_required

def delete\_card(card\_id):

card\_to\_delete = Card.query.get\_or\_404(card\_id)

column = card\_to\_delete.column

board = column.board

if not current\_user.can\_edit\_board(board):

flash('У вас нет прав для удаления карточек на этой доске.', 'danger')

if request.headers.get('X-Requested-With') == 'XMLHttpRequest':

return jsonify(success=False, error='Нет прав'), 403

return redirect(url\_for('view\_board', board\_id=board.id))

card\_title = card\_to\_delete.title

db.session.delete(card\_to\_delete)

db.session.commit()

if request.headers.get('X-Requested-With') == 'XMLHttpRequest':

return jsonify(success=True, message=f'Карточка "{card\_title}" удалена.')

flash(f'Карточка "{card\_title}" удалена.', 'success')

return redirect(url\_for('view\_board', board\_id=board.id))

@app.route('/api/cards/<int:card\_id>/move', methods=['POST'])

@login\_required

def move\_card(card\_id):

card = Card.query.get\_or\_404(card\_id)

board = card.column.board

if not current\_user.can\_edit\_board(board):

return jsonify(success=False, error="Нет прав для изменения этой карточки."), 403

data = request.get\_json()

if not data or 'new\_column\_id' not in data:

return jsonify(success=False, error="Отсутствует ID новой колонки."), 400

new\_column\_id = data.get('new\_column\_id')

new\_column = Column.query.get(new\_column\_id)

if not new\_column or new\_column.board\_id != board.id:

return jsonify(success=False, error="Некорректный ID новой колонки."), 400

if card.column\_id != new\_column\_id:

card.column\_id = new\_column\_id

db.session.commit()

return jsonify(success=True, message="Карточка перемещена.")

return jsonify(success=True, message="Карточка осталась в той же колонке.")

# --- Маршруты для комментариев (AJAX) ---

@app.route('/cards/<int:card\_id>/comments', methods=['GET'])

@login\_required

def get\_comments(card\_id):

card = Card.query.get\_or\_404(card\_id)

if not current\_user.can\_edit\_board(card.column.board):

return jsonify(success=False, error="Нет доступа к комментариям этой карточки."), 403

comments = card.comments.order\_by(Comment.timestamp.asc()).all()

comments\_data = []

for comment in comments:

comments\_data.append({

'id': comment.id,

'text': comment.text,

'timestamp': comment.timestamp.strftime('%d.%m.%Y %H:%M'),

'author': {

'id': comment.author.id,

'username': comment.author.username,

'avatar\_url': comment.author.get\_avatar()

},

'can\_edit': current\_user.id == comment.author.id,

'can\_delete': current\_user.id == comment.author.id

})

return jsonify(success=True, comments=comments\_data)

@app.route('/cards/<int:card\_id>/comments/add', methods=['POST'])

@login\_required

def add\_comment(card\_id):

card = Card.query.get\_or\_404(card\_id)

if not current\_user.can\_edit\_board(card.column.board):

return jsonify(success=False, error="Вы не можете комментировать на этой доске."), 403

form = CommentForm()

if form.validate\_on\_submit():

comment = Comment(text=form.text.data, author=current\_user, card\_id=card.id)

db.session.add(comment)

db.session.commit()

return jsonify(success=True, comment={

'id': comment.id,

'text': comment.text,

'timestamp': comment.timestamp.strftime('%d.%m.%Y %H:%M'),

'author': {

'id': current\_user.id,

'username': current\_user.username,

'avatar\_url': current\_user.get\_avatar()

},

'can\_edit': True,

'can\_delete': True

}), 201

errors = {field: error[0] for field, error in form.errors.items()}

return jsonify(success=False, errors=errors), 400

@app.route('/comments/<int:comment\_id>/edit', methods=['POST'])

@login\_required

def edit\_comment(comment\_id):

comment = Comment.query.get\_or\_404(comment\_id)

if comment.author != current\_user:

return jsonify(success=False, error="Вы не можете редактировать этот комментарий."), 403

form = CommentForm()

if form.validate\_on\_submit():

comment.text = form.text.data

comment.timestamp = datetime.utcnow()

db.session.commit()

return jsonify(success=True, comment={

'id': comment.id,

'text': comment.text,

'timestamp': comment.timestamp.strftime('%d.%m.%Y %H:%M'),

'author': {

'id': comment.author.id,

'username': comment.author.username,

'avatar\_url': comment.author.get\_avatar()

},

'can\_edit': True,

'can\_delete': True

})

errors = {field: error[0] for field, error in form.errors.items()}

return jsonify(success=False, errors=errors), 400

@app.route('/comments/<int:comment\_id>/delete', methods=['POST'])

@login\_required

def delete\_comment(comment\_id):

comment = Comment.query.get\_or\_404(comment\_id)

if comment.author != current\_user:

return jsonify(success=False, error="Вы не можете удалить этот комментарий."), 403

db.session.delete(comment)

db.session.commit()

return jsonify(success=True, message="Комментарий удален.")

# --- Маршруты для ТЕГОВ (AJAX) ---

@app.route('/api/boards/<int:board\_id>/tags', methods=['GET'])

@login\_required

def get\_board\_tags(board\_id):

board = Board.query.get\_or\_404(board\_id)

if not current\_user.can\_edit\_board(board):

return jsonify(success=False, error="Нет доступа к тегам этой доски."), 403

tags = board.tags.order\_by(Tag.name).all()

tags\_data = [{'id': tag.id, 'name': tag.name, 'color': tag.color} for tag in tags]

return jsonify(success=True, tags=tags\_data)

@app.route('/api/boards/<int:board\_id>/tags/create', methods=['POST'])

@login\_required

def create\_tag\_for\_board(board\_id):

board = Board.query.get\_or\_404(board\_id)

if not current\_user.can\_edit\_board(board):

return jsonify(success=False, error="Нет прав для создания тегов на этой доске."), 403

form = TagForm()

if form.validate\_on\_submit():

try:

new\_tag = Tag(name=form.name.data, color=form.color.data, board\_id=board.id)

db.session.add(new\_tag)

db.session.commit()

return jsonify(success=True, tag={'id': new\_tag.id, 'name': new\_tag.name, 'color': new\_tag.color}), 201

except exc.IntegrityError:

db.session.rollback()

return jsonify(success=False, errors={'name': f'Тег с именем "{form.name.data}" уже существует на этой доске.'}), 400

except Exception as e:

db.session.rollback()

app.logger.error(f"Error creating tag for board {board\_id}: {e}")

return jsonify(success=False, error="Внутренняя ошибка сервера при создании тега."), 500

errors = {field: error[0] for field, error in form.errors.items()}

return jsonify(success=False, errors=errors), 400

@app.route('/api/tags/<int:tag\_id>/edit', methods=['POST'])

@login\_required

def edit\_tag(tag\_id):

tag\_to\_edit = Tag.query.get\_or\_404(tag\_id)

board = tag\_to\_edit.board

if not current\_user.can\_edit\_board(board):

return jsonify(success=False, error="Нет прав для редактирования тегов этой доски."), 403

form = TagForm()

if form.validate\_on\_submit():

try:

if tag\_to\_edit.name != form.name.data:

existing\_tag = Tag.query.filter(Tag.name == form.name.data, Tag.board\_id == board.id, Tag.id != tag\_id).first()

if existing\_tag:

return jsonify(success=False, errors={'name': f'Тег с именем "{form.name.data}" уже существует на этой доске.'}), 400

tag\_to\_edit.name = form.name.data

tag\_to\_edit.color = form.color.data

db.session.commit()

return jsonify(success=True, tag={'id': tag\_to\_edit.id, 'name': tag\_to\_edit.name, 'color': tag\_to\_edit.color})

except exc.IntegrityError:

db.session.rollback()

return jsonify(success=False, errors={'name': f'Тег с именем "{form.name.data}" уже существует (ошибка Integrity).'}), 400

except Exception as e:

db.session.rollback()

app.logger.error(f"Error editing tag {tag\_id}: {e}")

return jsonify(success=False, error="Внутренняя ошибка сервера при редактировании тега."), 500

errors = {field: error[0] for field, error in form.errors.items()}

return jsonify(success=False, errors=errors), 400

@app.route('/api/tags/<int:tag\_id>/delete', methods=['POST'])

@login\_required

def delete\_tag(tag\_id):

tag\_to\_delete = Tag.query.get\_or\_404(tag\_id)

board = tag\_to\_delete.board

if not current\_user.can\_edit\_board(board):

return jsonify(success=False, error="Нет прав для удаления тегов этой доски."), 403

try:

tag\_name = tag\_to\_delete.name

db.session.delete(tag\_to\_delete)

db.session.commit()

return jsonify(success=True, message=f'Тег "{tag\_name}" удален.')

except Exception as e:

db.session.rollback()

app.logger.error(f"Error deleting tag {tag\_id}: {e}")

return jsonify(success=False, error="Внутренняя ошибка сервера при удалении тега."), 500