

**МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ П.П. СЕМЕНОВА-ТЯН-ШАНСКОГО»
(ЛГПУ имени П.П. Семенова-Тян-Шанского)**

Институт естественных, математических и технических наук

Кафедра информатики, информационных технологий и защиты информации

Волков Александр Сергеевич

**«Разработка сервиса по обработке заявок в службу технической поддержки
отдела ИТ»
(курсовая работа)**

выполнена в рамках изучения дисциплины
«Веб программирование»

Направление подготовки: 09.03.01 Информатика и вычислительная техника
Профиль: «Интеллектуальные системы обработки информации и
управление»

Научный руководитель:
Кандидат технических
наук, доцент, заведующий
кафедрой ИИТиЗИ.
Скуднев Д. М.

Липецк – 2023 г.

Оглавление

Введение.....	3
Глава 1. Web Технологии	4
1.1 Язык разметки HTML	4
1.2 Каскадная таблица стилей CSS3	5
1.3 JavaScript универсальный язык	7
1.4 Фреймворк JavaScript – React	8
Глава 2 проектирование программного продукта	9
2.1 проектирование программного продукта	9
Глава 3 Реализация программного продукта	10
3.1 описание процесса разработки программного продукта	10
3.2 написание инструкций использования программного продукта.....	11
Заключение.....	13
Список используемой литературы	14
Приложение 1.....	15
Приложение 2.....	17
Приложение 3.....	20
Приложение 4.....	23

Введение

В современном бизнесе, особенно в сфере обслуживания, все больше возрастает необходимость в эффективной обработке обращений в службу технической поддержки. С развитием информационных технологий, растущими требованиями пользователей и увеличивающимся объемом обращений, компаниям необходимо найти новые подходы к управлению и обработке запросов.

Целью данной курсовой работы является разработка сервиса по обработке обращений в службу технической поддержки, который поможет организациям улучшить качество обслуживания клиентов, повысить эффективность работы сотрудников и сократить время ответа на запросы.

Для достижения данной цели необходимо выполнить следующие задачи:

1. Изучить существующие методы и подходы к обработке обращений в службу технической поддержки.
2. Проанализировать требования пользователей и выявить основные проблемы, с которыми они сталкиваются при обращении в службу технической поддержки.
3. Разработать архитектуру и функциональные возможности сервиса, учитывая требования пользователей и особенности организации.
4. Реализовать сервис, используя современные технологии и методы разработки программного обеспечения.
5. Провести тестирование и оценку эффективности разработанного сервиса путем сравнения показателей до и после внедрения

Результаты данной работы могут быть полезны для организаций, которые стремятся улучшить свою службу технической поддержки и обеспечить более эффективную обработку обращений. Кроме того, разработанный сервис может послужить основой для дальнейших исследований и разработок в области улучшения качества обслуживания клиентов и оптимизации работы службы технической поддержки.

Глава 1. Web Технологии

1.1 Язык разметки HTML

HTML (HyperText Markup Language) - это язык разметки, используемый для создания структуры и представления содержимого веб-страниц. Вот основные характеристики HTML:

1. Структура документа: HTML определяет структуру документа с помощью различных элементов и тегов. Каждый элемент представляет собой различные части документа, такие как заголовки, параграфы, списки, таблицы и другие.

2. Теги: HTML использует теги для обозначения различных элементов и их свойств. Теги заключают содержимое элемента и определяют его тип и функцию. Например, `<h1>` используется для заголовков первого уровня.

3. Семантика: HTML предоставляет семантические элементы, которые помогают определить смысл и значение содержимого. Это позволяет поисковым системам и другим инструментам лучше понимать структуру страницы и ее содержимое. Например, `<header>`, `<nav>`, `<article>`, `<section>`, `<footer>` - это некоторые из семантических элементов HTML.

HTML является основным языком разметки для создания веб-страниц. Он используется в сочетании с CSS (Cascade Style Sheets) для стилизации и внешнего вида страницы, а также с JavaScript для добавления интерактивности и динамического поведения. HTML-документы интерпретируются веб-браузерами, которые отображают содержимое и структуру страницы для пользователей.

HTML широко используется для создания веб-сайтов, веб-приложений и электронных документов. Он является стандартом для разработки веб-контента и поддерживается всеми современными веб-браузерами. HTML постоянно развивается, и последние версии, такие как HTML5, включают новые возможности и элементы для создания более интерактивных и мощных веб-приложений.

1.2 Каскадная таблица стилей CSS3

CSS (Cascading Style Sheets) - это язык стилей, который был создан для оформления и визуального представления веб-страниц. Вот некоторые из особенностей и применений CSS:

1. Каскадность и наследование: Стили в CSS применяются по принципу каскадности, что означает, что браузер применяет стили относительно приоритета селекторов и специфичности правил. Это позволяет легко переопределять стили и создавать гибкие иерархии стилей. Кроме того, некоторые свойства CSS наследуются от родительских элементов, что делает применение стилей более эффективным.

2. Адаптивность и отзывчивость: CSS предоставляет мощные возможности для создания адаптивных и отзывчивых веб-страниц, которые автоматически адаптируются к различным экранам и устройствам. С помощью медиа-запросов и других техник CSS можно создавать различные макеты, изменять размеры и расположение элементов в зависимости от размера экрана.

3. Модульность и повторное использование: CSS поддерживает концепцию модульности, позволяя разработчикам создавать отдельные файлы стилей для разных компонентов и модулей веб-страницы. Это делает стили более организованными, удобными для поддержки и повторного использования в других проектах.

CSS широко используется в веб-разработке для стилизации и оформления веб-страниц, включая элементы, такие как текст, фоны, границы, размеры, расположение, цвета и т.д. Он применяется на всех типах веб-сайтов, включая персональные блоги, корпоративные сайты, интернет-магазины, новостные порталы и другие онлайн-проекты. CSS также используется в современных веб-фреймворках и библиотеках, таких как Bootstrap, Foundation, Material-UI и др., которые предоставляют готовые стилевые компоненты и шаблоны для ускорения процесса разработки веб-

приложений.

1.3 JavaScript универсальный язык

JavaScript - это интерпретируемый язык программирования, который был создан для придания интерактивности и динамичности веб-страницам. Вот некоторые из особенностей и применений JavaScript:

1. Манипуляция DOM: JavaScript предоставляет возможность манипулировать структурой и содержимым веб-страницы с помощью DOM (Document Object Model). Это означает, что JavaScript может создавать, изменять и удалять элементы на странице, изменять их свойства и стили, а также реагировать на события, происходящие с элементами.

2. Обработка данных: JavaScript позволяет обрабатывать данные, выполнять математические операции, работать с текстом, массивами и объектами. Он поддерживает различные структуры данных и операторы, что делает его мощным инструментом для обработки и манипуляции данными на веб-странице.

3. Асинхронность: JavaScript поддерживает асинхронное выполнение кода с помощью функций обратного вызова (callback), промисов (Promise) и асинхронных функций (async/await). Это позволяет выполнять задачи, которые требуют времени, такие как загрузка данных с сервера или обращение к внешним API, без блокировки интерфейса пользователя.

4. Модульность: JavaScript поддерживает модульную организацию кода, что позволяет разделять его на отдельные модули и переиспользовать их в разных частях проекта.

JavaScript широко используется в веб-разработке для создания динамических веб-страниц, веб-приложений и игр. Он также используется в серверной разработке с помощью платформы Node.js, что позволяет разрабатывать полноценные веб-серверы и приложения, работающие на стороне сервера. JavaScript также может быть использован для разработки мобильных приложений с использованием фреймворков и инструментов, таких как React Native и Ionic.

1.4 Фреймворк JavaScript – React

React — это библиотека JavaScript для построения пользовательских интерфейсов, разработанная командой Facebook в 2013 году. React был создан, чтобы облегчить процесс разработки сложных, интерактивных веб-интерфейсов, особенно там, где требуется обработка большого количества пользовательских данных в реальном времени.

Основные особенности React:

1. Компонентный подход: React построен на концепции компонент, которые могут быть повторно использованы и комбинированы для построения сложных интерфейсов. Компоненты могут содержать собственное состояние и логику, и их можно вкладывать друг в друга, создавая иерархию.

2. Виртуальный DOM: React использует концепцию "виртуального DOM", что позволяет библиотеке оптимизировать обновления и рендеринг в реальном DOM браузера. Это существенно улучшает производительность, особенно при работе с большими объемами данных.

3. Однонаправленный поток данных: React следует принципу "однонаправленного потока данных" или "нисходящего потока данных", что делает структуру приложения более понятной и предсказуемой.

4. JSX: React использует JSX, синтаксис, схожий с HTML, внутри JavaScript. JSX обеспечивает простой и интуитивно понятный способ создания и описания компонентов.

5. Состояние компонента: В React каждый компонент может иметь свое собственное "состояние", которое можно изменять в ответ на пользовательские действия или другие изменения.

React используется во многих крупных веб-приложениях, включая Facebook и Instagram. Кроме того, с помощью React Native можно создавать мобильные приложения на React, которые работают как на iOS, так и на Android.

Глава 2 проектирование программного продукта

2.1 проектирование программного продукта

Процесс разработки сервиса для обработки заявок в службу технической поддержки включает в себя несколько этапов, которые обеспечивают эффективную обработку запросов пользователей и предоставление эффективной поддержки.

Первым этапом является создание логики сайта. Для этого будет использоваться БД встроенная в каждый браузер и это localStorage. Так как это тестовая версия можно будет остановиться на ней, ведь для подключения React к БД таким как MySQL, MongoDB и так далее, нужно разворачивать локальный сервер, что неудобно при тестировании на разных ПК, либо аренда виртуального сервера.

В модели моей БД используются такие поля как:

1. id – индивидуальный номер запроса. Тип данных – целые числа.
2. date – дата поста. Тип данных – строки. Дата и время подставляются автоматически системой.
3. fio – ФИО кто задает вопрос. Тип данных – текст.
4. post – Кто пишет Сотрудник/Пользователь. Тип данных – текст.
5. text – Текст проблемы. Тип данных – текст. Ограничен 10 символами.
6. active – Решено или нет. Тип данных – Boolean. По умолчанию значение false (0), при решении меняется на true (1).

Вторым этапом идет проектирование разделов. На данном этапе

Глава 3 Реализация программного продукта

3.1 описание процесса разработки программного продукта

Для начала делим экран на навигационную шапку и окно действий. Далее создаем 3 компонента SupportPage.js будет страница, где нужно оставлять тикет, LoginPage.js эта страница позволит нам пропускать к разбору тикетов только сотрудников и поэтому 3 компонент EmployeePage.js будет доступен только сотрудникам.

В SupportPage.js создаем форму с полями ФИО, Должность, Описать ошибку. Тут же создаем localStorage, useEffect, useState.

useState – нам нужен будет для того чтобы держать состояние отправленного тикета. Далее это состояние идет в useEffect где проверяется что оно не пустое. Создает новую переменную для сохранения старых данных из localStorage и потом делает слияние нового состояния и старых данных. Далее он отправляет это обратно в localStorage под ключом “items”.

В LoginPage.js создаем форму с 3 элементами:

1. Input – имя пользователя.
2. Input – пароль.
3. Button – кнопка с надписью вход.

Тут же опять у нас нету БД поэтому мы локально задаем пароль и логин. Пусть будет (password / volkov). Теперь при нажатии на вход у нас будет сравниваться логин и пароль заданный в if и введенный в input. Если все верно показываем содержимое в EmployeePage.js.

В EmployeePage.js создаем 3 кнопки с фильтрами и прописываем компонент как должна выводиться информация в 1 тикете. А именно:

1. Время
2. От кого
3. Должность
4. Текст
5. Статус

6. А также 2 кнопки “Решено/Отменить решение”, “Удалить задачу”

3.2 написание инструкций использования программного продукта

Шаг 1: Установка зависимостей

1. Убедитесь, что у вас установлен Node.js на вашем компьютере.
2. Склонируйте репозиторий проекта или загрузите его себе на компьютер.
3. Откройте терминал и перейдите в директорию проекта.
4. Выполните команду `npm install` для установки всех зависимостей проекта.

Шаг 2: Запуск приложения

1. В терминале выполните команду `npm start` для запуска приложения.
2. Приложение будет доступно по адресу `http://localhost:3000` в вашем веб-браузере.

Шаг 3: Создание тикета

1. На главной странице приложения нажмите кнопку "Создать тикет".
2. Заполните форму с необходимыми данными, такими как ФИО, должность и описание ошибки/вопроса.
3. Нажмите кнопку "Отправить" для создания тикета.

Шаг 4: Разбор тикетов

1. На главной странице приложения выберите вкладку "Разбор тикетов".
2. Вы увидите список всех задач, созданных пользователями.
3. Для каждой задачи будет указано время/дата создания, ФИО, должность, текст и статус (активная или решенная).
4. Вы можете фильтровать задачи по статусу с помощью кнопок "Все задачи", "Активные" и "Решенные".
5. Чтобы отметить задачу как решенную, нажмите кнопку "Решено". Это изменит статус задачи на "Решенная задача".
6. Чтобы отменить решение задачи, нажмите кнопку "Отменить решение". Статус задачи вернется к "Активная задача".

7. Чтобы удалить задачу, нажмите кнопку "Удалить задачу".

Шаг 5: Вход в личный кабинет

1. Если у вас есть учетная запись сотрудника, выберите вкладку "Разбор тикетов" на главной странице.
2. Если у вас нет учетной записи, нажмите кнопку "Войти".
3. Введите имя пользователя и пароль (используйте "user" и "password").
4. Нажмите кнопку "Войти" для входа в личный кабинет.

Шаг 6: Выход из личного кабинета

1. Если вы находитесь в личном кабинете, нажмите кнопку "Выйти" в правом верхнем углу.
2. Вы будете перенаправлены на главную страницу приложения.

Шаг 7: Анализ тикетов (доступно только для аутентифицированных пользователей)

1. Если вы находитесь в личном кабинете сотрудника, выберите одну из вкладок, "Все задачи" показывает абсолютно все задачи, которые есть, "Активные" показывает те задачи, которые еще не решены, у них статус – активные задачи, "Решенные" показывает те задачи, у которых статус – решенная задача.
2. Вы увидите количество задач по каждой категории и сможете проанализировать все ли вы успеваете.

Шаг 8: Выход из приложения

1. Чтобы полностью выйти из приложения, закройте вкладку веб-браузера или остановите запущенный процесс сервера в терминале.

Это полная инструкция по использованию приложения для курсовой работы. Следуя этим шагам, вы сможете создавать тикеты, разбирать их, входить в личный кабинет, анализировать тикеты и выполнять другие функции, предоставленные приложением.

Заключение

В заключение данной работы можно отметить следующие ключевые моменты:

В ходе выполнения данной курсовой работы был разработан сервис по обработке заявок в службу технической поддержки. Целью разработки было улучшение качества обслуживания клиентов, повышение эффективности работы сотрудников и сокращение времени ответа на заявки.

Для достижения этой цели были выполнены следующие задачи:

Изучение существующих методов и подходов к обработке заявок в службе технической поддержки. Был проведен анализ существующих решений и определены основные проблемы, с которыми сталкиваются пользователи при обращении в службу технической поддержки.

Разработка архитектуры и функциональных возможностей сервиса. Были определены требования пользователей и особенности организации, на основе которых была разработана структура и функциональность сервиса.

Реализация сервиса с использованием современных технологий и методов разработки программного обеспечения. Были применены языки программирования, фреймворки и инструменты, соответствующие требованиям проекта.

Результаты данной работы могут быть полезны для организаций, стремящихся улучшить свой сервис технической поддержки и обеспечить более эффективную обработку заявок. Разработанный сервис может служить основой для дальнейших исследований и разработок в области улучшения качества обслуживания клиентов и оптимизации работы службы технической поддержки.

Однако следует отметить, что разработка такого сервиса является лишь началом процесса. Для его успешного внедрения и использования необходимо провести дополнительное тестирование, обучение сотрудников и участие пользователей в процессе дальнейшей разработки и улучшения сервиса.

Список используемой литературы

1. Алекс, Бэнкс React и Redux. Функциональная веб-разработка. Руководство / Бэнкс Алекс. - М.: Питер, 2018. - **458** с.
2. Брайан, П. Хоган HTML5 и CSS3. Веб-разработка по стандартам нового поколения / Брайан П. Хоган. - М.: Питер, 2013. - **128** с.
3. Дакетт, Д. HTML и CSS. Разработка и создание веб-сайтов (+ CD-ROM) / Д. Дакетт. - М.: Эксмо, 2017. - **656** с.
4. Дакетт, Джон Javascript и jQuery. Интерактивная веб-разработка / Джон Дакетт. - М.: Эксмо, 2014. - **146** с.
5. Дари, Кристиан. AJAX и PHP. Разработка динамических веб-приложений / К. Дари. - М.: Символ-плюс, **2013**. - **745** с.
6. Дари, Кристиан AJAX и PHP. Разработка динамических веб-приложений / Кристиан Дари. - М.: Символ-плюс, **2014**. - **475** с.
7. Дэвид, Хэррон Node.js. Разработка серверных веб-приложений на JavaScript / Хэррон Дэвид. - М.: ДМК Пресс, 2016. - **667** с.
8. Мэтью, Дэвид HTML5. Разработка веб-приложений / Дэвид Мэтью. - М.: Рид Групп, **2016**. - 320 с.

Приложение 1

Ссылка на github с кодом - <https://github.com/AlexWolf/serviceSupport>



Код файла App.js

```
import React, { useState } from "react";
import {
  BrowserRouter as Router,
  Routes,
  Route,
  Link,
  Navigate,
} from "react-router-dom";
import SupportPage from "../components/SupportPage";
import EmployeePage from "../components/EmployeePage";
import TicketAnalysisPage from "../components/TicketAnalysisPage";
import LoginPage from "../components/LoginPage";
import "../App.css";

function App() {
  const [isAuthenticated, setIsAuthenticated] = useState(false);

  const handleLogin = () => {
    setIsAuthenticated(true);
  };

  const handleLogout = () => {
    setIsAuthenticated(false);
  };

  return (
    <Router>
      <div>
        <nav>
          <ul>
            <li>
              <Link to="/">  Создать тикет</Link>
            </li>
            <li>
              <Link to="/employee">  Разбор тикетов</Link>
            </li>
            {isAuthenticated ? (
              <li>
                <button onClick={handleLogout} className="logout-button">
                  Выйти
                </button>
              </li>
            ) : null}
          </ul>
        </nav>
```

```

<Routes>
  <Route
    path="/"
    element={ <SupportPage isAuthenticated={isAuthenticated} /> }
  />
  <Route
    path="/employee"
    element={
      isAuthenticated ? <EmployeePage /> : <Navigate to="/login" />
    }
  />
  <Route
    isAuthenticated={isAuthenticated}
    onLogin={handleLogin}
    path="/login"
    element={
      isAuthenticated ? (
        <Navigate to="/" />
      ) : (
        <LoginPage onLogin={handleLogin} />
      )
    }
  />
  <Route
    path="/ticket-analysis"
    element={
      isAuthenticated ? (
        <TicketAnalysisPage />
      ) : (
        <Navigate to="/login" />
      )
    }
  />
</Routes>
</div>
</Router>
);
}

```

```

export default App;

```


Приложение 2

Код файла EmployeePage.js

```
import React, { useEffect, useState } from "react";
import "../EmployeePage.css";

function EmployeePage() {
  const [items, setItems] = useState([]);
  const [filter, setFilter] = useState("all"); // "all" - все задачи, "active" - активные, "resolved" -
  решенные

  useEffect(() => {
    const items = JSON.parse(localStorage.getItem("items"));
    if (items) {
      setItems(items);
    }
  }, []);

  const toggleStatus = (itemId) => {
    const updatedItems = items.map((item) => {
      if (item.id === itemId) {
        return {
          ...item,
          active: !item.active,
        };
      }
      return item;
    });
    setItems(updatedItems);
    localStorage.setItem("items", JSON.stringify(updatedItems));
  };

  const deleteHandle = (itemId) => {
    const updatedItems = items.filter((item) => item.id !== itemId);
    setItems(updatedItems);
    localStorage.setItem("items", JSON.stringify(updatedItems));
  };

  const handleFilter = (selectedFilter) => {
    setFilter(selectedFilter);
  };

  // Фильтрация задач в соответствии с выбранным фильтром
  const filteredItems = items.filter((item) => {
    if (filter === "resolved") {
      return item.active;
    } else if (filter === "active") {
      return !item.active;
    }
  });
  return true;
}
```

```

});

// Функция для подсчета количества задач
const countTasks = (filterType) => {
  if (filterType === "resolved") {
    return items.filter((item) => item.active).length;
  } else if (filterType === "active") {
    return items.filter((item) => !item.active).length;
  } else {
    return items.length;
  }
};

// Если нет задач "Список пуст"
const isEmptyList = filteredItems.length === 0;

return (
  <React.Fragment>
    <div className="employee-container">
      <h2 className={ "name-tab"}>Вкладка для сотрудника</h2>
      <div className={ "btn-category"}>
        <button onClick={() => handleFilter("all")}>
          Все задачи ( {countTasks("all")} )
        </button>
        <button onClick={() => handleFilter("active")}>
          Активные ( {countTasks("active")} )
        </button>
        <button onClick={() => handleFilter("resolved")}>
          Решенные ( {countTasks("resolved")} )
        </button>
      </div>
    </div>
    {isEmptyList ? (
      <p className={ "null_task"}>Список пуст...</p>
    ) : (
      filteredItems.map((item) => {
        return (
          <div className={ "task"} key={item.id}>
            <p>Время/Дата: {item.date}</p>
            <p>От кого: {item.fio}</p>
            <p>Должность: {item.post}</p>
            <p>Текст: {item.text}</p>
            <p className={ "status"}>
              Статус: {item.active ? "Решенная задача" : "Активная задача"}
            </p>
            <div className={ "action-buttons"}>
              {item.active ? (
                <button onClick={() => toggleStatus(item.id)}>
                  Отменить решение
                </button>
              ) : (
                <button onClick={() => toggleStatus(item.id)}>Решено</button>
              )}
            </div>
          </div>
        );
      })
    )}
  </React.Fragment>
);

```

```

    })
    <button
      className={"cancel-button"}
      onClick={() => deleteHandle(item.id)}
    >
      Удалить задачу
    </button>
  </div>
</div>
);
})
})
</React.Fragment>
);
}

export default EmployeePage;

```

Приложение 3

Код файла SupportPage.js

```
import React, { useEffect, useRef, useState } from "react";
import moment from "moment";
import "../SupportPage.css";

function SupportPage() {
  const [items, setItems] = useState([]);

  useEffect(() => {
    const existingItems = JSON.parse(localStorage.getItem("items")) || [];
    const updatedItems = [...items, ...existingItems];
    localStorage.setItem("items", JSON.stringify(updatedItems));
  }, [items]);

  const [fio, setFio] = useState("");
  const [postUser, setPostUser] = useState("");
  const [textError, setTextError] = useState("");
  const [isFioEmpty, setIsFioEmpty] = useState(false);
  const [isPostUserEmpty, setIsPostUserEmpty] = useState(false);
  const [isTextErrorEmpty, setIsTextErrorEmpty] = useState(false);

  const fioChange = (e) => {
    setFio(e.target.value);
    setIsFioEmpty(false);
  };

  const postUserChange = (e) => {
    setPostUser(e.target.value);
    setIsPostUserEmpty(false);
  };

  const textErrorChange = (e) => {
    setTextError(e.target.value);
    setIsTextErrorEmpty(false);
  };

  const SendTicketHandle = (e) => {
    // e.preventDefault();
  };

  const handleSubmit = (e) => {
    e.preventDefault();

    const date = moment().format("HH:mm DD-MM-YYYY");

    if (!fio) {
      setIsFioEmpty(true);
      return;
    }
  }
}
```

```

    }

    if (!postUser) {
      setIsPostUserEmpty(true);
      return;
    }

    if (!textError) {
      setIsTextErrorEmpty(true);
      return;
    }

    let newTicket = [
      {
        id: Date.now(),
        date: date,
        fio: fio,
        post: postUser,
        text: textError,
        active: false,
      },
    ];

    setItems((prevState) => [...newTicket, ...prevState]);

    setFio("");
    setPostUser("");
    setTextError("");
  };

  return (
    <div className="support-container">
      <form onSubmit={handleSubmit} className="support-form">
        <h2 className={ "name-tab"}>Написать тикет</h2>
        <label htmlFor="fio">ФИО:</label>
        <br />
        <input
          value={fio}
          onChange={fioChange}
          type="text"
          id="fio"
          name="fio"
          placeholder={"Иванов Иван Иванович"}
          className={"formUser"}
        />
        {isFioEmpty && (
          <p className={"error"}>*Пожалуйста, заполните поле ФИО.</p>
        )}
        <br />
        <label>Должность:</label>
        <br />
        <input

```

```

        value={postUser}
        onChange={postUserChange}
        list="postUser"
        name="postUser"
        placeholder={"Выберите из списка"}
        className={"formUser"}
    />
    {isPostUserEmpty && (
        <p className={"error"}>*Пожалуйста, выберите должность.</p>
    )}
    <datalist id="postUser">
        <option value="Рядовой пользователь" />
        <option value="Сотрудник/Тестер" />
    </datalist>
    <br />
    <label>Опишите ошибку/вопрос:</label>
    <br />
    <textarea
        value={textError}
        onChange={textErrorChange}
        name="message"
        rows="10"
        cols="30"
        placeholder={"Опишите ошибку/вопрос"}
        className={"formUser"}
    ></textarea>
    {isTextErrorEmpty && (
        <p className={"error"}>
            *Пожалуйста, заполните поле описания ошибки/вопроса.
        </p>
    )}
    <br />
    <button type="submit" onClick={SendTicketHandle}>
        Отправить
    </button>
</form>
</div>
);
}

```

```

export default SupportPage;

```

Приложение 4

Код файла LoginPage.js

```
import React, { useState } from "react";
import "../LoginPage.css";

function LoginPage({ isAuthenticated, onLogin }) {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");

  const handleSubmit = (e) => {
    e.preventDefault();

    // Проверка учетных данных
    if (username === "volkov" && password === "password") {
      onLogin(); // Вызываем функцию onLogin при успешном входе
    } else {
      setError("Неверные учетные данные");
    }
  };

  return (
    <div className="login-container">
      <form onSubmit={handleSubmit} className="login-form">
        <h2>Вход</h2>
        {error && <p className="error">{error}</p>}
        <label htmlFor="username">Имя пользователя:</label>
        <input
          type="text"
          id="username"
          value={username}
          onChange={(e) => setUsername(e.target.value)}
        />
        <br />
        <label htmlFor="password">Пароль:</label>
        <input
          type="password"
          id="password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
        />
        <br />
        <button type="submit">Войти</button>
      </form>
    </div>
  );
}

export default LoginPage;
```