Самостоятельная работа: Создание страницы "Контакты разработчика"

Цель самостоятельной работы

Самостоятельно реализовать новую страницу "Контакты разработчика", закрепив навыки:

- Создания новых компонентов и страниц в React
- Настройки маршрутизации
- Работы с изображениями в React (импорт и использование)
- Создания CSS Modules для стилизации
- Добавления навигационных ссылок

Задание

Создать страницу **"Контакты разработчика"** со следующим содержимым:

1. **Фотография разработчика** (можно использовать любое изображение-заглушку или свое фото)
2. **Имя и краткая информация** о разработчике
3. **Электронный адрес** (в виде ссылки `mailto:`)
4. **Ссылки на социальные сети** (GitHub, Telegram, VK и т.д.) - по желанию
5. **Форма обратной связи** (необязательное усложнение)

Страница должна быть доступна через навигационное меню (Header).

Часть 1: Подробный алгоритм вставки изображения

Что нужно знать перед началом

В React есть **три способа** добавить изображение:

Способ	Когда использовать	Пример
`import`	Изображение внутри `src/` (обрабатывается Vite)	`import photo from './photo.jpg'`
`public` папка	Статические файлы, которые не меняются	`<img src="/images/photo.jpg" />`
URL ссылка	Изображение с другого сайта	`<img src="https://example.com/photo.jpg" />`

Для этой работы рекомендуется способ с `import`, так как Vite оптимизирует изображение.

Алгоритм вставки фото через import

Шаг 1: Выберите или создайте изображение

- Форматы: `.jpg`, `.png`, `.webp`, `.svg`
- Размер: рекомендуется до 500x500px (можно использовать placeholder image)
- Название файла: например, `avatar.jpg`

Шаг 2: Разместите изображение в правильной папке

```
text
blog-ideas/src/
├── pages/
│   ├── ContactsPage.tsx
│   ├── ContactsPage.module.css
│   └── assets/           ← Создайте эту папку
│       └── avatar.jpg    ← Поместите сюда фото
```

Команды для создания папки:

```
powershell
cd blog-ideas/src/pages
mkdir assets
```

Шаг 3: Импортируйте изображение в компоненте

```tsx
// В верху файла ContactsPage.tsx
import photo from './assets/avatar.jpg';
```

Что происходит при импорте:

- Vite обрабатывает файл
- В переменную `photo` сохраняется **обработанный путь** к файлу
- При сборке изображение получает хеш в имени (например, `avatar_abc123.jpg`)

Шаг 4: Используйте изображение в JSX

```tsx
<img
  src={photo}
  alt="Фото разработчика"
  className={styles.photo}
/>
```

Важно: `src={photo}`, а не `src="photo"` (без фигурных скобок будет искать файл с именем "photo")

Шаг 5: Добавьте стили для изображения (опционально)

```css
/* ContactsPage.module.css */
.photo {
  width: 200px;
  height: 200px;
  border-radius: 50%;  /* Сделает фото круглым */
  object-fit: cover;   /* Чтобы фото не искажалось */
  border: 3px solid #007bff;
}
```

Часть 2: Подробный алгоритм вставки электронного адреса

Что нужно знать перед началом

Электронный адрес на веб-странице можно оформить как:

1. **Обычный текст** - пользователь скопирует вручную
2. **Ссылка** `mailto:` - при клике открывается почтовый клиент

Рекомендуется использовать `mailto:`

Синтаксис mailto ссылки

```html
<a href="mailto:адрес@example.com">адрес@example.com</a>
```

Дополнительные параметры mailto:

Параметр	Что делает	Пример
`subject`	Тема письма	`mailto:dev@example.com?subject=Вопрос`
`body`	Тело письма	`mailto:dev@example.com?body=Здравствуйте!`
Несколько параметров	Через `&`	`mailto:dev@example.com?subject=Вопрос&body=Текст`

Пример с параметрами:

```html
<a href="mailto:dev@example.com?subject=Вопрос%20по%20блогу%20идей&body=Здравствуйте%2C%20">
  Написать разработчику
</a>
```

Обратите внимание: пробелы заменяются на `%20`, запятые на `%2C`

Алгоритм добавления email

Шаг 1: Определите email адрес

Для учебных проектов можно использовать заглушку:

- `student@example.com`
- `your-email@mail.ru`
- Или укажите свой реальный email

Шаг 2: Создайте ссылку в компоненте

```tsx
<div className={styles.contactInfo}>
  <h3>📧 Электронная почта</h3>
  <a href="mailto:student@example.com" className={styles.emailLink}>
    student@example.com
  </a>
</div>
```

Шаг 3: Добавьте стили для ссылки

```css
```

```css
/* ContactsPage.module.css */
.emailLink {
  color: #007bff;
  text-decoration: none;
  font-size: 1.1rem;
  transition: color 0.3s ease;
}

.emailLink:hover {
  color: #0056b3;
  text-decoration: underline;
}
```

Шаг 4: (Опционально) Добавьте защиту от спама

```tsx
// Простейшая защита от спам-ботов
const email = "student" + "@" + "example.com";

// Использование
<a href={`mailto:${email}`}>{email}</a>
```

Часть 3: Требования к странице "Контакты разработчика"

Обязательные элементы

1. **Фотография разработчика**
   - Должна корректно отображаться
   - Иметь стилизацию (круглую или с тенью)
   - Должен быть атрибут `alt` с описанием
2. **Информация о разработчике**
   - Имя (реальное или вымышленное)
   - Роль/специализация (например, "Frontend Developer")
   - Краткое описание (2-3 предложения)
3. **Электронный адрес**
   - В виде ссылки `mailto:`
   - При клике открывается почтовый клиент

    - Стилизован как ссылка
4. **Социальные сети (по желанию)**
    - GitHub
    - Telegram
    - LinkedIn
    - VK

Дополнительные элементы (для повышенной оценки)

1. **Форма обратной связи**
    - Поля: имя, email, сообщение
    - Отправка данных (можно просто `console.log`)
2. **Карта с местоположением** (iframe с Яндекс/Google картами)
3. **Кнопка копирования email** (с уведомлением)

Часть 4: Ожидаемый результат

После выполнения работы, при переходе на страницу `/contacts`, пользователь должен видеть:

text

```
┌─────────────────────────────────────────┐
│              Контакты                   │
│         [Фото разработчика]             │
│                                         │
│            Иван Иванов                  │
│         Frontend Developer              │
│                                         │
│    Привет! Я создаю современные веб-    │
│    приложения на React и TypeScript.    │
│                                         │
│    ✉ student@example.com                │
│                                         │
│    Социальные сети:                     │
│    [GitHub] [Telegram] [VK]             │
└─────────────────────────────────────────┘
```

Пошаговый алгоритм выполнения (для студентов)

Важно: Ниже приведен алгоритм действий. Выполните шаги самостоятельно, сверяясь с результатом.

Этап 1: Создание файлов страницы

```
text
Действие 1.1: Создайте файл ContactsPage.tsx
  - Путь: src/pages/ContactsPage.tsx

Действие 1.2: Создайте файл стилей
  - Путь: src/pages/ContactsPage.module.css

Действие 1.3: Создайте папку assets (если нет)
  - Путь: src/pages/assets/

Действие 1.4: Поместите изображение
  - Скопируйте фото в src/pages/assets/avatar.jpg
  - Или используйте ссылку на placeholder: https://via.placeholder.com/200
```

Этап 2: Создание содержимого страницы

```
text
Действие 2.1: Импортируйте изображение
  - Добавьте: import photo from './assets/avatar.jpg';

Действие 2.2: Создайте функциональный компонент
  - Название: ContactsPage
  - Используйте export default

Действие 2.3: Добавьте JSX разметку
  - Контейнер с классом styles.container
  - Заголовок h1
  - Контейнер для фото
  - Блок с информацией о разработчике
  - Блок с email (ссылка mailto)
  - Блок с социальными сетями
```

Этап 3: Стилизация страницы

```
text
Действие 3.1: Создайте CSS классы
  - .container - центровка и отступы
  - .photo - размер, форма, отступы
  - .info - стили текста
  - .emailLink - стили ссылки
  - .socialLinks - контейнер для иконок

Действие 3.2: Добавьте адаптивность
  - @media (max-width: 600px) - для мобильных устройств
```

Этап 4: Настройка маршрутизации

```
text
Действие 4.1: Импортируйте ContactsPage в App.tsx
  - Добавьте: import ContactsPage from './pages/ContactsPage';

Действие 4.2: Добавьте новый Route
  - <Route path="/contacts" element={<ContactsPage />} />
  - Разместите после других маршрутов
```

Этап 5: Обновление навигации

```
text
Действие 5.1: Откройте Header.tsx

Действие 5.2: Добавьте новую ссылку
  - <Link to="/contacts" className={styles.link}>Контакты</Link>

Действие 5.3: (Опционально) Обновите getLinkClass
  - Добавьте проверку для '/contacts'
```

Этап 6: Проверка

```
text
Действие 6.1: Запустите сервер разработки
  - pnpm run dev

Действие 6.2: Проверьте навигацию
  - Перейдите по ссылке "Контакты"

Действие 6.3: Проверьте отображение
  - Фото загружается?
  - Email кликабелен?
  - Все стили применены?
```

```
Действие 6.4: Проверьте адаптивность
  - Уменьшите окно браузера
  - Проверьте отображение на мобильных размерах
```

Этап 7: (Необязательно) Форма обратной связи

```
text
Действие 7.1: Добавьте состояние для формы
  - const [formData, setFormData] = useState({ name: '', email: '', message: '' })

Действие 7.2: Создайте поля ввода
  - input для имени
  - input для email
  - textarea для сообщения

Действие 7.3: Добавьте обработчик отправки
  - handleSubmit с e.preventDefault()
  - console.log(formData)
  - Очистка формы или уведомление

Действие 7.4: Стилизуйте форму
  - Добавьте классы для .form, .formGroup, .formInput, .formButton
```

Критерии оценки

Критерий	Балл	Отметка о выполнении
Создан файл ContactsPage.tsx	1	☐
Создан файл ContactsPage.module.css	1	☐
Фото отображается корректно	2	☐
Использован import для фото	1	☐
Добавлен alt атрибут	1	☐
Email оформлен как mailto ссылка	2	☐

Критерий	Балл	Отметка о выполнении
Добавлена информация о разработчике	1	☐
Добавлен маршрут в App.tsx	1	☐
Добавлена ссылка в Header	1	☐
Стилизация соответствует макету	2	☐
Адаптивность для мобильных устройств	2	☐
Максимальный балл	**15**	

Дополнительные баллы:

Дополнительный критерий	Балл
Форма обратной связи	+3
Кнопка копирования email	+2
Карта с местоположением	+2
Анимация при наведении	+1

Пример кода (справочный материал)

Минимальная структура ContactsPage.tsx

tsx

```tsx
import styles from './ContactsPage.module.css';
// TODO: Импортируйте фото

function ContactsPage() {
  // TODO: Добавьте состояние для формы (опционально)

  return (
```

```
    <div className={styles.container}>
      <h1>Контакты разработчика</h1>

      {/* TODO: Добавьте фото */}
      {/* TODO: Добавьте информацию */}
      {/* TODO: Добавьте email ссылку */}
      {/* TODO: Добавьте социальные сети */}
    </div>
  );
}

export default ContactsPage;
```

Минимальная структура ContactsPage.module.css

```
css
.container {
  max-width: 800px;
  margin: 0 auto;
  padding: 40px 20px;
  text-align: center;
}

/* TODO: Добавьте стили для фото */
/* TODO: Добавьте стили для информации */
/* TODO: Добавьте стили для email ссылки */
/* TODO: Добавьте медиа-запрос для мобильных устройств */
```

Порядок сдачи работы

1. Запустите приложение и убедитесь, что страница "Контакты" работает
2. Сделайте скриншот страницы
3. Запушьте изменения на GitHub:
   ```
   powershell

   git add .
   git commit -m "feat: add contacts page"
   git push
   ```
4. Задеплойте обновление на GitHub Pages:
   ```
   powershell
   ```