Sign up

Email address

Choose a password

At least 16 characters OR at least 8 characters including a number and a letter.

Submit

OR

Email me a code

Secured by Userfront

Sign up

Email address

Choose a password

At least 16 characters OR at least 8 characters including a number and a letter.

Submit

OR

Email me a code

Log in

Email address or username

Password

Submit

OR

Email me a code

Reset your password

Email address

Submit

Secured by Userfront

Test Mode: http

Log out

Welcome

You're logged in!

Secured by Userfront

Profile

Username: katie123

Name: Ekaterina

Caption: lorem ispum

Email: ekaterinabarkun0@gmail.com

Редактировать

katie123

Ekaterina

lorem ispum

ekaterinabarkun0@gmail.com

Выберите файл Файл не выбран

Сохранить

Введите ваше сообщение...

Выбрать файлы Файл не выбран

Отправить

123123123123
qwerty

Выбрать файлы Без названия (2).jpg

Отправить

123123123123 qwerty

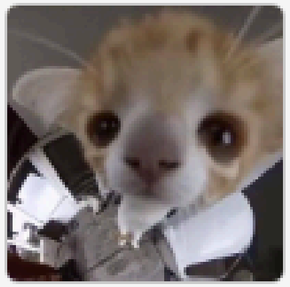
1 секунду назад

Удалить ♡ **0**

Комментарии

Добавить комментарий...

123



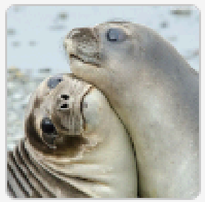
2 секунды назад

Удалить 0

Комментарии

Добавить комментарий...

123123123123 qwerty



33 секунд назад

Удалить 1

Комментарии

hi!

hello

Добавить комментарий...

App.jsx

```
return (
  <Router>
    <div className="menu">
      <Link to="/">Главная</Link>
      <Link to="/post">Написать пост</Link>
      <Link to="/profile">Профиль</Link>
      <Link to="/registration/home">Регистрация</Link>
    </div>
    <Routes>
      <Route path="/" element={
        <RequireAuth>
          <HomeComponent posts={posts} deletePost={deletePost} addComment={addComment} toggleLike={toggleLike} />
        </RequireAuth>
      } />
     <Route path="/post" element={
<RequireAuth>
  <PostComponent addPost={addPost}/>
</RequireAuth>
} />

      <Route path="/profile" element={<RequireAuth><ProfileComponent /></RequireAuth>} />
      <Route path="/registration/*" element={<Registration />} />
    </Routes>
  </Router>
```

```jsx
function EditProfile({ profileData, onSave }) {
  const [name, setName] = useState(profileData.name || '');
  const [unName, setUnName] = useState(profileData.unName || '');
  const [caption, setCaption] = useState(profileData.caption || '');
  const [image, setImage] = useState(profileData.image || null);
  const [email, setEmail] = useState(profileData.email ||  '');

  const [error, setError] = useState('');

  const handleImageChange = async (event) => {
    const file = event.target.files[0];

    const reader = new FileReader();
    reader.onloadend = () => {
      setImage(reader.result);
    };
    reader.readAsDataURL(file);
  };

  const handleSave = async () => {
    try {
      await Userfront.user.update({
        name: name,
        username: unName,
        email: email,
        data: { caption: caption, image: image },
      });
      onSave({ unName, name, email, caption, image });
    } catch (error) {
      console.error('Failed to update profile:', error.response.data);
      setError('Failed to update profile. Please try again.');
    }
  };
```

```jsx
function ProfileComponent() {
  const [activeTab, setActiveTab] = useState('profile');
  const [profileData, setProfileData] = useState({
    unName: '',
    name: '',
    email: Userfront.user.email,
    caption: '',
    image: null
  });

  useEffect(() => {
    const savedData = JSON.parse(localStorage.getItem('profileData'));
    if (savedData) {
      setProfileData(savedData);
    }
  }, []);

  const handleTabChange = (tab) => {
    setActiveTab(tab);
  };

  const handleProfileSave = (data) => {
    setProfileData(data);
    localStorage.setItem('profileData', JSON.stringify(data));
    setActiveTab('profile');
  };
```

Home.jsx

```
  const diffInMilliseconds = now - postDate;
  const diffInSeconds = Math.floor(diffInMilliseconds / 1000);
  const diffInMinutes = Math.floor(diffInSeconds / 60);
  const diffInHours = Math.floor(diffInMinutes / 60);
  const diffInDays = Math.floor(diffInHours / 24);

  if (diffInSeconds < 1) {
    return 'только что';
  } else if (diffInSeconds < 60) {
    return `${diffInSeconds} ${getSecondsText(diffInSeconds)} назад`;
  } else if (diffInDays >= 1) {
    return `${postDate.getDate()} ${getMonthName(postDate.getMonth())}`;
  } else if (diffInHours >= 1) {
    return `${diffInHours} ${getHoursText(diffInHours)} назад`;
  } else {
    return `${diffInMinutes} ${getMinutesText(diffInMinutes)} назад`;
  }
};

const getMonthName = (monthIndex) => {
  const months = [
    'января', 'февраля', 'марта', 'апреля',
    'мая', 'июня', 'июля', 'августа',
    'сентября', 'октября', 'ноября', 'декабря'
  ];
  return months[monthIndex];
};

const getHoursText = (hours) => {
  if (hours === 1) {
```

```jsx
return (
  <div className="home-container">
    {posts.length > 0 ? (
      posts.map((post, index) => (
        <div key={index} className="post-item">
          <p>{post.message}</p>
          {post.images && post.images.length > 0 && (
            <div className="post-images">
              {post.images.map((image, idx) => (
                <img key={idx} src={image} alt={`Attached ${idx}`} className="post-image" />
              ))}
            </div>
          )}
          <span className="post-date">{formatDate(post.date)}</span>
          <button onClick={() => deletePost(index)} className="delete-button">Удалить</button>
          <button onClick={() => toggleLike(index)} className={`like-button ${post.liked ? 'liked' : ''}`}>
            {post.liked ? '❤️' : '🤍'} {post.likes}
          </button>
          <div className="comments-section">
            <h4>Комментарии</h4>
            {post.comments.map((comment, idx) => (
              <p key={idx}>{comment}</p>
            ))}
            <input
              type="text"
              placeholder="Добавить комментарий..."
              onKeyDown={(e) => {
                if (e.key === 'Enter' && e.target.value.trim() !== '') {
                  addComment(index, e.target.value.trim());
                  e.target.value = '';
                }
              }}
            />
```

Home.jsx

```jsx
export default function PostComponent({ addPost}) {
 const [message, setMessage] = useState('');
 const [images, setImages] = useState([]);

 const handleMessageChange = (event) => {
   setMessage(event.target.value);
 };

 const handleImageChange = (event) => {
   if (event.target.files) {
     const files = Array.from(event.target.files);
     const imagePromises = files.map(file => {
       return new Promise((resolve, reject) => {
         const reader = new FileReader();
         reader.onload = (e) => resolve(e.target.result);
         reader.onerror = reject;
         reader.readAsDataURL(file);
       });
     });

     Promise.all(imagePromises)
       .then(images => setImages(images))
       .catch(error => console.error("Error reading files:", error));
   }
 };

 const handleMessageSubmit = () => {
   if (message.trim() !== '') {
     const post = {
```

```jsx
 const handleMessageSubmit = () => {
     const post = {

       message,
       date: new Date().toISOString(),
       images,
     };
     addPost(post);
     setMessage('');
     setImages([]);
   }
 };
```

На данный момент я собираюсь реализовать в своём проекте следующие механики:

- уже реализованы

- не до конца реализовано

- планирую в будущем реализовать

- очень постараюсь сделать

---

1. # Создание главной страницы (общие возможности):
    - #Добавление стилей CSS
    - #Наличие меню переключения между страницами
    - # Просмотр записей по датам
    - #Авторизация
    - # Регистрация
    - # Сортировка по тегам
    - # Реализация возможности подписки на аккаунты
    - # Реализация добавления подписчиков
    - # Реализация возможности сохранения записей
    - # Реализация возможности поставить лайк
    - # Отображение записей (в том числе людей на которых вы подписаны начиная от новых)

2. # Создание страницы профиля
    - # Возможность редактирования имени
    - # Возможность редактирования аватарки
    - # Поле для имени, уникального никнейма и электронной почты
    - # Выбор текущего языка
    - # Добавление описания профиля
    - # Возможность отписки
    - # Просмотр всех своих записей

3. # Создание страницы на которой создаются посты
    - #Добавление стилей CSS
    - # Возможность добавления записи на страницу
    - # Возможность её удаления
    - # Отображение даты и времени публикации
    - # Возможность редактирования записи
    - #Добавление картинки к записи
    - # Добавление вкладочки со смайликами в запись

4. # Страница поиска
    - #Возможность искать аккаунты пользователей по никнейму и переходить по ним
    - #Рандомные рекомендованные аккаунты

5. # Страница с уведомлениями
    - # Все взаимодействия с вашим аккаунтом должны отображаться на странице

6. # Страница с чатом с пользователями
    - # Возможность написать пользователям сети в личные сообщения