Отчет по Домашнему Заданию по курсу
"Разработка Интернет-Приложений"

Выполнил:
Студент группы
ИУ5-55Б
Хижняков Вадим Максимович

Москва, МГТУ – 2021

Задание:
Разработка прототипа SPA приложения.

Ссылка на репозиторий:
https://github.com/DonVadimon/graphql-try

Ссылка на приложение:
https://dv-graphql-try-630b6b.netlify.app

Реализация:
Приложение представляет из себя доску постов с возможностью регистрации и авторизации.
Домашнее задание реализовано в виде монорепозитория.
Для создания серверной части приложения был выбран фреймворк express.js.
В качестве хранилища данных используется база данных MongoDB, размещенная на удаленном хосте Atlassian.
Для клиентской части использовалась библиотека React.
Для стилизации компонентов использовалась библиотека styled-components. Для разработки использовалась утилита storybook.
Для взаимодействия между клиентом и сервером используется GraphQL.

Текст программы:

```
/server/mocks/posts.js

const posts = [
   {
       id: 1,
       author: 1,
       title: 'first post!',
       content: 'graphql is cool',
   },
   {
       id: 2,
```

```
        author: 2,
        title: 'vadim lox',
        content: 'ne nu a che on bichit',
    },
];

module.exports = { posts };
```

/server/mocks/users.js

```
const users = [
    {
        id: 1,
        username: 'DonVadimon',
        password: 'qwerty',
        age: 19,
        avatar:
'https://sun9-37.userapi.com/impf/c846523/v846523943/1a9e5d/fWocEUzzWV4.jpg?size=797x598&qu
ality=96&sign=08bfcb39308a6fdda4b304f3f0155b6e&type=album',
    },
    {
        id: 2,
        username: 'Sanya',
        password: 'sanya_lox',
        age: 19,
        avatar:
'https://sun9-33.userapi.com/impf/c850124/v850124974/70535/M5GAM3CKsdY.jpg?size=240x239&qua
lity=96&sign=be72dbb27d9adcd3896f99843758b30b&type=album',
    },
];

module.exports = { users };
```

/server/package.json

```
{
    "name": "server",
    "version": "1.0.0",
    "description": "",
    "main": "index.js",
    "scripts": {
        "dev": "nodemon index.js",
        "lint": "eslint \"./**/*.+(js|jsx|ts|tsx|json)\"",
        "lint-fix": "eslint --fix \"./**/*.+(js|jsx|ts|tsx|json)\"",
        "prettier": "prettier --write \"./**/*.+(js|jsx|ts|tsx|json)\""
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "dependencies": {
        "cors": "^2.8.5",
        "express": "4.17.1",
        "express-graphql": "^0.12.0",
```

```
        "graphql": "^15.5.1",
        "graphql-tools": "^7.0.5",
        "nodemon": "^2.0.7",
        "eslint-plugin-prettier": "^3.4.1",
        "eslint-config-airbnb-base": "^14.2.1"
    },
    "devDependencies": {
        "@types/express": "^4.17.13"
    }
}


/server/index.js

const express = require('express');
const { graphqlHTTP } = require('express-graphql');
const cors = require('cors');
const { schema } = require('./schema');
let { users } = require('./mocks/users');
let { posts } = require('./mocks/posts');

const app = express();
app.use(cors());

const PORT = process.env.PORT || 8000;

app.get('/', (_req, res) => res.send('Hello World!'));

const root = {
    getAllUsers: () => users,
    getUser: ({ id }) => users.find((user) => user.id === Number(id)),
    createUser: ({ input }) => {
        const avatar =
            input.avatar ||

'https://sun9-2.userapi.com/impf/c847120/v847120865/12441c/wVW70T08Oas.jpg?size=600x600&qua
lity=96&sign=0475cb87976fd939dab397904e6c2610&type=album';
        const user = {
            id: Date.now(),
            ...input,
            avatar,
        };
        users.push(user);
        return user;
    },
    getAllPosts: () => posts,
    getPost: ({ id }) => posts.find((post) => post.id === Number(id)),
    getUserPosts: ({ author }) => posts.filter((post) => post.author === Number(author)),
    createPost: ({ input }) => {
        const post = {
            id: Date.now(),
            ...input,
        };
        posts.push(post);
        return post;
```

```
    },
    deletePost: ({ id }) => {
        const deletedPost = posts.find((post) => Number(post.id) === Number(id));
        posts = posts.filter((post) => Number(post.id) !== Number(id));
        return deletedPost;
    },
    login: ({ input }) =>
        users.find(({ username, password }) => username === input.username && password ===
input.password),
    deleteUser: ({ id }) => {
        const deletedUser = users.find((user) => Number(user.id) === Number(id));
        users = users.filter((user) => Number(user.id) !== Number(id));
        return deletedUser;
    },
};

app.use(
    '/graphql',
    graphqlHTTP({
        graphiql: true,
        schema,
        rootValue: root,
    }),
);

app.listen(PORT, () =>
    // eslint-disable-next-line no-console
    console.log(`Server started on http://localhost:${PORT}, see graphiql on
http://localhost:${PORT}/graphql`),
);
```

/server/.eslintignore

package.json

/server/schema/schema.graphql

```
type User {
 id: ID!
 username: String!
 password: String!
 age: Int!
 avatar: String
}

type Post {
 id: ID
 author: ID
 title: String
 content: String
}

input UserInput {
```

```
 id: ID
 username: String!
 password: String!
 age: Int!
 avatar: String
}

input PostInput {
 id: ID
 author: ID!
 title: String!
 content: String!
}

input LoginInput {
 username: String!
 password: String!
}

type Mutation {
 createUser(input: UserInput): User
 deleteUser(id: ID): User
 login(input: LoginInput): User
}

type Query {
 getAllUsers: [User]
 getUser(id: ID): User
 getAllPosts: [Post]
 getPost(id: ID): Post
 getUserPosts(author: ID): [Post]
}

type Mutation {
 createPost(input: PostInput): Post
 deletePost(id: ID): Post
}
```

/server/schema/index.js

```
const { loadSchemaSync, GraphQLFileLoader } = require('graphql-tools');
const path = require('path');

const schema = loadSchemaSync(path.resolve(__dirname, 'schema.graphql'), {
   loaders: [new GraphQLFileLoader()],
});

module.exports = { schema };
```

/server/.eslintrc

```
{
   "env": {
```

```
        "es6": true,
        "node": true
    },
    "extends": ["airbnb-base", "prettier"],
    "plugins": ["prettier"],
    "globals": {
        "Atomics": "readonly",
        "SharedArrayBuffer": "readonly"
    },
    "parserOptions": {
        "ecmaVersion": 2018,
        "sourceType": "module"
    },
    "rules": {
        "prettier/prettier": "error",
        "class-methods-use-this": "off",
        "no-param-reassign": "off",
        "camelcase": "off",
        "no-unused-vars": ["error", { "argsIgnorePattern": "next" }]
    }
}


/package.json

{
    "name": "graphql-try",
    "private": true,
    "workspaces": [
        "client",
        "server"
    ],
    "version": "1.0.0",
    "main": "index.js",
    "license": "MIT",
    "scripts": {
        "start": "node server",
        "rmall": "rm -rf node_modules && rm -rf ./client/node_modules && rm -rf
./server/node_modules",
        "lint": "cd server && yarn lint && cd ../client && yarn lint",
        "lint-fix": "cd server && yarn lint-fix && cd ../client && yarn lint-fix",
        "prettier": "cd server && yarn prettier && cd ../client && yarn prettier"
    },
    "devDependencies": {
        "@types/express": "^4.17.13",
        "@typescript-eslint/eslint-plugin": "^4.16.1",
        "@typescript-eslint/parser": "^4.16.1",
        "classnames": "^2.3.1",
        "eslint": "^7.21.0",
        "eslint-config-airbnb-base": "^14.2.1",
        "eslint-config-prettier": "^8.1.0",
        "eslint-plugin-eslint-comments": "^3.2.0",
        "eslint-plugin-import": "^2.22.1",
        "eslint-plugin-prettier": "^3.4.1",
        "eslint-plugin-react": "^7.22.0",
```

```
       "eslint-plugin-react-hooks": "^4.2.0",
       "eslint-plugin-simple-import-sort": "^7.0.0",
       "eslint-plugin-standard": "^5.0.0",
       "husky": "^4.3.8",
       "lint-staged": "^11.1.2",
       "prettier": "^2.3.2"
   }
}


/.editorconfig

root = true

[*]
charset = utf-8
end_of_line = lf
indent_size = 4
indent_style = space
max_line_length = 120
insert_final_newline = true
trim_trailing_whitespace = true


/.prettierignore

# Add files here to ignore them from prettier formatting

/dist
/build
/coverage
package-lock.json
yarn.lock


/Procfile

worker: node server


/.huskyrc

{
 "hooks": {
   "pre-commit": "yarn lint-fix && yarn prettier"
 }
}


/client/package.json

{
   "name": "client",
   "version": "0.1.0",
   "private": true,
```

```
  "dependencies": {
      "@apollo/client": "^3.3.20",
      "final-form": "^4.20.2",
      "graphql": "^15.5.1",
      "react": "^17.0.2",
      "react-dom": "^17.0.2",
      "react-final-form": "^6.5.3",
      "react-router-dom": "^5.2.1",
      "react-scripts": "4.0.3",
      "styled-components": "^5.3.1",
      "typescript": "^4.1.2",
      "web-vitals": "^1.0.1"
  },
  "scripts": {
      "start": "react-scripts start",
      "build": "react-scripts build",
      "test": "react-scripts test --watchAll=false",
      "eject": "react-scripts eject",
      "lint": "eslint \"src/**/*.+(js|jsx|ts|tsx|json)\"",
      "lint-fix": "eslint --fix \"src/**/*.+(js|jsx|ts|tsx|json)\"",
      "prettier": "prettier --write \"src/**/*.+(js|jsx|ts|tsx|json)\"",
      "storybook": "start-storybook -p 6006 -s public",
      "build-storybook": "build-storybook -s public"
  },
  "devDependencies": {
      "@storybook/addon-actions": "^6.3.8",
      "@storybook/addon-essentials": "^6.3.8",
      "@storybook/addon-links": "^6.3.8",
      "@storybook/node-logger": "^6.3.8",
      "@storybook/preset-create-react-app": "^3.2.0",
      "@storybook/react": "^6.3.8",
      "@svgr/webpack": "^5.5.0",
      "@testing-library/jest-dom": "^5.11.4",
      "@testing-library/react": "^11.1.0",
      "@testing-library/user-event": "^12.1.10",
      "@types/jest": "^26.0.15",
      "@types/node": "^12.0.0",
      "@types/react": "^17.0.0",
      "@types/react-dom": "^17.0.0",
      "@types/react-router-dom": "^5.1.8",
      "@types/styled-components": "^5.1.13",
      "@typescript-eslint/eslint-plugin": "^4.16.1",
      "@typescript-eslint/parser": "^4.16.1",
      "classnames": "^2.3.1",
      "eslint": "^7.21.0",
      "eslint-config-prettier": "^8.1.0",
      "eslint-plugin-eslint-comments": "^3.2.0",
      "eslint-plugin-import": "^2.22.1",
      "eslint-plugin-react": "^7.22.0",
      "eslint-plugin-react-hooks": "^4.2.0",
      "eslint-plugin-simple-import-sort": "^7.0.0",
      "eslint-plugin-standard": "^5.0.0",
      "husky": "^4.3.8",
      "lint-staged": "^11.1.2",
      "prettier": "^2.3.2",
```

```
        "redux-devtools-extension": "^2.13.9",
        "storybook-addon-apollo-client": "^4.0.9"
    },
    "eslintConfig": {
        "extends": [
            "react-app",
            "react-app/jest"
        ],
        "overrides": [
            {
                "files": [
                    "**/*.stories.*"
                ],
                "rules": {
                    "import/no-anonymous-default-export": "off"
                }
            }
        ]
    },
    "browserslist": {
        "production": [
            ">0.2%",
            "not dead",
            "not op_mini all"
        ],
        "development": [
            "last 1 chrome version",
            "last 1 firefox version",
            "last 1 safari version"
        ]
    }
}


/client/tsconfig.json

{
    "compilerOptions": {
        "baseUrl": "src",
        "target": "es5",
        "lib": ["dom", "dom.iterable", "esnext"],
        "allowJs": true,
        "skipLibCheck": true,
        "esModuleInterop": true,
        "allowSyntheticDefaultImports": true,
        "strict": true,
        "forceConsistentCasingInFileNames": true,
        "noFallthroughCasesInSwitch": true,
        "module": "esnext",
        "moduleResolution": "node",
        "resolveJsonModule": true,
        "isolatedModules": true,
        "noEmit": true,
        "jsx": "react-jsx"
    },
```

```
   "include": ["src"],
   "exclude": ["node_modules"]
}
```

/client/.storybook/global.css

```
* {
   margin: 0;
   font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
'Ubuntu', 'Cantarell', 'Fira Sans',
       'Droid Sans', 'Helvetica Neue', sans-serif;
   -webkit-font-smoothing: antialiased;
   -moz-osx-font-smoothing: grayscale;
   box-sizing: border-box;
}
```

/client/.storybook/preview.js

```
import { MockedProvider } from '@apollo/client/testing';
import { BrowserRouter } from 'react-router-dom';

import './global.css';

export const parameters = {
   apolloClient: {
       MockedProvider,
   },
   actions: { argTypesRegex: '^on[A-Z].*' },
   controls: {
       matchers: {
           color: /(background|color)$/i,
           date: /Date$/,
       },
   },
};

export const decorators = [(story) => <BrowserRouter>{story()}</BrowserRouter>];
```

/client/.storybook/main.js

```
module.exports = {
   stories: ['../src/**/*.stories.mdx', '../src/**/*.stories.@(js|jsx|ts|tsx)'],
   addons: [
       '@storybook/addon-links',
       '@storybook/addon-essentials',
       '@storybook/preset-create-react-app',
       'storybook-addon-apollo-client',
   ],
};
```

/client/src/context/index.ts

```
import { createContext } from 'react';

import { IContextValue } from './types';

export const Context = createContext<IContextValue>({ userId: -1, setUserId: () =>
undefined });


/client/src/context/types.ts

import React, { SetStateAction } from 'react';

export interface IContextValue {
   userId: number;
   setUserId: React.Dispatch<SetStateAction<number>>;
}


/client/src/reportWebVitals.ts

import { ReportHandler } from 'web-vitals';

const reportWebVitals = (onPerfEntry?: ReportHandler) => {
   if (onPerfEntry && onPerfEntry instanceof Function) {
       void import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
           getCLS(onPerfEntry);
           getFID(onPerfEntry);
           getFCP(onPerfEntry);
           getLCP(onPerfEntry);
           getTTFB(onPerfEntry);
       });
   }
};

export default reportWebVitals;


/client/src/GraphQl/mutations/posts.ts

import { gql } from '@apollo/client';

export const CREATE_POST = gql`
   mutation createPost($input: PostInput) {
       createPost(input: $input) {
           title
           content
           author
       }
   }
`;

export const DELETE_POST = gql`
   mutation deletePost($id: ID) {
       deletePost(id: $id) {
```

```
            id
            title
        }
    }
`;


/client/src/GraphQl/mutations/user.ts

import { gql } from '@apollo/client';

export const CREATE_USER = gql`
    mutation createUser($input: UserInput) {
        createUser(input: $input) {
            id
        }
    }
`;

export const DELETE_USER = gql`
    mutation deleteUser($id: ID) {
        deleteUser(id: $id) {
            id
        }
    }
`;

export const LOGIN = gql`
    mutation login($input: LoginInput) {
        login(input: $input) {
            username
            age
            id
        }
    }
`;


/client/src/GraphQl/queries/posts.ts

import { gql } from '@apollo/client';
import { IPost } from 'types/Post.types';

export interface IGetAllPostsResp {
    getAllPosts: IPost[];
}

export interface IGetUserPostsResp {
    getUserPosts: IPost[];
}

export const GET_ALL_POSTS = gql`
    query getAllPosts {
        getAllPosts {
            id
```

```
            author
            title
            content
        }
    }
`;

export const GET_USER_POSTS = gql`
    query getUserPosts($author: ID) {
        getUserPosts(author: $author) {
            title
            content
        }
    }
`;


/client/src/GraphQl/queries/user.ts

import { gql } from '@apollo/client';
import { IUser } from 'types/User.types';

export interface IGetAllUsersResp {
    getAllUsers: IUser[];
}

export const GET_ALL_USERS = gql`
    query getAllUsers {
        getAllUsers {
            id
            username
            age
            avatar
        }
    }
`;

export const GET_USER = gql`
    query getUser($id: ID) {
        getUser(id: $id) {
            username
            avatar
            age
            id
        }
    }
`;

export const GET_USER_MIN_INFO = gql`
    query getUser($id: ID) {
        getUser(id: $id) {
            username
            avatar
        }
    }
```

```
`;


/client/src/tools/getCookie.ts

export const getCookie = (name: string) => {
   const matches = new RegExp('(?:^|; )' + name.replace(/([.$?*|{}()[]\\\/\+^])/g, '\\$1')
+ '=([^;]*)').exec(
       document.cookie,
   );
   return matches ? decodeURIComponent(matches[1]) : undefined;
};


/client/src/tools/setCookie.ts

export function setCookie<T extends string | number | boolean>(
   name: string,
   value: T,
   options: { path?: string; expires?: Date } = { path: '/', expires: new Date(Date.now() +
86400e3 * 3) },
) {
   let updatedCookie = encodeURIComponent(name) + '=' + encodeURIComponent(value);

   const expires = options.expires?.toUTCString();

   updatedCookie += `; path=${options.path || '/'}; expires=${
       expires || new Date(Date.now() + 86400e3 * 3).toUTCString()
   }`;

   document.cookie = updatedCookie;
}


/client/src/react-app-env.d.ts

/// <reference types="react-scripts" />


/client/src/setupTests.ts

// jest-dom adds custom jest matchers for asserting on DOM nodes.
// allows you to do things like:
// expect(element).toHaveTextContent(/react/i)
// learn more: https://github.com/testing-library/jest-dom
import '@testing-library/jest-dom';


/client/src/assets/index.ts

export { ReactComponent as Logo } from './icons/logo.svg';
export { ReactComponent as Minus } from './icons/minus.svg';
export { ReactComponent as Plus } from './icons/plus.svg';
export { ReactComponent as Trash } from './icons/trash.svg';
export { default as Axoops } from './images/Axoops.png';
```

/client/src/hooks/useUserId.tsx

```
import { Dispatch, SetStateAction, useEffect, useState } from 'react';
import { getCookie } from 'tools/getCookie';

export const useUserId = (): [userId: number, setUserId: Dispatch<SetStateAction<number>>]
=> {
   const [userId, setUserId] = useState(-1);

   useEffect(() => {
       setUserId(() => Number(getCookie('login')) || -1);
   }, []);

   return [userId, setUserId];
};
```

/client/src/hooks/useLogout.tsx

```
import { useCallback, useContext } from 'react';
import { useHistory } from 'react-router';
import { Context } from 'context';
import { setCookie } from 'tools/setCookie';

export const useLogout = () => {
   const history = useHistory();

   const { setUserId } = useContext(Context);

   const logout = useCallback(() => {
       setCookie('login', '-1');
       history.push('/');
       setUserId(-1);
   }, [history, setUserId]);

   return logout;
};
```

/client/src/components/DeleteButton/DeleteButton.stories.tsx

```
import React from 'react';
import { Meta, Story } from '@storybook/react/types-6-0';
import { Trash } from 'assets';

import { DeleteButton } from './DeleteButton';

export default {
   title: 'Components/Atoms/DeleteButton',
   component: DeleteButton,
   argTypes: {},
} as Meta;
```

```
export const Default: Story = (args) => (
   <DeleteButton {...args}>
       <Trash />
   </DeleteButton>
);
Default.args = {
   onClick: () => alert('click'),
};


/client/src/components/DeleteButton/DeleteButton.tsx

import styled from 'styled-components';

export const DeleteButton = styled.button`
   background: none;
   border: 1px solid tomato;
   border-radius: 4px;
   color: tomato;
   transition: all linear 100ms;
   padding: 8px;
   margin: 0;
   position: absolute;
   top: 20px;
   right: 20px;
   display: flex;
   justify-content: center;
   align-items: center;
   cursor: pointer;

   &:hover {
       background: tomato;
       color: white;
   }

   svg {
       width: 32px;
       height: 32px;
   }
`;


/client/src/components/DeleteButton/index.ts

export { DeleteButton } from './DeleteButton';


/client/src/components/NewPostForm/NewPostForm.tsx

import React, { useCallback, useContext } from 'react';
import { Field, Form } from 'react-final-form';
import { useMutation } from '@apollo/client';
import { Context } from 'context';
import { FormApi } from 'final-form';
import { CREATE_POST } from 'GraphQl/mutations/posts';
```

```
import { IPost, PostInput } from 'types/Post.types';

import { Button } from 'components/Button';
import { Input } from 'components/Input';
import { Textarea } from 'components/Textarea';

import { FormItem, StyledForm } from './NewPostForm.styles';

const initialValues: Omit<PostInput, 'author'> = {
   title: '',
   content: '',
};

export const NewPostForm = () => {
   const { userId } = useContext(Context);

   const validateText = useCallback((value: string) => (value ? undefined : 'error'), []);

   const [newPost] = useMutation<IPost, { input: PostInput }>(CREATE_POST);

   const createPost = useCallback(
       (values: Omit<PostInput, 'author'>, form: FormApi<Omit<PostInput, 'author'>,
Partial<PostInput>>) => {
           newPost({
               variables: {
                   input: { ...values, author: userId },
               },
           })
               .then(() => form.reset())
               .catch((err) => console.error(err));
       },
       [newPost, userId],
   );

   return (
       <Form<Omit<PostInput, 'author'>> onSubmit={createPost}
initialValues={initialValues}>
           {({ handleSubmit }) => (
               <StyledForm>
                   <Field<string> name="title" validate={validateText}>
                       {({ input, meta }) => (
                           <FormItem>
                               <Input
                                   placeholder="Title"
                                   error={!!meta.error && meta.touched}
                                   type="text"
                                   autoComplete="off"
                                   {...input}
                               />
                           </FormItem>
                       )}
                   </Field>
                   <Field<string> name="content" validate={validateText}>
                       {({ input, meta }) => (
                           <FormItem>
```

```
                            <Textarea
                                placeholder="Content"
                                error={!!meta.error && meta.touched}
                                autoComplete="off"
                                {...input}
                            />
                        </FormItem>
                    )}
                </Field>
                <Button type="button" onClick={handleSubmit}>
                    Create new Post
                </Button>
            </StyledForm>
        )}
    </Form>
  );
};


/client/src/components/NewPostForm/NewPostForm.stories.tsx

import React from 'react';
import { Meta, Story } from '@storybook/react/types-6-0';
import { Container } from 'App/App.styles';
import { CREATE_POST } from 'GraphQl/mutations/posts';

import { NewPostForm } from './NewPostForm';

export default {
  title: 'Components/Atoms/NewPostForm',
  component: NewPostForm,
  argTypes: {},
} as Meta;

export const Default: Story = () => (
  <Container>
    <NewPostForm />
  </Container>
);
Default.parameters = {
  apolloClient: {
    mocks: [
      {
        request: {
          query: CREATE_POST,
        },
        result: {
          data: {
            title: 'new post',
            content: 'Lorem ipsum dolor sit amet consectetur adipisicing elit.
Aspernatur, ipsum?',
            author: 1,
          },
        },
      },
```

```
      ],
   },
};


/client/src/components/NewPostForm/NewPostForm.styles.tsx

import styled from 'styled-components';

export const StyledForm = styled.form`
   display: flex;
   flex-direction: column;
   justify-content: space-between;
   align-items: center;
   width: 80%;
   margin-bottom: 20px;
`;

export const FormItem = styled.div`
   width: 100%;
   max-width: 800px;
   margin: 12px 0;
`;


/client/src/components/NewPostForm/index.ts

export { NewPostForm } from './NewPostForm';


/client/src/components/PostList/PostList.styles.tsx

import styled from 'styled-components';

export const Container = styled.div`
   width: 100%;
   display: flex;
   flex-direction: column;
   align-items: center;
`;

export const ListItem = styled.div`
   margin: 12px 0;
   width: 60%;
`;


/client/src/components/PostList/PostList.types.ts

import { IPost } from 'types/Post.types';

export interface IPostListProps {
   posts: IPost[];
}
```

```
/client/src/components/PostList/PostList.tsx

import React from 'react';

import { PostCard } from 'components/PostCard';

import { Container, ListItem } from './PostList.styles';
import { IPostListProps } from './PostList.types';

export const PostList: React.FC<IPostListProps> = ({ posts }) => {
   return (
       <Container>
           {posts.map((post, index) => (
               <ListItem key={index}>
                   <PostCard post={post} />
               </ListItem>
           ))}
       </Container>
   );
};


/client/src/components/PostList/PostList.stories.tsx

import React from 'react';
import { Meta, Story } from '@storybook/react/types-6-0';
import { IPost } from 'types/Post.types';

import { PostList } from './PostList';
import { IPostListProps } from './PostList.types';

export default {
   title: 'Components/Atoms/PostList',
   component: PostList,
   argTypes: {},
} as Meta;

export const Default: Story<IPostListProps> = (args) => <PostList {...args} />;
Default.args = {
   posts: [
       {
           id: 1,
           title: 'title',
           content: 'Lorem ipsum dolor sit amet consectetur adipisicing elit. Distinctio,
consequatur.',
           author: 1,
       },
       {
           id: 2,
           title: 'title',
           content: 'Lorem ipsum dolor sit amet consectetur adipisicing elit. Distinctio,
consequatur.',
           author: 1,
       },
```

```
      {
          id: 3,
          title: 'title',
          content: 'Lorem ipsum dolor sit amet consectetur adipisicing elit. Distinctio,
consequatur.',
          author: 1,
      },
      {
          id: 4,
          title: 'title',
          content: 'Lorem ipsum dolor sit amet consectetur adipisicing elit. Distinctio,
consequatur.',
          author: 1,
      },
   ] as IPost[],
};


/client/src/components/PostList/index.ts

export { PostList } from './PostList';


/client/src/components/NewUserForm/NewUserForm.tsx

import React, { useCallback, useContext } from 'react';
import { Field, Form } from 'react-final-form';
import { useHistory } from 'react-router';
import { useMutation } from '@apollo/client';
import { Context } from 'context';
import { FormApi } from 'final-form';
import { CREATE_USER } from 'GraphQl/mutations/user';
import { setCookie } from 'tools/setCookie';
import { IUser } from 'types/User.types';

import { Avatar } from 'components/Avatar';
import { Button } from 'components/Button';
import { Counter } from 'components/Counter';
import { Input } from 'components/Input';

import { FormItem, StyledForm } from './NewUserForm.styles';
import { NewUserFormValues } from './NewUserForm.types';

const initialValues: NewUserFormValues = {
   username: '',
   password: '',
   password2: '',
   age: 18,
   avatar: '',
};

const IMG_LINK_REGEX = /^http(s)?:\/\/.+\..*$/gm;

export const NewUserForm = () => {
   const validateText = useCallback((value: string) => (value ? undefined : 'error'), []);
```

```
  const validateAge = useCallback((value: number) => (value < 18 ? 'error' : undefined),
[]);

  const validateImg = useCallback(
      (value: string) => (value && value.match(IMG_LINK_REGEX) ? undefined : 'error'),
      [],
  );

  const [newUser, { error }] = useMutation<{ createUser: Pick<IUser, 'id'>
}>(CREATE_USER);

  const { setUserId } = useContext(Context);

  const history = useHistory();

  const registerUser = useCallback(
      (
          { username, password, avatar, age }: NewUserFormValues,
          form: FormApi<NewUserFormValues, Partial<NewUserFormValues>>,
      ) => {
          newUser({
              variables: {
                  input: { username, password, avatar, age },
              },
          })
              .then((data) => {
                  if (data.data?.createUser?.id && !error) {
                      setCookie('login', data.data.createUser.id.toString());
                      setUserId(Number(data.data.createUser.id.toString()));
                  }
                  return data.data?.createUser.id;
              })
              .then((id) => id && history.push(`/user/${id}`))
              .then(() => form.reset())
              .catch((err) => console.error(err));
      },
      [error, history, newUser, setUserId],
  );

  return (
      <Form<NewUserFormValues>
          onSubmit={registerUser}
          initialValues={initialValues}
          validate={(values) => (values.password !== values.password2 ? { password2:
'error' } : undefined)}
      >
          {({ handleSubmit, values: { avatar } }) => (
              <StyledForm>
                  <Field<string> name="username" validate={validateText}>
                      {({ input, meta }) => (
                          <FormItem>
                              <Input
                                  placeholder="Username"
                                  error={!!meta.error && meta.touched}
```

```
                    type="text"
                    autoComplete="off"
                    {...input}
                />
            </FormItem>
        )}
    </Field>
    <Field<string> name="password" validate={validateText}>
        {({ input, meta }) => (
            <FormItem>
                <Input
                    placeholder="Password"
                    error={!!meta.error && meta.touched}
                    type="password"
                    autoComplete="off"
                    {...input}
                />
            </FormItem>
        )}
    </Field>
    <Field<string> name="password2" validate={validateText}>
        {({ input, meta }) => (
            <FormItem>
                <Input
                    placeholder="Repeat Password"
                    error={!!meta.error && meta.touched}
                    type="password"
                    autoComplete="off"
                    {...input}
                />
            </FormItem>
        )}
    </Field>
    <Field<number> name="age" validate={validateAge}>
        {({ input, meta }) => (
            <FormItem>
                <Counter
                    onMinusClick={() => input.onChange(input.value - 1)}
                    onPlusClick={() => input.onChange(input.value + 1)}
                    value={input.value}
                    error={!!meta.error}
                />
            </FormItem>
        )}
    </Field>
    <Field<string> name="avatar" validate={validateImg}>
        {({ input, meta }) => (
            <FormItem>
                <Input
                    placeholder="Avatar Link"
                    error={!!meta.error && meta.touched}
                    type="text"
                    autoComplete="off"
                    {...input}
                />
```

```
                        </FormItem>
                    )}
                </Field>
                <FormItem>
                    {avatar && !validateImg(avatar) && <Avatar width="200px"
height="200px" src={avatar} />}
                </FormItem>
                <Button type="button" onClick={handleSubmit}>
                    Register
                </Button>
            </StyledForm>
        )}
    </Form>
  );
};


/client/src/components/NewUserForm/NewUserForm.styles.tsx

import styled from 'styled-components';

export const StyledForm = styled.form`
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  align-items: center;
  width: 80%;
  margin-bottom: 20px;
`;

export const FormItem = styled.div`
  margin: 12px 0;
`;



/client/src/components/NewUserForm/NewUserForm.types.ts

import { UserInput } from 'types/User.types';

export type NewUserFormValues = UserInput & { password2: string };


/client/src/components/NewUserForm/index.ts

export { NewUserForm } from './NewUserForm';


/client/src/components/PostCard/PostCard.stories.tsx

import React from 'react';
import { Meta, Story } from '@storybook/react/types-6-0';
import { DELETE_POST } from 'GraphQl/mutations/posts';

import { PostCard } from './PostCard';
```

```
import { IPostCardProps } from './PostCard.types';

export default {
   title: 'Components/Atoms/PostCard',
   component: PostCard,
   argTypes: {},
} as Meta;

export const Default: Story<IPostCardProps> = (args) => <PostCard {...args} />;
Default.args = {
   post: {
       id: 1,
       content: 'Lorem, ipsum dolor sit amet consectetur adipisicing elit. Molestiae,
repellendus!',
       title: 'Post',
       author: 1,
   },
};
Default.parameters = {
   apolloClient: {
       mocks: [
           {
               request: {
                   query: DELETE_POST,
               },
               result: {
                   data: {
                       id: 1,
                       title: 'deleted title',
                   },
               },
           },
       ],
   },
};


/client/src/components/PostCard/PostCard.styles.tsx

import styled from 'styled-components';

export const Container = styled.div`
   display: flex;
   flex-direction: column;
   background: #b3e0e0;
   color: #033535;
   border-radius: 20px;
   padding: 12px 88px 12px 20px;
   width: 100%;
   position: relative;
   min-height: 92px;
`;

export const Title = styled.h3`
   margin: 0 0 12px 0;
```

```
   font-size: 24px;
   color: white;
   text-shadow: 0 0 8px #033535;
`;

export const Content = styled.p`
   margin: 0;
   font-size: 20px;
   word-break: break-all;
`;


/client/src/components/PostCard/PostCard.tsx

import React, { useCallback, useContext } from 'react';
import { useMutation } from '@apollo/client';
import { Trash } from 'assets';
import { Context } from 'context';
import { DELETE_POST } from 'GraphQl/mutations/posts';

import { DeleteButton } from 'components/DeleteButton';

import { Container, Content, Title } from './PostCard.styles';
import { IPostCardProps } from './PostCard.types';

export const PostCard: React.FC<IPostCardProps> = ({ post: { title, content, id, author }
}) => {
   const [deletePost] = useMutation(DELETE_POST);

   const onDelete = useCallback(
       () =>
           deletePost({
               variables: {
                   id,
               },
           }),
       [deletePost, id],
   );

   const { userId } = useContext(Context);

   return (
       <Container>
           {(userId === Number(author) || userId === 1) && (
               <DeleteButton onClick={onDelete}>
                   <Trash />
               </DeleteButton>
           )}
           <Title>{title}</Title>
           <Content>{content}</Content>
       </Container>
   );
};
```

```
/client/src/components/PostCard/index.ts

export { PostCard } from './PostCard';


/client/src/components/PostCard/PostCard.types.ts

import { IPost } from 'types/Post.types';

export interface IPostCardProps {
   post: IPost;
}


/client/src/components/Spinner/Spinner.tsx

import React from 'react';
import { Axoops } from 'assets';

import { IconContainer } from './Spinner.styles';

export const Spinner = React.memo(() => (
   <IconContainer>
       <img src={Axoops} alt="don_vadimon" />
   </IconContainer>
));


/client/src/components/Spinner/Spinner.styles.tsx

import styled, { keyframes } from 'styled-components';

const Spin = keyframes`
   from {
       transform: rotate(0deg);
   }
   to {
       transform: rotate(360deg);
   }
`;

export const IconContainer = styled.div`
   height: 400px;
   width: 400px;
   pointer-events: none;
   animation: ${Spin} infinite 10s linear;

   img {
       width: inherit;
       height: inherit;
   }
`;


/client/src/components/Spinner/Spinner.stories.tsx
```

```
import React from 'react';
import { Meta, Story } from '@storybook/react/types-6-0';

import { Spinner } from './Spinner';

export default {
   title: 'Components/Atoms/Spinner',
   component: Spinner,
   argTypes: {},
} as Meta;

export const Default: Story = (args) => <Spinner {...args} />;
```

/client/src/components/Spinner/index.ts

```
export { Spinner } from './Spinner';
```

/client/src/components/StyledLink/StyledLink.tsx

```
import styled from 'styled-components';

export const StyledLink = styled.a`
   margin: 0;
   width: fit-content;
   color: #61dafb;
   font-size: 24px;
   text-decoration: underline;
`;
```

/client/src/components/StyledLink/index.ts

```
export { StyledLink } from './StyledLink';
```

/client/src/components/Avatar/Avatar.tsx

```
import styled from 'styled-components';

export const Avatar = styled.img`
   width: ${({ width }) => width || '100px'};
   height: ${({ height }) => height || '100px'};
   border-radius: 50%;
   object-fit: cover;
   cursor: pointer;
`;
```

/client/src/components/Avatar/index.ts

```
export { Avatar } from './Avatar';
```

```
/client/src/components/Input/Input.tsx

import { IErrorStyles } from 'global.styles';
import styled from 'styled-components';

export const Input = styled.input<IErrorStyles>`
   background: white;
   border-width: 1px;
   border-style: solid;
   border-color: ${({ error }) => (error ? 'tomato' : 'white')};
   box-shadow: ${({ error }) => (error ? '0 0 8px tomato' : 'none')};
   outline: none;
   padding: 8px;
   border-radius: 8px;
   width: 100%;
   height: 40px;
   font-size: 20px;
   color: teal;
`;


/client/src/components/Input/Input.stories.tsx

import React from 'react';
import { Meta, Story } from '@storybook/react/types-6-0';
import { IErrorStyles } from 'global.styles';

import { Input } from './Input';

export default {
   title: 'Components/Atoms/Input',
   component: Input,
   argTypes: {},
} as Meta;

export const Default: Story<IErrorStyles> = (args) => (
   <div style={{ background: 'teal', padding: 100 }}>
       <Input {...args} />
   </div>
);

export const Error: Story<IErrorStyles> = (args) => (
   <div style={{ background: 'teal', padding: 100 }}>
       <Input {...args} />
   </div>
);
Error.args = {
   error: true,
};


/client/src/components/Input/index.ts

export { Input } from './Input';
```

```
/client/src/components/Button/Button.tsx

import styled from 'styled-components';

export const Button = styled.button`
   background: #00fdbe;
   padding: 12px;
   font-size: 24px;
   color: white;
   border-radius: 8px;
   border: none;
   outline: none;
   transition: all linear 100ms;
   cursor: pointer;

   &:hover {
       background: #1ad1a3;
   }
`;


/client/src/components/Button/Button.stories.tsx

import React from 'react';
import { Meta, Story } from '@storybook/react/types-6-0';

import { Button } from './Button';

export default {
   title: 'Components/Atoms/Button',
   component: Button,
   argTypes: {},
} as Meta;

export const Default: Story = (args) => <Button {...args}>Button</Button>;


/client/src/components/Button/index.ts

export { Button } from './Button';


/client/src/components/ServerErrorMessage/ServerErrorMessage.styles.tsx

import styled from 'styled-components';

export const Container = styled.div`
   position: fixed;
   top: 0;
   left: 0;
   background: #282c34;
   color: white;
   display: flex;
```

```
   justify-content: center;
   align-items: center;
   width: 100vw;
   height: 100vh;

   h1 {
       font-size: 60px;
   }
`;


/client/src/components/ServerErrorMessage/ServerErrorMessage.tsx

import React from 'react';

import { Container } from './ServerErrorMessage.styles';

export const ServerErrorMessage = React.memo(() => (
   <Container>
       <h1>500 | Server Error</h1>
   </Container>
));


/client/src/components/ServerErrorMessage/index.ts

export { ServerErrorMessage } from './ServerErrorMessage';


/client/src/components/Counter/Counter.types.ts

export interface ICounterProps {
   value: number;
   onPlusClick: () => void;
   onMinusClick: () => void;
   error?: boolean;
}


/client/src/components/Counter/Counter.tsx

import React from 'react';
import { Minus, Plus } from 'assets';

import { Button, Container, ValueContainer } from './Counter.styles';
import { ICounterProps } from './Counter.types';

export const Counter: React.FC<ICounterProps> = ({ value, onPlusClick, onMinusClick, error
= false }) => {
   return (
       <Container>
           <Button type="button" onClick={onMinusClick} rounded="left">
               <Minus />
           </Button>
           <ValueContainer error={error}>{value}</ValueContainer>
```

```
            <Button type="button" onClick={onPlusClick} rounded="right">
                <Plus />
            </Button>
        </Container>
    );
};


/client/src/components/Counter/Counter.styles.tsx

import { IErrorStyles } from 'global.styles';
import styled from 'styled-components';

export const Container = styled.div`
    display: flex;
    justify-content: space-between;
    align-items: center;
`;

export const Button = styled.button<{ rounded: 'left' | 'right' }>`
    background: white;
    border: 1px solid teal;
    border-radius: ${({ rounded }) => (rounded === 'left' ? '8px 0 0 8px' : '0 8px 8px 0')};
    color: teal;
    &:hover {
        background: teal;
        color: white;
    }
    padding: 12px;
    height: 60px;
    display: flex;
    justify-content: center;
    align-items: center;
    transition: all linear 100ms;
    cursor: pointer;

    svg {
        width: 24px;
        height: 24px;
    }
`;

export const ValueContainer = styled.div<IErrorStyles>`
    width: 100px;
    height: 60px;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 32px;
    background: white;
    color: ${({ error }) => (error ? 'tomato' : 'teal')};
    border-bottom: 1px solid teal;
    border-top: 1px solid teal;
    user-select: none;
`;
```

```
/client/src/components/Counter/index.ts

export { Counter } from './Counter';


/client/src/components/UserCard/UserCard.types.ts

import { IUser } from 'types/User.types';

export interface IUserCardProps {
   user: IUser;
}


/client/src/components/UserCard/UserCard.tsx

import React, { useCallback, useContext } from 'react';
import { useMutation } from '@apollo/client';
import { Trash } from 'assets';
import { Context } from 'context';
import { DELETE_USER } from 'GraphQl/mutations/user';

import { Avatar } from 'components/Avatar';
import { DeleteButton } from 'components/DeleteButton';
import { ServerErrorMessage } from 'components/ServerErrorMessage';

import { Container } from './UserCard.styles';
import { IUserCardProps } from './UserCard.types';

export const UserCard: React.FC<IUserCardProps> = ({ user: { username, age, id, avatar } })
=> {
   const { userId, setUserId } = useContext(Context);

   const [deleteUser, { error }] = useMutation(DELETE_USER);

   const onDelete = useCallback(
       () =>
           deleteUser({
               variables: {
                   id,
               },
           })
               .then(() => (id === userId ? setUserId(-1) : undefined))
               .catch((err) => console.error(err)),
       [deleteUser, id, setUserId, userId],
   );

   return error ? (
       <ServerErrorMessage />
   ) : (
       <Container>
           <Avatar src={avatar} alt={username} />
           <h1>
```

```
                {username}, {age} лет. Пошлый.
            </h1>
            {userId === 1 && Number(id) !== 1 && (
                <DeleteButton onClick={onDelete}>
                    <Trash />
                </DeleteButton>
            )}
        </Container>
    );
};
```

/client/src/components/UserCard/index.ts

```
export { UserCard } from './UserCard';
```

/client/src/components/UserCard/UserCard.styles.tsx

```
import styled from 'styled-components';

export const Container = styled.div`
    background: teal;
    width: 100%;
    max-width: 800px;
    height: 200px;
    color: white;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 40px;
    border-radius: 20px;
    position: relative;

    h1 {
        font-size: 32px;
        margin-left: 12px;
    }
`;
```

/client/src/components/UserList/UserList.styles.ts

```
import styled from 'styled-components';

export const ListItem = styled.div`
    margin: 12px 0;
`;
```

/client/src/components/UserList/index.ts

```
export { UserList } from './UserList';
```

```
/client/src/components/UserList/UserList.tsx

import React from 'react';
import { useQuery } from '@apollo/client';
import { GET_ALL_USERS, IGetAllUsersResp } from 'GraphQl/queries/user';

import { ServerErrorMessage } from 'components/ServerErrorMessage';
import { Spinner } from 'components/Spinner';
import { UserCard } from 'components/UserCard';

import { ListItem } from './UserList.styles';

export const UserList = () => {
   const { data, loading, error } = useQuery<IGetAllUsersResp>(GET_ALL_USERS, {
pollInterval: 1000 });

   return (
       <div>
           {error && <ServerErrorMessage />}
           {loading ? (
               <Spinner />
           ) : (
               <div>
                   {data?.getAllUsers.map((user, index) => (
                       <ListItem key={index}>
                           <UserCard user={user} />
                       </ListItem>
                   ))}
               </div>
           )}
       </div>
   );
};


/client/src/components/LoginForm/LoginForm.stories.tsx

import React from 'react';
import { Meta, Story } from '@storybook/react/types-6-0';
import { Container } from 'App/App.styles';
import { LOGIN } from 'GraphQl/mutations/user';

import { LoginForm } from './LoginForm';

export default {
   title: 'Components/Atoms/LoginForm',
   component: LoginForm,
   argTypes: {},
} as Meta;

export const Default: Story = () => (
   <Container>
       <LoginForm />
   </Container>
);
```

```
Default.parameters = {
   apolloClient: {
       mocks: [
           {
               request: {
                   query: LOGIN,
                   variables: {
                       username: 'User',
                       password: '123123',
                   },
               },
               result: {
                   data: {
                       username: 'User',
                       age: 20,
                       id: 1,
                   },
               },
           },
       ],
   },
};


/client/src/components/LoginForm/LoginForm.styles.tsx

import styled from 'styled-components';

export const Container = styled.div`
   width: 100%;
   display: flex;
   flex-direction: column;
   align-items: center;
`;

export const FormItem = styled.div`
   width: 100%;
   max-width: 800px;
   padding: 12px;
   display: flex;
   justify-content: center;
`;


/client/src/components/LoginForm/LoginForm.tsx

import React, { useCallback, useContext } from 'react';
import { Field, Form } from 'react-final-form';
import { useHistory } from 'react-router';
import { useMutation } from '@apollo/client';
import { Context } from 'context';
import { FormApi } from 'final-form';
import { LOGIN } from 'GraphQl/mutations/user';
import { setCookie } from 'tools/setCookie';
import { IUser, LoginInput } from 'types/User.types';
```

```
import { Button } from 'components/Button';
import { Input } from 'components/Input';
import { Spinner } from 'components/Spinner';

import { Container, FormItem } from './LoginForm.styles';

const initialValues: LoginInput = {
   username: '',
   password: '',
};

export const LoginForm = () => {
   const [login, { error, loading }] = useMutation<{ login: IUser }, { input: LoginInput
}>(LOGIN);

   const { setUserId } = useContext(Context);

   const history = useHistory();

   const handleSubmit = useCallback(
       (values: LoginInput, form: FormApi<LoginInput, LoginInput>) => {
           login({
               variables: {
                   input: values,
               },
           })
               .then((data) => {
                   if (data.data?.login && !error) {
                       setCookie('login', data.data?.login.id.toString());
                       setUserId(Number(data.data?.login.id.toString()));
                   }
                   return Number(data.data?.login.id);
               })
               .then((id) => history.push(`/user/${id}`))
               .catch((err) => {
                   form.reset();
                   console.error(err);
               });
       },
       [error, history, login, setUserId],
   );

   return loading ? (
       <Spinner />
   ) : (
       <Form initialValues={initialValues} onSubmit={handleSubmit}>
           {({ handleSubmit }) => (
               <Container>
                   <FormItem>
                       <Field<string> name="username">
                           {({ input, meta }) => (
                               <Input
                                   type="text"
                                   placeholder="Username"
```

```
                                autoComplete="off"
                                {...input}
                                error={!!meta.error}
                            />
                        )}
                    </Field>
                </FormItem>
                <FormItem>
                    <Field<string> name="password">
                        {({ input, meta }) => (
                            <Input
                                type="password"
                                placeholder="Password"
                                autoComplete="off"
                                {...input}
                                error={!!meta.error}
                            />
                        )}
                    </Field>
                </FormItem>
                <FormItem>
                    <Button onClick={handleSubmit}>Login</Button>
                </FormItem>
            </Container>
        )}
    </Form>
  );
};


/client/src/components/LoginForm/index.ts

export { LoginForm } from './LoginForm';


/client/src/components/Textarea/Textarea.tsx

import { IErrorStyles } from 'global.styles';
import styled from 'styled-components';

export const Textarea = styled.textarea<IErrorStyles>`
  width: 100%;
  min-height: 120px;
  resize: none;
  background: white;
  border: 1px solid ${({ error }) => (error ? 'tomato' : 'white')};
  box-shadow: ${({ error }) => (error ? '0 0 8px tomato' : 'none')};
  border-radius: 8px;
  outline: none;
  padding: 12px;
  font-size: 20px;
  caret-color: teal;
  color: teal;
`;
```

```
/client/src/components/Textarea/index.ts

export { Textarea } from './Textarea';


/client/src/components/Textarea/Textarea.stories.tsx

import React from 'react';
import { Meta, Story } from '@storybook/react/types-6-0';
import { IErrorStyles } from 'global.styles';

import { Textarea } from './Textarea';

export default {
   title: 'Components/Atoms/Textarea',
   component: Textarea,
   argTypes: {},
} as Meta;

export const Default: Story<IErrorStyles> = (args) => (
   <div style={{ background: 'teal', padding: 100 }}>
       <Textarea {...args} />
   </div>
);

export const Error: Story<IErrorStyles> = (args) => (
   <div style={{ background: 'teal', padding: 100 }}>
       <Textarea {...args} />
   </div>
);
Error.args = {
   error: true,
};


/client/src/components/Navbar/Navbar.tsx

import React, { useContext } from 'react';
import { useHistory } from 'react-router-dom';
import { useQuery } from '@apollo/client';
import { Context } from 'context';
import { GET_USER_MIN_INFO } from 'GraphQl/queries/user';
import { useLogout } from 'hooks/useLogout';
import { UserMinInfo } from 'types/User.types';

import { Avatar } from 'components/Avatar';
import { ServerErrorMessage } from 'components/ServerErrorMessage';

import { Container, Item, Username } from './Navbar.styles';

export const Navbar = () => {
   const history = useHistory();

   const { userId } = useContext(Context);
```

```
    const logout = useLogout();

    const { data, loading, error } = useQuery<{ getUser: UserMinInfo }>(GET_USER_MIN_INFO, {
        variables: {
            id: userId,
        },
    });

    return (
        <Container>
            {error && <ServerErrorMessage />}
            {!loading && !error && data?.getUser && (
                <Item onClick={() => history.push(`/user/${userId}`)}>
                    <Avatar width="80px" height="80px" src={data.getUser?.avatar} />
                    <Username>{data.getUser.username}</Username>
                </Item>
            )}
            <Item onClick={() => history.push('/')}>Users</Item>
            <Item onClick={() => history.push('/posts')}>Posts</Item>
            {userId < 0 ? (
                <>
                    <Item onClick={() => history.push('/login')}>Login</Item>
                    <Item onClick={() => history.push('/register')}>Register</Item>
                </>
            ) : (
                <Item onClick={logout}>Logout</Item>
            )}
        </Container>
    );
};


/client/src/components/Navbar/Navbar.styles.tsx

import styled from 'styled-components';

export const Container = styled.div`
    display: flex;
    width: 100vw;
    justify-content: space-between;
    align-items: center;
    background: #01cece;
    color: white;
    margin-bottom: 20px;
`;

export const Item = styled.div`
    padding: 8px;
    height: 100px;
    width: 100%;
    background: #01cece;
    transition: all linear 100ms;
    cursor: pointer;
    display: flex;
```

```
   justify-content: center;
   align-items: center;
   font-size: 24px;
   user-select: none;

   &:hover {
       background: #0cb6b6;
   }
`;

export const Username = styled.h3`
   margin: 0 0 0 12px;
`;
```

/client/src/components/Navbar/Navbar.stories.tsx

```
import React from 'react';
import { Meta, Story } from '@storybook/react/types-6-0';

import { Navbar } from './Navbar';

export default {
   title: 'Components/Atoms/Navbar',
   component: Navbar,
   argTypes: {},
} as Meta;

export const Default: Story = () => <Navbar />;
```

/client/src/components/Navbar/index.ts

```
export { Navbar } from './Navbar';
```

/client/src/pages/Index.tsx

```
import React from 'react';

import { StyledLink } from 'components/StyledLink';
import { UserList } from 'components/UserList';

export const Index = () => (
   <>
       <h1>Community</h1>
       <UserList />
       <StyledLink href="https://vk.com/pudge1stpick" target="_blank">
           Creator
       </StyledLink>
   </>
);
```

/client/src/pages/Register.tsx

```
import React from 'react';

import { NewUserForm } from 'components/NewUserForm';

export const Register = () => {
   return (
       <>
           <h1>Register</h1>
           <NewUserForm />
       </>
   );
};
```

/client/src/pages/Posts.tsx

```
import React, { useContext } from 'react';
import { useQuery } from '@apollo/client';
import { Context } from 'context';
import { GET_ALL_POSTS, IGetAllPostsResp } from 'GraphQl/queries/posts';

import { NewPostForm } from 'components/NewPostForm';
import { PostList } from 'components/PostList';
import { ServerErrorMessage } from 'components/ServerErrorMessage';
import { Spinner } from 'components/Spinner';

export const Posts = () => {
   const { userId } = useContext(Context);

   const { data, loading, error } = useQuery<IGetAllPostsResp>(GET_ALL_POSTS, {
pollInterval: 1000 });

   return (
       <>
           <h1>All Posts</h1>
           {userId >= 0 && <NewPostForm />}
           {error && <ServerErrorMessage />}
           {loading ? <Spinner /> : data?.getAllPosts && <PostList posts={data.getAllPosts}
/>}
       </>
   );
};
```

/client/src/pages/UserPage.tsx

```
import React, { useContext } from 'react';
import { useQuery } from '@apollo/client';
import { Context } from 'context';
import { GET_USER_POSTS, IGetUserPostsResp } from 'GraphQl/queries/posts';
import { GET_USER } from 'GraphQl/queries/user';
import { IUser } from 'types/User.types';

import { PostList } from 'components/PostList';
```

```
import { UserCard } from 'components/UserCard';

export const UserPage = () => {
   const { userId } = useContext(Context);

   const { data: userData } = useQuery<{ getUser: IUser }>(GET_USER, {
       variables: {
           id: userId,
       },
   });

   const { data: postsData } = useQuery<IGetUserPostsResp>(GET_USER_POSTS, {
       variables: {
           author: userId,
       },
   });

   return (
       <>
           {userData?.getUser && <UserCard user={userData.getUser} />}
           {postsData?.getUserPosts && <PostList posts={postsData.getUserPosts} />}
       </>
   );
};


/client/src/pages/Login.tsx

import React from 'react';

import { LoginForm } from 'components/LoginForm';

export const Login = () => {
   return (
       <>
           <h1>Login</h1>
           <LoginForm />
       </>
   );
};


/client/src/App/App.styles.tsx

import styled from 'styled-components';

export const Container = styled.div`
   background-color: ${(props) => props.color || '#282c34'};
   min-height: 100vh;
   display: flex;
   flex-direction: column;
   align-items: center;
   justify-content: flex-start;
   font-size: calc(10px + 2vmin);
   color: white;
```

```
`;


/client/src/App/index.ts

export { App } from './App';


/client/src/App/App.tsx

import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
import { Context } from 'context';
import { useUserId } from 'hooks/useUserId';
import { Index } from 'pages/Index';
import { Login } from 'pages/Login';
import { Posts } from 'pages/Posts';
import { Register } from 'pages/Register';
import { UserPage } from 'pages/UserPage';

import { Navbar } from 'components/Navbar';

import { Container } from './App.styles';

export const App = () => {
   const [userId, setUserId] = useUserId();

   return (
       <Context.Provider value={{ userId, setUserId }}>
           <Router>
               <Container>
                   <Navbar />
                   <Switch>
                       <Route exact path="/">
                           <Index />
                       </Route>
                       <Route path="/posts">
                           <Posts />
                       </Route>
                       <Route path="/user/:userid">
                           <UserPage />
                       </Route>
                       <Route path="/login">
                           <Login />
                       </Route>
                       <Route path="/register">
                           <Register />
                       </Route>
                   </Switch>
               </Container>
           </Router>
       </Context.Provider>
   );
};
```

```
/client/src/index.tsx

import React from 'react';
import ReactDOM from 'react-dom';
import { ApolloClient, ApolloProvider, InMemoryCache } from '@apollo/client';

import { App } from './App';
import { Global } from './global.styles';
import reportWebVitals from './reportWebVitals';

const client = new ApolloClient({
   uri:
       process.env.NODE_ENV === 'production'
           ? 'https://graphql-try-server.herokuapp.com/graphql'
           : 'http://localhost:8000/graphql',
   cache: new InMemoryCache(),
});

ReactDOM.render(
   <ApolloProvider client={client}>
       <React.StrictMode>
           <Global />
           <App />
       </React.StrictMode>
   </ApolloProvider>,
   document.getElementById('root'),
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();


/client/src/types/User.types.ts

import { ID } from './shared.types';

export interface IUser {
   id: ID;
   username: string;
   password: string;
   age: number;
   avatar?: string;
}

export type UserInput = Omit<IUser, 'id'>;

export type LoginInput = Omit<UserInput, 'age'>;

export type UserMinInfo = Pick<IUser, 'username' | 'avatar'>;


/client/src/types/shared.types.ts
```

```
export type ID = string | number;


/client/src/types/Post.types.ts

import { ID } from './shared.types';

export interface IPost {
   id: ID;
   author: ID;
   title: string;
   content: string;
}

export type PostInput = Omit<IPost, 'id'>;


/client/src/global.styles.tsx

import { createGlobalStyle } from 'styled-components';

export const Global = createGlobalStyle`
   * {
       margin: 0;
       font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
           'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
           sans-serif;
       -webkit-font-smoothing: antialiased;
       -moz-osx-font-smoothing: grayscale;
       box-sizing: border-box;
   }
`;

export interface IErrorStyles {
   error?: boolean;
}


/client/public/index.html

<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="utf-8" />
   <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
   <meta name="viewport" content="width=device-width, initial-scale=1" />
   <meta name="theme-color" content="#000000" />
   <meta
     name="description"
     content="Web site created using create-react-app"
   />
   <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
   <!--
     manifest.json provides metadata used when your web app is installed on a
```

```
    user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
  -->
  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
  <!--
    Notice the use of %PUBLIC_URL% in the tags above.
    It will be replaced with the URL of the `public` folder during the build.
    Only files inside the `public` folder can be referenced from the HTML.

    Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
    work correctly both with client-side routing and a non-root public URL.
    Learn how to configure a non-root public URL by running `npm run build`.
  -->
  <title>React App</title>
 </head>
 <body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
  <!--
    This HTML file is a template.
    If you open it directly in the browser, you will see an empty page.

    You can add webfonts, meta tags, or analytics to this file.
    The build step will place the bundled scripts into the <body> tag.

    To begin the development, run `npm start` or `yarn start`.
    To create a production bundle, use `npm run build` or `yarn build`.
  -->
 </body>
</html>
```

/client/public/robots.txt

```
# https://www.robotstxt.org/robotstxt.html
User-agent: *
Disallow:
```

/client/public/manifest.json

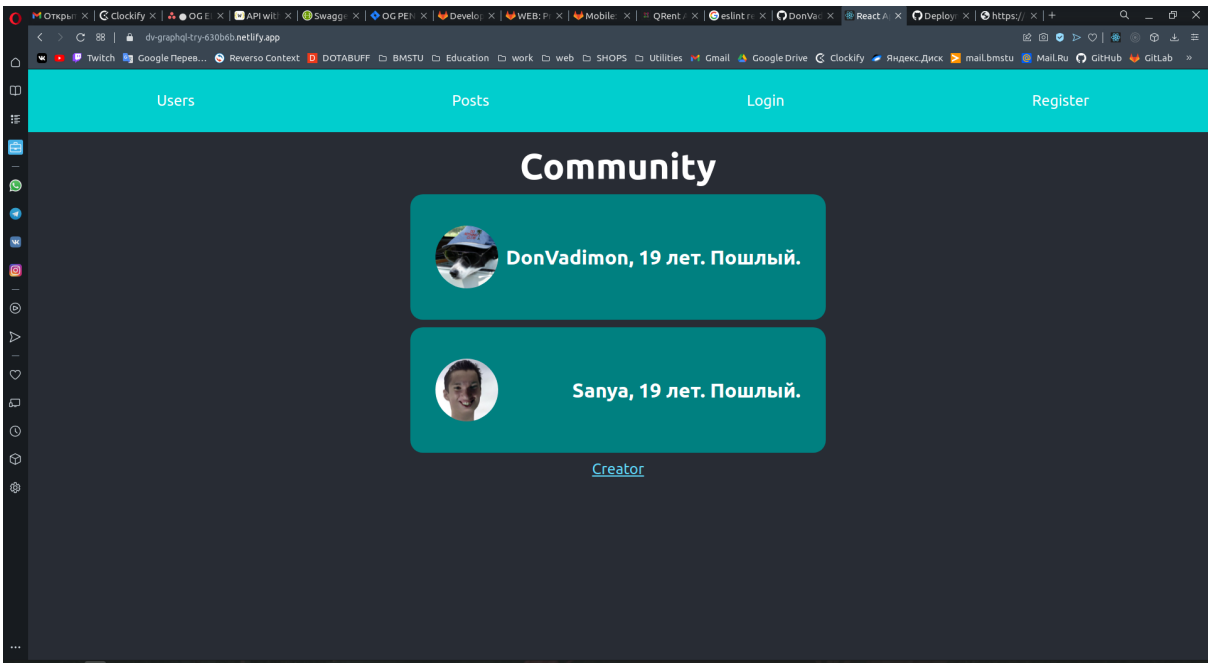
```
{
 "short_name": "React App",
 "name": "Create React App Sample",
 "icons": [
   {
     "src": "favicon.ico",
     "sizes": "64x64 32x32 24x24 16x16",
     "type": "image/x-icon"
   },
   {
     "src": "logo192.png",
     "type": "image/png",
     "sizes": "192x192"
   },
```

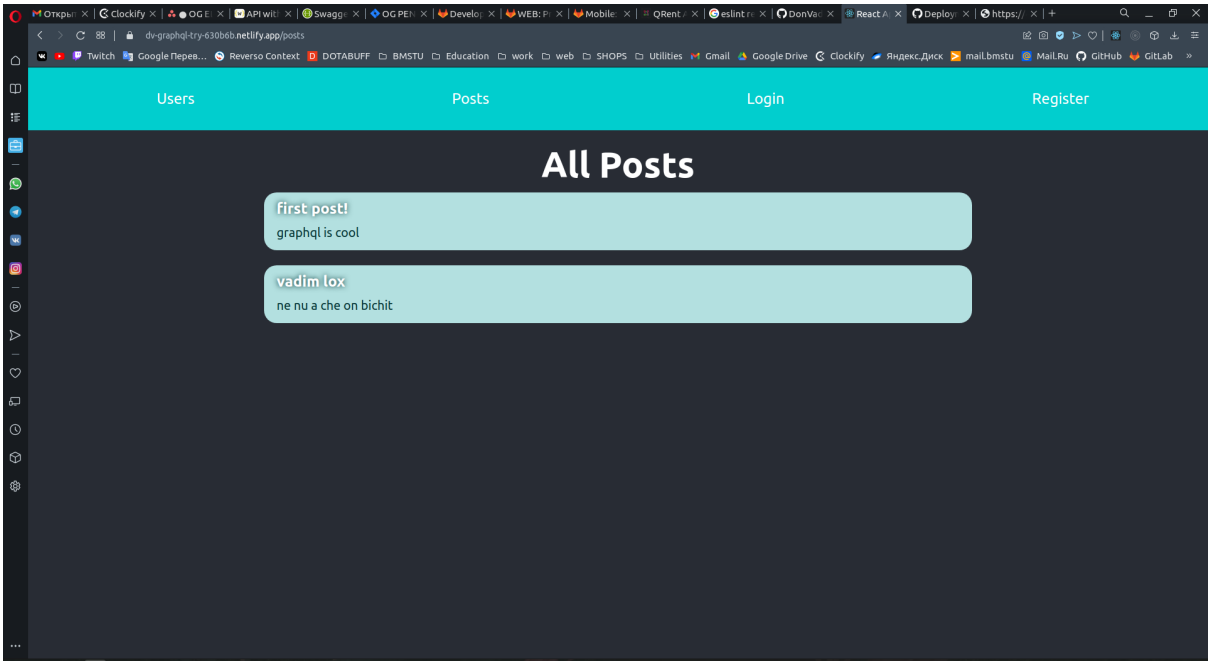
```
   {
     "src": "logo512.png",
     "type": "image/png",
     "sizes": "512x512"
   }
 ],
 "start_url": ".",
 "display": "standalone",
 "theme_color": "#000000",
 "background_color": "#ffffff"
}


/client/.eslintrc

{
   "env": {
       "browser": true,
       "es2020": true
   },
   "plugins": ["@typescript-eslint", "react", "standard", "react-hooks", "import",
"simple-import-sort"],
   "extends": [
       "prettier",
       "eslint:recommended",
       "plugin:react/recommended",
       "plugin:@typescript-eslint/eslint-recommended",
       "plugin:@typescript-eslint/recommended",
       "plugin:@typescript-eslint/recommended-requiring-type-checking",
       "plugin:import/errors",
       "plugin:import/warnings",
       "plugin:import/typescript",
       "plugin:eslint-comments/recommended",
       "plugin:react-hooks/recommended"
   ],
   "parser": "@typescript-eslint/parser",
   "parserOptions": {
       "project": "tsconfig.json",
       "tsconfigRootDir": "."
   },
   "rules": {
       "eslint-comments/no-unused-disable": "error",
       "eslint-comments/no-use": "error",
       "no-debugger": "error",
       "prefer-const": "error",
       "react-hooks/rules-of-hooks": "error",
       "react-hooks/exhaustive-deps": "error",
       "no-console": ["warn", { "allow": ["error"] }],
       "no-unused-expressions": "error",
       "react/prop-types": "off",
       "react/display-name": "off",
       "@typescript-eslint/explicit-function-return-type": "off",
       "@typescript-eslint/no-empty-function": "error",
       "@typescript-eslint/no-empty-interface": "error",
       "@typescript-eslint/no-explicit-any": "error",
```

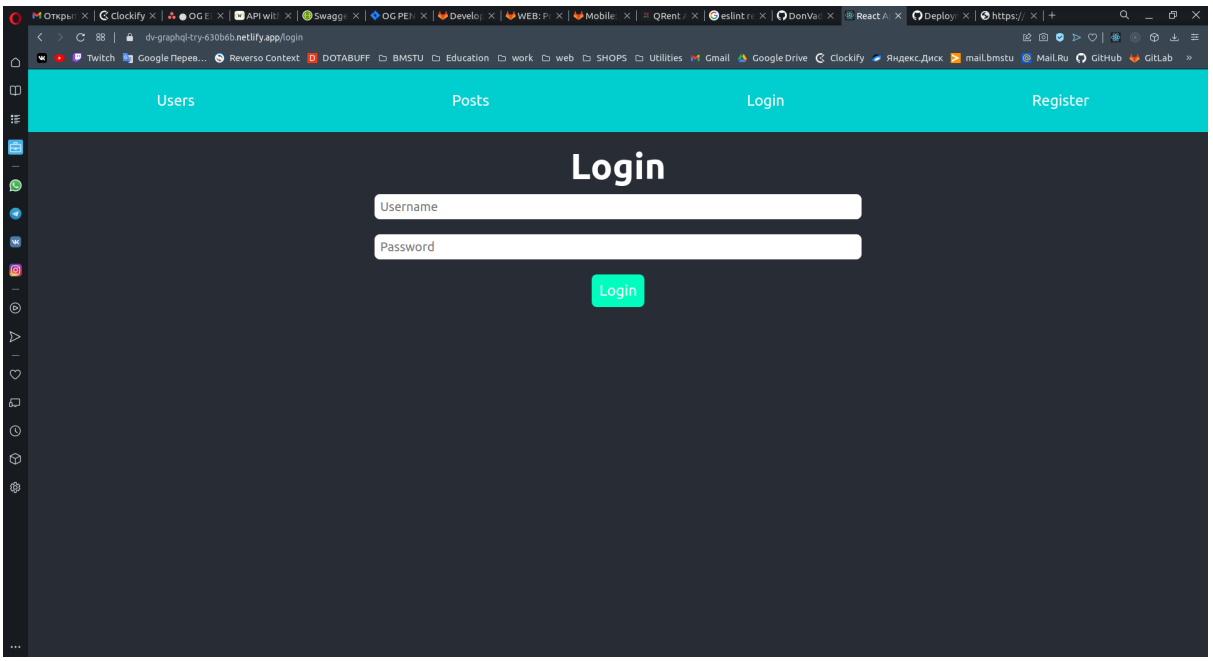
```
      "@typescript-eslint/explicit-module-boundary-types": "off",
      "@typescript-eslint/no-unused-vars": "error",
      "import/no-relative-parent-imports": "error",
      "import/no-duplicates": "error",
      "import/no-unresolved": "off",
      "simple-import-sort/exports": "error",
      "simple-import-sort/imports": [
          "error",
          {
              "groups": [
                  // Node.js builtins. You could also generate this regex if you use a
`.js` config.
                  // For example: `^(${require("module").builtinModules.join("|")})(/|$)`
                  [

"^(assert|buffer|child_process|cluster|console|constants|crypto|dgram|dns|domain|events|fs|
http|https|module|net|os|path|punycode|querystring|readline|repl|stream|string_decoder|sys|
timers|tls|tty|url|util|vm|zlib|freelist|v8|process|async_hooks|http2|perf_hooks)(/.*|$)"
                  ],
                  // Packages. `react` related packages come first.
                  ["^react", "^@?\\w"],
                  // Internal packages.
                  ["^(@|@company|@ui|components|utils|config|vendored-lib)(/.*|$)"],
                  // Side effect imports.
                  ["^\\u0000"],
                  // Parent imports. Put `..` last.
                  ["^\\.\\.(?!/?$)", "^\\.\\./?$"],
                  // Other relative imports. Put same-folder imports and `.` last.
                  ["^\\./(?=.*/)(?!/?$)", "^\\.(?!/?$)", "^\\./?$"],
                  // Style imports.
                  ["^.+\\.s?css$"]
              ]
          }
      ]
   },
   "settings": {
      "react": {
          "version": "detect"
      }
   }
}


/.graphqlconfig

{
 "schemaPath": "./server/schema/schema.graphql"
}


/.prettierrc

{
   "arrowParens": "always",
   "bracketSpacing": true,
```

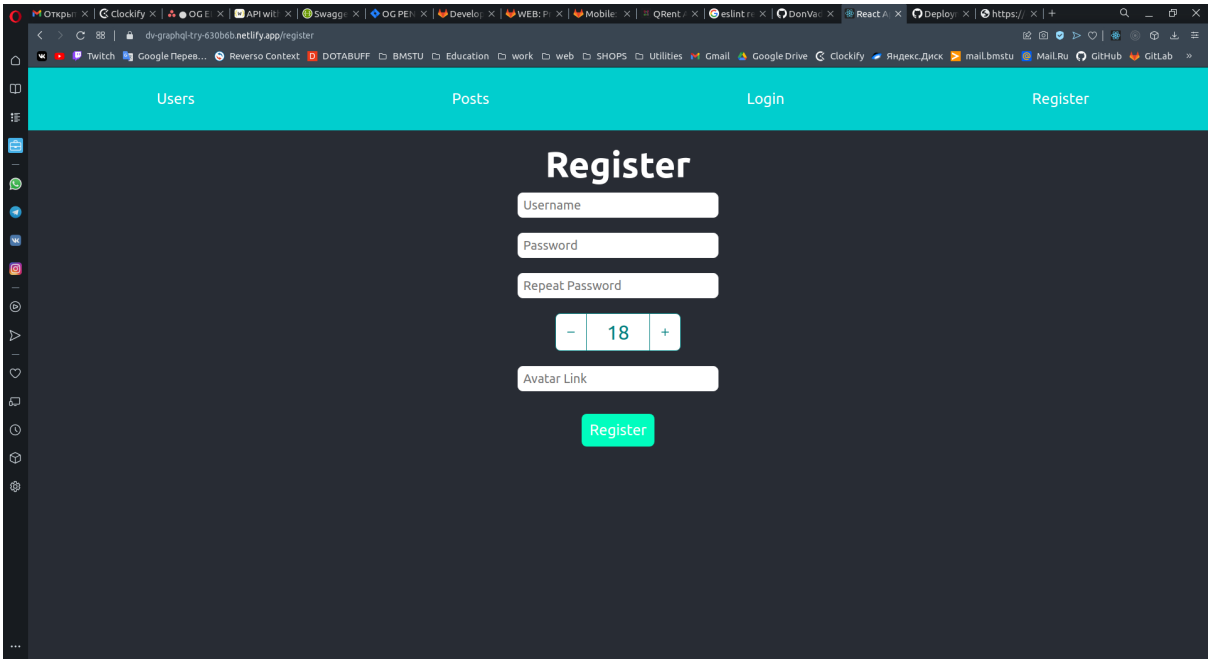
```
    "endOfLine": "auto",
    "jsxBracketSameLine": false,
    "jsxSingleQuote": false,
    "parser": "typescript",
    "printWidth": 120,
    "proseWrap": "preserve",
    "semi": true,
    "singleQuote": true,
    "tabWidth": 4,
    "trailingComma": "all",
    "useTabs": false,
    "overrides": [
        {
            "files": ["**/*.json", ".prettierrc", ".eslintrc", "**/*.(graphql|gql)"],
            "options": {
                "parser": "json"
            }
        },
        {
            "files": ["**/*.css"],
            "options": {
                "parser": "css"
            }
        }
    ]
}
```

Результат работы программы:


Users
Posts
Login
Register
Community
DonVadimon, 19 лет. Пошлый.
Sanya, 19 лет. Пошлый.
Creator


Users
Posts
Login
Register
All Posts
first post!
graphql is cool
vadim lox
ne nu a che on bichit


Users
Posts
Login
Register
Login
Username
Password
Login


Users
Posts
Login
Register
Register
Username
Password
Repeat Password
18
Avatar Link
Register