

Отчет по лабораторной работе № 8 по курсу
"Разработка Интернет-Приложений"

Выполнил:
Студент группы
ИУ5-55Б
Хижняков Вадим Максимович

Москва, МГТУ – 2021

Задание:

На основе методических указаний разработайте React-приложение. Для создания приложения необходимо решить следующие задачи:

1. Создать стартовый React-проект. Удалить неиспользуемый код. Организовать директории для страниц, компонентов, утилит и работы с сетью.
2. Организовать роутинг в веб-приложении.
3. Разработать базовые страницы, на которых будут отображаться сущности из выбранной вами предметной области:
 - a. Стартовая страница.
 - b. Страница просмотра списка объектов.
 - c. Страница просмотра конкретного объекта.
4. Вынести переиспользуемые компоненты в отдельные файлы:
 - a. Для навигации по приложению можно добавить header.
 - b. Для отображения дополнительной информации (данные о студенте и предметной области) можно использовать footer.
 - c. Источники ввода-вывода (поля ввода (inputs)/формы/текстовые блоки).
 - d. Переиспользуемые таблицы/гриды.
5. Добавить асинхронные запросы в разработанный API, чтобы страница получала данные с сервера.
6. Если в Вашем проекте реализована сложная логика работы с состоянием приложения, то рекомендуется добавить пользовательские хуки.
7. Страницы приложения должны хорошо отображаться как на больших, так и на маленьких экранах.

Текст программы:

Dockerfile

```
FROM node:12 as build-deps
WORKDIR /usr/src/app
COPY package.json yarn.lock ./
RUN yarn
COPY . ./
RUN yarn build

FROM nginx:1.12-alpine
COPY --from=build-deps /usr/src/app/build /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Dockerfile.dev

```
FROM node:12 as build-deps
WORKDIR /usr/src/app
COPY package.json yarn.lock ./
RUN yarn
COPY . ./
CMD ["yarn", "start"]
```

src/config.ts

```
export const BACKEND_URI = process.env.REACT_APP_BACKEND_URI as string;
```

src/types/routes.types.ts

```
export interface Route {
  (...pathSegments: (string | number)[]): string;
}

export type APIRoutes<T> = {
  [K in keyof T]: Route;
};
```

src/types/cars.types.ts

```
export type Car = {
  pk: number;
  name: string;
  price: number;
  description: string;
  main_image?: string;
  score: number;
  manufacturer: number;
```

```
};

export type CreateCarDto = Omit<Car, 'pk'>;

export type UpdateCarDto = Partial<Car>;
```

src/api/axios.ts

```
import axios from 'axios';

import { BACKEND_URI } from 'config';

export const axiosInstance = axios.create({
  baseURL: BACKEND_URI,
  withCredentials: true,
});
```

src/api/car/routes.ts

```
import { Car, CreateCarDto, UpdateCarDto } from 'types/cars.types';
import { APIRoutes } from 'types/routes.types';

export interface CarsApi {
  getAll: () => Promise<Car[]>;
  create: (dto: CreateCarDto) => Promise<Car>;
  deleteByPk: (pk: number) => Promise<Car>;
  updateByPk: (pk: number, dto: UpdateCarDto) => Promise<Car>;
}

export const CAR_PREFIX = '/cars';

export const CAR_ROUTES: APIRoutes<CarsApi> = {
  getAll: () => `${CAR_PREFIX}/`,
  create: () => `${CAR_PREFIX}/`,
  deleteByPk: (pk = ':pk') => `${CAR_PREFIX}/${pk}/`,
  updateByPk: (pk = ':pk') => `${CAR_PREFIX}/${pk}/`,
};
```

src/api/car/fetchAllCars.ts

```
import { axiosInstance } from 'api/axios';
import { Car } from 'types/cars.types';

import { CAR_ROUTES } from './routes';

export const fetchAllCars = (): Promise<Car[]> =>
  axiosInstance.get<Car[]>(CAR_ROUTES.getAll()).then((response) => response.data);
```

src/api/car/fetchCreateCar.ts

```
import { axiosInstance } from 'api/axios';
```

```
import { Car, CreateCarDto } from 'types/cars.types';

import { CAR_ROUTES } from './routes';

export const fetchCreateCar = (dto: CreateCarDto): Promise<Car> =>
  axiosInstance.post<Car>(CAR_ROUTES.create(), dto).then((response) => response.data);
```

src/api/car/fetchDeleteCarByPk.ts

```
import { axiosInstance } from 'api/axios';
import { Car } from 'types/cars.types';

import { CAR_ROUTES } from './routes';

export const fetchDeleteCarByPk = (pk: number): Promise<Car> =>
  axiosInstance.delete<Car>(CAR_ROUTES.deleteByPk(pk)).then((response) => response.data);
```

src/api/car/fetchUpdateCarByPk.ts

```
import { axiosInstance } from 'api/axios';
import { Car, UpdateCarDto } from 'types/cars.types';

import { CAR_ROUTES } from './routes';

export const fetchUpdateCarByPk = (pk: number, dto: UpdateCarDto): Promise<Car> =>
  axiosInstance.put<Car>(CAR_ROUTES.updateByPk(pk), dto).then((response) =>
response.data);
```

src/api/car/car.service.ts

```
import { Car, CreateCarDto, UpdateCarDto } from 'types/cars.types';

import { fetchAllCars } from './fetchAllCars';
import { fetchCreateCar } from './fetchCreateCar';
import { fetchDeleteCarByPk } from './fetchDeleteCarByPk';
import { fetchUpdateCarByPk } from './fetchUpdateCarByPk';
import { CarsApi } from './routes';

class CarsService implements CarsApi {
  create(dto: CreateCarDto): Promise<Car> {
    return fetchCreateCar(dto);
  }

  deleteByPk(pk: number): Promise<Car> {
    return fetchDeleteCarByPk(pk);
  }

  updateByPk(pk: number, dto: Partial<Car>): Promise<Car> {
    return fetchUpdateCarByPk(pk, dto);
  }
}
```

```

    getAll(): Promise<Car[]> {
        return fetchAllCars();
    }

    update(pk: number, dto: UpdateCarDto): Promise<Car> {
        return fetchUpdateCarByPk(pk, dto);
    }
}

export const carsService = new CarsService();

```

context/index.ts

```

import React, { createContext } from 'react';
import { Car } from 'types/cars.types';
import { Manufacturer } from 'types/manufacturers.types';

export type AppContext = {
    cars: Car[];
    manufacturers: Manufacturer[];
    loaded: boolean;
    setLoaded: React.Dispatch<React.SetStateAction<boolean>>;
    error: boolean;
};

export const AppContext = createContext<AppContext>({
    cars: [],
    manufacturers: [],
    loaded: false,
    setLoaded: () => undefined,
    error: false,
});

```

hooks/useAppContext.ts

```

import { useContext } from 'react';
import { AppContext } from 'context';

export const useAppContext = () => useContext(AppContext);

```

src/components/CarCard/CarCard.types.ts

```

import { Car } from 'types/cars.types';

export type CarCardProps = {
    car: Car;
};

```

src/components/CarCard/CarCard.tsx

```
import React, { useCallback, useMemo } from 'react';
import { useHistory } from 'react-router';
import { carsService } from 'api/car/car.service';
import { zhiga } from 'assets';
import cnBind, { Argument } from 'classnames/bind';
import { useAppContext } from 'hooks/useAppContext';

import { CarCardProps } from './CarCard.types';

import styles from './CarCard.module.css';

const cx = cnBind.bind(styles) as (...args: Argument[]) => string;

export const CarCard = ({ car }: CarCardProps): JSX.Element => {
  const history = useHistory();

  const onClick = useCallback(() => history.push(`/car/${car.pk}`), [car.pk, history]);

  const { manufacturers, setLoaded } = useAppContext();

  const manufacturer = useMemo(
    () => manufacturers.find(({ pk }) => pk === car.manufacturer),
    [car.manufacturer, manufacturers],
  );

  const onDelete = useCallback(
    () => carsService.deleteByPk(car.pk).then(() => setLoaded(false)),
    [car.pk, setLoaded],
  );

  return (
    <div className={cx('container')}>
      <img className={cx('image')} src={car?.main_image ?? zhiga} alt={car.name}
onClick={onClick} />
      <div className={cx('info')}>
        <h1>{car.name}</h1>
        <h2>Производитель {manufacturer?.name}</h2>
        <p className={cx('description')}>{car.description}</p>
        <p className={cx('score')}>Оценка экспертов: {car.score} из 10</p>
        <button type="button" onClick={onDelete}>
          Delete
        </button>
      </div>
    </div>
  );
};
```

src/components/CarCard/CarCard.module.css

```
.container {
  border: 1px solid black;
  margin: 12px;
```

```

display: flex;
flex-direction: column;
border-radius: 32px;
overflow: hidden;
cursor: pointer;
}

.image {
  width: 100%;
}

.info {
  margin-left: 0;
  padding: 32px;
}

.info h1 {
  margin: 0;
}

@media screen and (min-width: 768px) {
  .container {
    flex-direction: row;
    flex-wrap: wrap;
  }

  .image {
    width: 320px;
  }

  .info {
    margin-left: 32px;
  }
}

.container button {
  padding: 8px;
  margin-top: 20px;
  font-size: 1.2em;
  background: tomato;
  color: white;
  border: 2px solid tomato;
  border-radius: 12px;
  cursor: pointer;
}

```

src/components/CarCard/index.ts

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


Далее буду приведены только файлы с расширением .tsx

src/components/CarForm/CarForm.tsx

```
import React, { useCallback } from 'react';
import { Field, Form } from 'react-final-form';
import { useHistory } from 'react-router-dom';
import { carsService } from 'api/car/car.service';
import cnBind, { Argument } from 'classnames/bind';
import { useAppContext } from 'hooks/useAppContext';
import { CreateCarDto } from 'types/cars.types';

import { Input } from 'components/Input';

import { CarFormProps } from './CarForm.types';

import styles from './CarForm.module.css';

const cx = cnBind.bind(styles) as (...args: Argument[]) => string;

export const CarForm = ({ initialValues, pk }: CarFormProps) => {
  const { manufacturers, setLoaded } = useAppContext();

  const history = useHistory();

  const onSubmit = useCallback(
    (values: CreateCarDto) => {
      (pk ? carsService.update(pk, values) : carsService.create(values))
        .then(() => setLoaded(false))
        .then(() => history.push('/'));
    },
    [history, pk, setLoaded],
  );

  return (
    <Form<CreateCarDto> onSubmit={onSubmit} initialValues={{ ...initialValues,
main_image: undefined }}>
      {({ handleSubmit }) => (
        <div className={cx('container')}>
          <p>name</p>
          <Input required name='name' />
          <p>price</p>
          <Input type="number" required name='price' />
          <p>description</p>
          <Input required name='description' />
          <p>score</p>
          <Input required name='score' />
          <p>manufacturer</p>
          <Field type="select" name="manufacturer">
            {({ input }) => (
              <select {...input}>
                {manufacturers.map(({ name, pk }) => (
                  <option key={pk} value={pk}>
                    {name}
                  </option>
                ))}
              </select>
            )}
          </Field>
        </div>
      )}
    </Form>
  );
}
```

```

        ))}
        </select>
      )}
    </Field>
    <button type="button" onClick={handleSubmit}>
      Submit
    </button>
  </div>
)}
</Form>
);
};

```

src/components/Input/Input.tsx

```

import React from 'react';
import { Field } from 'react-final-form';
import cnBind, { Argument } from 'classnames/bind';

import { InputProps } from '../Input.types';

import styles from './Input.module.css';

const cx = cnBind.bind(styles) as (...args: Argument[]) => string;

export const Input = ({ className, name, required = false, ...props }: InputProps) => (
  <Field name={name} validate={(value) => (!value && required ? 'Required' : undefined)}>
    {( { input } ) => <input {...input} className={cx('input', className)} {...props} />}
  </Field>
);

```

src/components/Spinner/Spinner.tsx

```

import React from 'react';
import { spinner } from 'assets';
import cnBind, { Argument } from 'classnames/bind';

import styles from './Spinner.module.css';

const cx = cnBind.bind(styles) as (...args: Argument[]) => string;

export const Spinner = React.memo(() => (
  <div className={cx('spinner')}>
    <img src={spinner} alt="loading" />
  </div>
));

```

src/pages/MainPage.tsx

```

import React from 'react';
import { useAppContext } from 'hooks/useAppContext';

```

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

export const MainPage = () => {
  const { cars } = useAppContext();

  return (
    <div>
      {cars.map((car) => (
        <CarCard key={`_${car.pk}`} car={car} />
      ))}
    </div>
  );
};
```

src/pages/Details.tsx

```
import React, { useMemo } from 'react';
import { useParams } from 'react-router';
import { useAppContext } from 'hooks/useAppContext';

import { CarCard } from 'components/CarCard';
import { CarForm } from 'components/CarForm';

export const Details = () => {
  const { pk } = useParams<{ pk: string }>();

  const { cars } = useAppContext();

  const car = useMemo(() => cars.find(({ pk: _pk }) => _pk === Number(pk)), [cars, pk]);

  return (
    <div>
      {car ? (
        <div>
          <CarCard car={car} />
          <CarForm initialValues={car} pk={car.pk} />
        </div>
      ) : (
        <CarForm />
      )}
    </div>
  );
};
```

src/App/App.tsx

```
import React, { useCallback, useEffect, useState } from 'react';
import { Route, Switch } from 'react-router';
import { BrowserRouter, Link } from 'react-router-dom';
import { carsService } from 'api/car/car.service';
import { manufacturersService } from 'api/manufacturers/manufacturers.service';
```

```

import { AppContext } from 'context';
import { Details } from 'pages/Details';
import { MainPage } from 'pages/MainPage';
import { Car } from 'types/cars.types';
import { Manufacturer } from 'types/manufacturers.types';

import { Spinner } from 'components/Spinner';

import './App.css';

export const App = () => {
  const [cars, setCars] = useState<Car[]>([]);
  const [manufacturers, setManufacturers] = useState<Manufacturer[]>([]);
  const [loaded, setLoaded] = useState(false);
  const [error, setError] = useState(false);

  const handleError = useCallback((error: Error) => {
    console.error(error);
    setError(true);
  }, []);

  useEffect(() => {
    if (!loaded) {
      carsService
        .getAll()
        .then((cars) => setCars(cars))
        .catch(handleError);
    }
  }, [handleError, loaded]);

  useEffect(() => {
    if (!loaded) {
      manufacturersService
        .getAll()
        .then((manufacturers) => setManufacturers(manufacturers))
        .catch(handleError);
    }
  }, [handleError, loaded]);

  useEffect(() => setLoaded(!!cars.length && !!manufacturers.length), [cars, manufacturers]);

  return (
    <AppContext.Provider value={{ cars, manufacturers, loaded, setLoaded, error }}>
      <BrowserRouter>
        <div className="navbar">
          <Link to="/">Main Page</Link>
          <Link to="/car/create">Create New Car</Link>
        </div>
        {loaded ? (
          <Switch>
            <Route exact path="/">
              <MainPage />
            </Route>
            <Route path="/car/:pk">

```


```

        <Details />
      </Route>
      <Route path="/car/create">
        <Details />
      </Route>
    </Switch>
  ) : (
    <Spinner />
  )
}
</BrowserRouter>
</AppContext.Provider>
);
};

```

Результат работы программы:

[Main Page](#)
[Create New Car](#)



asdasd
 Производитель УАЗ
 asdasdasdasdasd
 Оценка экспертов: 4564 из 10
[Delete](#)

Name

Price

Description

Score

localhost:3000/car/create

Main Page

Create New Car

Name

Price

Description

Score


Manufacturer

autovaz

Submit

- Main Page

Create New Car




zhiga

Производитель autovaz

sdfsdfsdfsdf

Оценка экспертов: 123 из 10

Delete



asdasd

Производитель УАЗ

asdasdasdasdasd

Оценка экспертов: 4564 из 10

Delete