	Министерство науки и высшего образования Российской Федерации **Федеральное государственное бюджетное образовательное учреждение высшего образования** **«Московский государственный технический университет имени Н.Э. Баумана** **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

Отчет по домашнему заданию

Студент Стрихар Павел Андреевич_____
фамилия, имя, отчество

Группа ИУ5-51Б

Студент 15.01.2022 **Стрихар П.А.**
подпись, дата фамилия, и.о.

Преподаватель 15.01.2022 **Гапанюк Ю.Е.**
подпись, дата фамилия, и.о.

2022г.

Задание:

Необходимо доработать ваше приложение из лабораторной работы №7 или №8. Необходимо решить следующие задачи:

Модифицировать базу данных:

- База данных должна содержать основную таблицу (например, "акции") и вторую таблицу (например, "компания акции").
- Основная таблица ссылается через внешний ключ на вторую таблицу.

Доработать веб-сервис:

- Добавить в веб-сервис новый метод получения списка значений из второй таблицы.
- Доработать фронтенд:
- Добавить на страницу с данными основной таблицы кнопки добавления, редактирования и удаления значений через API сервиса. Для добавления и редактирования использовать страницу отображения конкретного объекта.
- На всех страницах должны отображаться значения из таблиц вместо идентификаторов.

На странице отображения конкретного объекта необходимо использовать компонент combobox (выпадающий список). Необходимо использовать в этом компоненте новый метод веб-сервиса для второй таблицы.

Текст программы:

client/:

api/types/:

ManufacturerType.ts:

```
export type ManufacturerType = {
  pk: number;
  name: string;
  adress: string;
  email: string;
};
```

ProductType.ts:

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

export type ProductType = {
  pk: number;
  name: string;
  price: number;
  weight: number;
  imgSrc: string;
  quantity: number;
  description?: string;
  id_manufacturer: number;
  manufacturer?: ManufacturerType;
};
```

api/product/:

addProduct.ts:

```
import { ProductType } from 'api/types';

export const addProduct = async (values: Partial<ProductType>) => {
  const body = new FormData();
  for (const [key, value] of Object.entries(values)) {
    body.append(key, value.toString());
  }
  const data = await fetch('http://127.0.0.1:8000/product/', {
```

```
    method: 'POST',
    body: body,
  });
  if (data.ok) {
    return true;
  } else {
    return false;
  }
};
```

getProductById.ts:

```
import { ProductType } from 'api/types';

export const getProductById = async (id: number) => {
  const data = await fetch(`http://127.0.0.1:8000/product/${id}`, {
    headers: {
      'Content-Type': 'json/application',
    },
  });
  if (data.ok) {
    const parsedData: ProductType = await data.json();
    return parsedData;
  } else {
    return undefined;
  }
};
```

getProducts.ts:

```
import { ProductType } from 'api/types';

export const getProducts = async () => {
  const data = await fetch('http://127.0.0.1:8000/product', {
    headers: {
      'Content-Type': 'json/application',
    },
  });
  if (data.ok) {
    const parsedData: ProductType[] = await data.json();
    return parsedData;
  } else {
    return undefined;
  }
};
```

removeProductById.ts:

```
export const removeProductById = async (id: number) => {
  const data = await fetch(`http://127.0.0.1:8000/product/${id}`, {
    method: 'DELETE',
    headers: {
      'Content-Type': 'json/application',
    },
  });
  if (data.ok) {
    return true;
  } else {
    return false;
  }
};
```

updateProductById.ts:

```
import { ProductType } from 'api/types';

export const updateProductById = async (
  id: number,
  values: Partial<ProductType>,
) => {
  const body = new FormData();
  for (const [key, value] of Object.entries(values)) {
    if (value) {
      body.append(key, value.toString());
    }
  }
  const data = await fetch(`http://127.0.0.1:8000/product/${id}`, {
    method: 'PUT',
    body: body,
  });
  if (data.ok) {
    return true;
  } else {
    return false;
  }
};
```

api/manufacturer/:

addManufacturer.ts

```ts
import { ManufacturerType } from 'api/types';

export const addManufacturer = async (values: Partial<ManufacturerType>) => {
  const body = new FormData();
  for (const [key, value] of Object.entries(values)) {
    body.append(key, value.toString());
  }
  const data = await fetch('http://127.0.0.1:8000/manufacturer/', {
    method: 'POST',
    body: body,
  });
  if (data.ok) {
    return true;
  } else {
    return false;
  }
};
```

getManufacturer.ts:

```ts
import { ManufacturerType } from 'api/types';

export const getManufacturer = async () => {
  const data = await fetch('http://127.0.0.1:8000/manufacturer', {
    headers: {
      'Content-Type': 'json/application',
    },
  });
  if (data.ok) {
    const parsedData: ManufacturerType[] = await data.json();
    return parsedData;
  } else {
    return undefined;
  }
};
```

getManufacturerById.ts:

```ts
import { ManufacturerType } from 'api/types';

export const getManufacturerById = async (id: number) => {
  const data = await fetch(`http://127.0.0.1:8000/manufacturer/${id}`, {
```

```
    headers: {
      'Content-Type': 'json/application',
    },
  });
  if (data.ok) {
    const parsedData: ManufacturerType = await data.json();
    return parsedData;
  } else {
    return undefined;
  }
};
```

removeManufacturerById.ts:

```
export const removeManufacturerById = async (id: number) => {
  const data = await fetch(`http://127.0.0.1:8000/manufacturer/${id}`, {
    method: 'DELETE',
    headers: {
      'Content-Type': 'json/application',
    },
  });
  if (data.ok) {
    return true;
  } else {
    return false;
  }
};
```

updateManufacturerById.ts:

```
import { ManufacturerType } from 'api/types';

export const updateManufacturerById = async (
  id: number,
  values: Partial<ManufacturerType>,
) => {
  const body = new FormData();
  for (const [key, value] of Object.entries(values)) {
    body.append(key, value.toString());
  }
  const data = await fetch(`http://127.0.0.1:8000/manufacturer/${id}`, {
    method: 'PUT',
    body: body,
  });
  if (data.ok) {
    return true;
```

```
  } else {
    return false;
  }
};
```

components/:

Button/:

Button.style.tsx:

```
import { Colors } from 'constants/colors';

import styled from 'styled-components';

import { ButtonProps } from './Button.types';

export const Button = styled.button<ButtonProps>`
  background-color: ${({ backgroundColor }) =>
    backgroundColor || 'transparent'};

  padding: ${({ round }) => (round ? '5px 5px' : '10px 16px')};
  border: 1px solid ${({ backgroundColor }) => backgroundColor || 'transparent'};
  border-radius: 16px;
  cursor: pointer;

  color: ${Colors.TEXT_MAIN_COLOR};
  font-family: Roboto;
  font-size: 14px;
  font-weight: 400;

  &:hover {
    border-color: ${({ backgroundColor, backgroundHoverColor }) =>
      backgroundHoverColor
        ? backgroundHoverColor
        : backgroundColor || 'transparent'};
    background-color: ${({ backgroundColor, backgroundHoverColor }) =>
      backgroundHoverColor
        ? backgroundHoverColor
        : backgroundColor || 'transparent'};
  }
`;
```

Button.types.tsx:

```
export type ButtonProps = {
  backgroundColor?: string;
  backgroundHoverColor?: string;
```

```
  round?: boolean;
};
```

Card/:

Card.styles.tsx:

```
import { Colors } from 'constants/colors';

import styled from 'styled-components';

export const CardWrapper = styled.div`
  display: flex;
  flex: 1 0 26%;
  position: relative;
  height: 300px;
  border: 1px solid ${Colors.BORDER_MAIN};
  border-radius: 16px;
  margin: 26px;
  padding: 20px;
  flex-direction: column;
  justify-content: space-between;
  align-items: center;
  background-color: ${Colors.CARD_MAIN};
`;
```

Card.tsx:

```
import { Colors } from 'constants/colors';

import React, { useCallback } from 'react';
import { useNavigate } from 'react-router';
import { removeProductById } from 'api';
import { ReactComponent as IcClose } from 'assets/icons/close.svg';

import { Button } from 'components/Button';
import { Image } from 'components/Image';
import { StyledLink } from 'components/StyledLink/StyledLink.style';
import { Text } from 'components/Text';

import { CardWrapper } from './Card.styles';
import { CardProps } from './Card.types';
import { CardNavbar } from './CardNavbar';

export const Card = ({ data, onRemove, detailed }: CardProps): JSX.Element => {
  const handleRemoveCard = useCallback(async () => {
    await removeProductById(data.pk);
    if (onRemove) {
      onRemove();
    }
```

```
  }, []);

  const navigator = useNavigate();

  const handleManfClick = useCallback(() => {
    const params = new URLSearchParams();
    params.append('manf', data.id_manufacturer.toString());
    navigator({ search: params.toString() });
  }, []);

  return (
    <CardWrapper>
      <CardNavbar>
        <Button round onClick={handleRemoveCard}>
          <IcClose width={16} height={16} />
        </Button>
      </CardNavbar>
      <Image src={data.imgSrc}></Image>
      <Text bold>{data.name}</Text>
      <Text>Количество: {data.quantity} шт.</Text>
      <Text>Стоимость: {data.price} руб.</Text>
      <Text>Вес: {data.weight} гр.</Text>
      {detailed && (
      <Text>Описание: {data.description}</Text>
      )}
      <div style={{ display: 'flex', alignItems: 'center' }}>
        <Text style={{ paddingRight: '10px' }}>
          Показать: {data.manufacturer?.name}
        </Text>
        <Button
          backgroundColor={Colors.MAIN}
          backgroundHoverColor={Colors.MAIN_HOVERED}
          onClick={handleManfClick}
        >
          Перейти
        </Button>
      </div>
      {!detailed ? (
        <StyledLink to={`/product/${data.pk}`}>
          <Button
            backgroundColor={Colors.MAIN}
            backgroundHoverColor={Colors.MAIN_HOVERED}
          >
            Открыть
          </Button>
        </StyledLink>
      ) : (
        <StyledLink to={`/product/${data.pk}/edit`}>
          <Button
            backgroundColor={Colors.MAIN}
            backgroundHoverColor={Colors.MAIN_HOVERED}
```

```
          >
            Редактировать
          </Button>
        </StyledLink>
      )}
    </CardWrapper>
  );
};
```

Card.types.tsx:

```
import { ProductType } from 'api/types';

export type CardProps = {
  data: ProductType;
  onRemove?: () => void;
  detailed?: boolean;
};
```

CardNavbar/:

CardNavbar.styles.tsx:

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

export const CardNavbar = styled.div`
  position: absolute;
  top: 5px;
  right: 5px;
`;
```

fields/:

CustomField/:

CustomField.styles.tsx:

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

export const CustomInput = styled.input`
  background: transparent;
  border: 1px solid #c8c8c8;
  padding: 8px 16px;
  border-radius: 16px;
  color: white;
  height: 20px;

  &::placeholder {
    color: #bfbfbf;
  }
```

```
  &:hover {
    background: #43484b;
  }

  &:focus {
    background: #4e5457;
  }
`;

export const ErrorMessage = styled.p`
  font-size: 10px;
  color: red;
  margin: 0;
  margin-top: 4px;
`;
```

CustomField.tsx:

```
import React from 'react';
import { Field } from 'react-final-form';

import { CustomInput, ErrorMessage } from './CustomField.styles';
import { CustomInputFieldProps } from './CustomField.types';

export const CustomField = <T extends string | number>(
  props: CustomInputFieldProps<T>,
): JSX.Element => {
  const { required, ...inputProps } = props;
  return (
    <Field<T> {...inputProps}>
      {({ input: { value, onChange, onFocus, onBlur }, meta: { touched } }) => (
        <div style={{ display: 'flex', flexDirection: 'column' }}>
          <CustomInput
            value={value}
            onInput={onChange}
            onFocus={onFocus}
            onBlur={onBlur}
            {...inputProps}
          />
          {required && !value && touched && (
            <ErrorMessage>Обязательное поле</ErrorMessage>
          )}
        </div>
      )}
    </Field>
  );
};
```

CustomField.types.ts

```ts
import { FieldProps, FieldRenderProps } from 'react-final-form';

export type CustomInputFieldProps<T extends string | number> = FieldProps<
  T,
  FieldRenderProps<T, HTMLElement, T>,
  HTMLElement,
  T
> &
  Omit<
    FieldProps<T, FieldRenderProps<T, HTMLElement, T>, HTMLElement, T>,
    'children'
  > & {
    name: string;
    required?: boolean;
  };
```

CustomSelector/:

CustomSelector.tsx:

```tsx
import React, { useCallback, useEffect, useState } from 'react';
import { Field } from 'react-final-form';
import Select, {
  ActionMeta,
  MultiValue,
  SingleValue,
  StylesConfig,
} from 'react-select';

import { CustomInputFieldProps, Option } from './CustomSeletor.types';

export const CustomSelector = (
  props: CustomInputFieldProps<Option>,
): JSX.Element => {
  const { required, options, ...inputProps } = props;
  const Styles: StylesConfig<
    Option,
    boolean,
    Record<string, string> & { options: Option[] }
  > = {
    container: (provided) => ({
      ...provided,
      width: 201,
    }),
    singleValue: (provided) => ({
      ...provided,
      color: 'white',
```

```
  }),
  menuList: (provided) => ({
    ...provided,
    padding: 0,
  }),
  menu: (state) => ({
    width: 199,
    position: 'absolute',
    padding: 0,
    backgroundColor: 'transparent',
    borderRadius: 4,
    zIndex: 10,
    border: `1px solid #c8c8c8`,
  }),
  option: (provided, state) => ({
    ...provided,
    boxShadow: 'none',
    maxWidth: 193,
    fontWeight: 12,
    color: 'white',
    whiteSpace: 'nowrap',
    textOverflow: 'ellipsis',
    overflow: 'hidden',
    borderRadius: 4,
    cursor: 'pointer',
    border: `1px solid #c8c8c8`,
    backgroundColor: '#181a1b',

    ':hover': {
      backgroundColor: '#43484b',
    },
  }),
  dropdownIndicator: (provided, state) => ({
    transition: 'transform 1s ease',
    transform: state.selectProps.menuIsOpen ? 'rotate(180deg)' : 'none',
    width: 20,
    height: 20,
  }),
  control: (provided, state) => ({
    ...provided,
    width: 203,
    height: 33,
    padding: 0,
    backgroundColor: 'transparent',
    borderRadius: 16,
    boxShadow: 'none',
    cursor: state.selectProps.menuIsOpen ? 'text' : 'pointer',
    border: `1px solid #c8c8c8`,
    ':hover': {
      border: `1px solid #c8c8c8`,
    },
```

```
  }),
  input: (provided) => ({
    ...provided,
    width: 156,
    fontWeight: 12,
    color: 'white',
  }),
  placeholder: (provided) => ({
    ...provided,
    fontWeight: 12,
    color: '#bfbfbf',
  }),
};

const [currentValue, setValue] = useState<Option>();

useEffect(() => {
  setValue(inputProps.initialValue);
}, [inputProps.initialValue]);

return (
  <Field<Option> {...inputProps}>
    {({ input: { onChange, onFocus, onBlur } }) => {
      const handleChange = useCallback(
        (
          newValue: SingleValue<Option> | MultiValue<Option>,
          actionMeta: ActionMeta<Option>,
        ) => {
          onChange(newValue as Option);
          setValue(newValue as Option);
        },
        [],
      );
      return (
        <div style={{ display: 'flex', flexDirection: 'column' }}>
          <Select
            styles={Styles}
            onChange={handleChange}
            isSearchable
            isClearable
            onFocus={onFocus}
            onBlur={onBlur}
            maxMenuHeight={120}
            placeholder={props.placeholder}
            options={options}
            {...inputProps}
            value={currentValue}
          />
        </div>
      );
    }}
```

```
    </Field>
  );
};
```

CustomSelector.types.ts:

```
import { FieldProps, FieldRenderProps } from 'react-final-form';

export type Option = {
  value: number;
  label: string;
};

export type CustomInputFieldProps<T extends Option> = FieldProps<
  T,
  FieldRenderProps<T, HTMLElement, T>,
  HTMLElement,
  T
> &
  Omit<
    FieldProps<T, FieldRenderProps<T, HTMLElement, T>, HTMLElement, T>,
    'children'
  > & {
    required?: boolean;
    options?: Array<Option>;
  };
```

Image/:

Image.styles.tsx:

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

export const Image = styled.img`
  width: 140px;
  height: 140px;
  border-radius: 16px;
`;
```

Navbar/:

Navbar.styles.tsx:

```
import { Colors } from 'constants/colors';

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

```
export const NavbarWrapper = styled.div`
  display: flex;
  flex-direction: row;
  justify-content: left;
  align-items: center;
  width: 100%;
  border: 1px solid ${Colors.BORDER_MAIN};
  height: 50px;
  background-color: ${Colors.BODY_MAIN};
`;
```

Navbar.tsx:

```
import React, { ChangeEvent, useCallback } from 'react';
import { Field, Form } from 'react-final-form';
import { URLSearchParamsInit } from 'react-router-dom';

import { Button } from 'components/Button';
import { CustomInput } from 'components/fields/CustomField/CustomField.styles';
import { StyledLink } from 'components/StyledLink/StyledLink.style';

import { NavbarWrapper } from './Navbar.styles';

export type NavbarProps = {
  detailed?: boolean;
  onChange?: (
    nextInit: URLSearchParamsInit,
    navigateOptions?:
      | {
          replace?: boolean | undefined;
          state?: any;
        }
      | undefined,
  ) => void;
};

export const Navbar = ({ detailed, onChange }: NavbarProps) => {
  const onFilter = useCallback((event: ChangeEvent<HTMLInputElement>) => {
    onChange && onChange({ productName: event.target.value });
  }, []);

  return (
    <NavbarWrapper>
      <StyledLink to="/product">
        <Button>Главная</Button>
      </StyledLink>
      <StyledLink to="/manufacturer">
        <Button>Производители</Button>
      </StyledLink>
```

```
      <StyledLink to="/product/add">
        <Button>Добавить товар</Button>
      </StyledLink>
      <StyledLink to="/manufacturer/add">
        <Button>Добавить производителя</Button>
      </StyledLink>
      {detailed && (
        <div style={{ display: 'flex', justifyContent: 'center ' }}>
          <Form
            onSubmit={() => {
              //
            }}
            render={() => (
              <Field name="name">
                {({ input }) => (
                  <CustomInput
                    {...input}
                    placeholder="Поиск по сайту..."
                    onChange={(e) => {
                      input.onChange(e);
                      onFilter(e);
                    }}
                  />
                )}
              </Field>
            )}
          />
        </div>
      )}
    </NavbarWrapper>
  );
};
```

StyledForm/:

StyledForm.styles.tsx:

```
import { Colors } from 'constants/colors';

import styled from 'styled-components';

export const StyledForm = styled.form`
  border-radius: 16px;
  width: 500px;
  margin: auto;
  display: flex;
  align-items: center;
  flex-direction: column;
  padding: 20px;
  border: 1px solid ${Colors.BORDER_MAIN};
  background-color: ${Colors.BODY_MAIN};
  justify-content: space-between;
```

```
`;
```

StyledLink/:

StyledLink.styles.tsx:

```
import { Link } from 'react-router-dom';
import styled from 'styled-components';

export const StyledLink = styled(Link)`
  text-decoration: none;

  &:focus,
  &:hover,
  &:visited,
  &:link,
  &:active {
    text-decoration: none;
  }
`;
```

Text/:

Text.styles.tsx:

```
import { Colors } from 'constants/colors';

import styled from 'styled-components';

import { TextProps } from './Text.types';

export const Text = styled.p<TextProps>`
  color: ${Colors.TEXT_MAIN_COLOR};
  font-family: Roboto;
  font-size: 16px;
  font-weight: ${({ bold }) => (bold ? '800' : '600')};

  margin: 0;
`;
```

Text.types.tsx:

```
export type TextProps = {
  bold?: boolean;
};
```

constants/:

colors.ts:

```
export enum Colors {
  MAIN = '#CC6600',
  MAIN_HOVERED = '#FF9933',

  TEXT_MAIN_COLOR = 'white',

  BODY_MAIN = '#330000',
  CARD_MAIN = '#330000',
  BORDER_MAIN = 'black',

  ALERT = '#cc0a24',
  ALERT_HOVERED = '#ad0219',
}
```

pages/:

PageAddProduct/:

PageAddProduct.styles.tsx:

```
import { Colors } from 'constants/colors';

import styled from 'styled-components';

export const InputWrapper = styled.div`
  display: flex;
  width: 100%;
  color: ${Colors.TEXT_MAIN_COLOR};
  font-family: Roboto;
  font-size: 14px;
  font-weight: 400;
  justify-content: space-between;
  align-items: center;
  height: auto;
`;
```

PageAddProduct.tsx:

```
import React, { useCallback, useEffect, useState } from 'react';
import { Form } from 'react-final-form';
import { TailSpin } from 'react-loader-spinner';
import { getManufacturer } from 'api/manufacturer';
import { addProduct } from 'api/product';
```

```tsx
import { ProductType } from 'api/types';

import { Button } from 'components/Button';
import { CustomField } from 'components/fields/CustomField';
import { CustomSelector } from 'components/fields/CustomSelector';
import { Option } from 'components/fields/CustomSelector/CustomSeletor.types';
import { Navbar } from 'components/Navbar/Navbar';
import { StyledForm } from 'components/StyledForm/StyledForm.styles';
import { sleep } from 'utils/sleep';

import { InputWrapper } from './PageAddProduct.styles';

import 'react-loader-spinner/dist/loader/css/react-spinner-loader.css';

export const PageAddHotel = () => {
  const [loading, setLoading] = useState(false);
  const [manufacturers, setManufacturers] = useState<Option[] | undefined>(
    undefined,
  );

  useEffect(() => {
    setLoading(true);

    getManufacturer().then(async (values) => {
      if (values) {
        await sleep(500);
        const options = values.map((val) => ({
          value: val.pk,
          label: val.name,
        }));
        setManufacturers(options);
      }
      setLoading(false);
    });
  }, []);

  const onSubmit = useCallback(
    async (
      values: Record<string, string | Record<string, string | number>>,
    ) => {
      setLoading(true);

      if (!values['name']) {
        return { error: 'error' };
      }

      const manufacturer = values['manufacturer'] as Record<string, number>;
      const newValues: Partial<ProductType> = {
        ...values,
        id_manufacturer: manufacturer['value'],
      };
```

```
    return addProduct(newValues).then(async (value) => {
      if (value) {
        await sleep(2000);
      } else {
        console.error('Error while POST hotel fetch');
      }
      setLoading(false);
    });
  },
  [],
);

return (
  <>
    <Navbar />
    <Form
      onSubmit={onSubmit}
      validate={(values) => {
        const errors: Record<string, string> = {};
        if (!values.name) {
          errors.name = 'Required';
        }
        return errors;
      }}
      render={({ submitting, form, handleSubmit }) =>
        loading ? (
          <div style={{ position: 'absolute', top: '50%', left: '50%' }}>
            <TailSpin arialLabel="loading-indicator" />
          </div>
        ) : (
          <StyledForm
            onSubmit={async (event) => {
              await handleSubmit(event);
              form.reset();
            }}
          >
            <InputWrapper>
              <label htmlFor="name">Название</label>
              <CustomField
                name="name"
                type="text"
                component="input"
                placeholder="Название"
                required
              />
            </InputWrapper>
            <InputWrapper>
              <label htmlFor="price">Стоимость</label>
              <CustomField
                name="price"
```

```
            type="text"
            component="input"
            placeholder="Стоимость"
          />
        </InputWrapper>
        <InputWrapper>
          <label htmlFor="imgSrc">Ссылка на изображение</label>
          <CustomField
            name="imgSrc"
            type="text"
            component="input"
            placeholder="Ссылка"
          />
        </InputWrapper>
        <InputWrapper>
          <label htmlFor="manufacturer">Производитель</label>
          <CustomSelector
            name="manufacturer"
            placeholder="Выберите..."
            component="input"
            options={manufacturers}
          />
        </InputWrapper>
        <InputWrapper>
          <label htmlFor="weight">Вес</label>
          <CustomField
            name="weight"
            type="text"
            component="input"
            placeholder="Вес"
          />
        </InputWrapper>
        <InputWrapper>
          <label htmlFor="description">Описание</label>
          <CustomField
            name="description"
            type="text"
            component="textarea"
            placeholder="Описание..."
          />
        </InputWrapper>
        <InputWrapper>
          <label htmlFor="quantity">Количество</label>
          <CustomField
            name="quantity"
            type="text"
            component="input"
            placeholder="Количество"
          />
        </InputWrapper>
        <Button type="submit" disabled={submitting}>
```

```
              Добавить
            </Button>
          </StyledForm>
        )
      }
    />
  </>
);
};
```

PageAddManufacturer/:

PageAddManufacturer.styles.tsx:

```
import { Colors } from 'constants/colors';

import styled from 'styled-components';

export const InputWrapper = styled.div`
  display: flex;
  width: 100%;
  color: ${Colors.TEXT_MAIN_COLOR};
  font-family: Roboto;
  font-size: 14px;
  font-weight: 400;
  justify-content: space-between;
  align-items: center;
`;
```

PageAddManufacturer.tsx:

```
import React, { useCallback, useState } from 'react';
import { Form } from 'react-final-form';
import { TailSpin } from 'react-loader-spinner';
import { addManufacturer } from 'api';

import { Button } from 'components/Button';
import { CustomField } from 'components/fields/CustomField';
import { Navbar } from 'components/Navbar/Navbar';
import { StyledForm } from 'components/StyledForm/StyledForm.styles';
import { sleep } from 'utils/sleep';

import { InputWrapper } from './PageAddManufacturer.styles';

import 'react-loader-spinner/dist/loader/css/react-spinner-loader.css';

export const PageAddCountry = () => {
  const [loading, setLoading] = useState(false);
```

```
const onSubmit = useCallback(
  async (
    values: Record<string, string | Record<string, string | number>>,
  ) => {
    setLoading(true);

    return addManufacturer(values).then(async (value) => {
      if (value) {
        await sleep(2000);
      } else {
        console.error('Error while POST manufacturer fetch');
      }
      setLoading(false);
    });
  },
  [],
);

return (
  <>
    <Navbar />
    <Form
      onSubmit={onSubmit}
      validate={(values) => {
        const errors: Record<string, string> = {};
        if (!values.name) {
          errors.name = 'Required';
        }
        return errors;
      }}
      render={({ submitting, form, handleSubmit }) =>
        loading ? (
          <div style={{ position: 'absolute', top: '50%', left: '50%' }}>
            <TailSpin arialLabel="loading-indicator" />
          </div>
        ) : (
          <StyledForm
            onSubmit={async (event) => {
              await handleSubmit(event);
              form.reset();
            }}
          >
            <InputWrapper>
              <label htmlFor="name">Название</label>
              <CustomField
                name="name"
                type="text"
                component="input"
                placeholder="Название"
                required
```

```
              />
            </InputWrapper>
            <InputWrapper>
              <label htmlFor="email">Email</label>
              <CustomField
                name="email"
                type="text"
                component="input"
                placeholder="e-mail"
              />
            </InputWrapper>
            <InputWrapper>
              <label htmlFor="adress">Адрес</label>
              <CustomField
                name="adress"
                type="text"
                component="input"
                placeholder="Адрес"
              />
            </InputWrapper>
            <Button type="submit" disabled={submitting}>
              Добавить
            </Button>
          </StyledForm>
        )
      }
    />
  </>
);
};
```

PageManufacterers/:

PageManufacturers.styles.tsx:

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

export const PageMainWrapper = styled.div`
  display: flex;
  flex-wrap: wrap;
`;
```

PageManufacturers.tsx:

```
import React, { useCallback, useEffect, useState } from 'react';
import { getManufacturer, getProducts } from 'api';
import { ManufacturerType } from 'api/types';
```

```
import { CountryCard } from 'components/CountryCard';
import { Navbar } from 'components/Navbar/Navbar';

import { PageMainWrapper } from './PageManufacturers.styles';

export const PageCountries = (): JSX.Element => {
  const [manfs, setManf] = useState<ManufacturerType[]>();

  useEffect(() => {
    getManufacturer().then((data) => {
      getProducts().then((products) => {
        const parsedData = data?.filter(
          (elem) =>
            products?.filter((product) => product.id_manufacturer === elem.pk)
              .length,
        );
        setManf(parsedData);
      });
    });
  }, []);

  const onRemoveClick = useCallback((id: number) => {
    setManf((prev) => prev?.filter((value) => value.pk !== id));
  }, []);

  return (
    <PageMainWrapper>
      <Navbar />
      {manfs?.map((manf) => (
        <CountryCard
          key={manf.pk}
          data={manf}
          onRemove={() => onRemoveClick(manf.pk)}
        />
      ))}
    </PageMainWrapper>
  );
};
```

PageManufacturer/:

PageManufacturer.styles.tsx:

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

export const PageCoutnryWrapper = styled.div`
  display: flex;
  flex-wrap: wrap;
```

```
`;
```

PageManufacturer.tsx:

```
import React, { useEffect, useState } from 'react';
import { useParams } from 'react-router';
import { getManufacturerById } from 'api';
import { ManufacturerType } from 'api/types';

import { CountryCard } from 'components/CountryCard';
import { Navbar } from 'components/Navbar/Navbar';

import { PageCoutnryWrapper } from './PageManufacturer.styles';

export const PageCountry = (): JSX.Element => {
  const [man, setMan] = useState<ManufacturerType>();
  const { id } = useParams();

  useEffect(() => {
    getManufacturerById(parseInt(id || '0')).then((data) => {
      setMan(data);
    });
  }, [id]);

  if (man) {
    return (
      <PageCoutnryWrapper>
        <Navbar />
        <CountryCard data={man} detailed />
      </PageCoutnryWrapper>
    );
  }
  return <></>;
};
```

PageEdit/:

PageEdit.styles.tsx:

```
import { Colors } from 'constants/colors';

import styled from 'styled-components';

export const InputWrapper = styled.div`
  display: flex;
  width: 100%;
  color: ${Colors.TEXT_MAIN_COLOR};
```

```
  font-family: Roboto;
  font-size: 14px;
  font-weight: 400;
  justify-content: space-between;
  height: auto;
  align-items: center;
`;
```

PageEdit.tsx:

```tsx
import React, { useCallback, useEffect, useMemo, useState } from 'react';
import { Form } from 'react-final-form';
import { TailSpin } from 'react-loader-spinner';
import { useParams } from 'react-router';
import { useNavigate } from 'react-router-dom';
import { getManufacturer, getProductById, updateProductById } from 'api';
import { ProductType } from 'api/types';

import { Button } from 'components/Button';
import { CustomField } from 'components/fields/CustomField';
import { CustomSelector } from 'components/fields/CustomSelector';
import { Option } from 'components/fields/CustomSelector/CustomSeletor.types';
import { Navbar } from 'components/Navbar/Navbar';
import { StyledForm } from 'components/StyledForm/StyledForm.styles';
import { sleep } from 'utils/sleep';

import { InputWrapper } from './PageEdit.styles';

import 'react-loader-spinner/dist/loader/css/react-spinner-loader.css';

export const PageEdit = () => {
  const [loading, setLoading] = useState(false);
  const [manfs, setManfs] = useState<Option[] | undefined>(undefined);
  const [initialValue, setInitialValue] = useState<
    | Record<string, string | number | Record<string, number | string>>
    | undefined
  >(undefined);

  const initialManf = useMemo(
    () => initialValue?.manufacturer as Option,
    [initialValue],
  );

  const { id } = useParams();
  const navigate = useNavigate();

  useEffect(() => {
    setLoading(true);
```

```
  getManufacturer().then(async (values) => {
    if (values) {
      await sleep(200);
      const options = values.map((val) => ({
        value: val.pk,
        label: val.name,
      }));
      setManfs(options);

      getProductById(parseInt(id || '0')).then(async (data) => {
        if (data) {
          await sleep(200);
          const formData = data as Record<
            string,
            string | number | Record<string, number | string>
          >;
          formData['manufacturer'] = {
            value: data.manufacturer?.pk || -1,
            label: data.manufacturer?.name || '',
          };
          setInitialValue(formData);
        }
      });
    }
    setLoading(false);
  });
}, [id]);

const onSubmit = useCallback(
  async (
    values: Record<string, string | Record<string, string | number>>,
  ) => {
    setLoading(true);

    if (!values['name']) {
      return { error: 'error' };
    }

    const { manufacturer, ...valuesWithoutOption } = values;
    const manufacturerId = manufacturer as Record<string, string | number>;
    const newValues: Partial<ProductType> = {
      ...valuesWithoutOption,
      id_manufacturer: manufacturerId['value'] as number,
    };
    console.log(JSON.stringify(newValues));

    return updateProductById(parseInt(id || '0'), newValues).then(
      async (value) => {
        if (value) {
          await sleep(2000);
        } else {
```

```
          console.error('Error while PUT hotel fetch');
        }
        setLoading(false);
        navigate('/');
      },
    );
  },
  [],
);

return (
  <>
    <Navbar />
    <Form
      onSubmit={onSubmit}
      initialValues={initialValue}
      validate={(values) => {
        const errors: Record<string, string> = {};
        if (!values.name) {
          errors.name = 'Required';
        }
        return errors;
      }}
      render={({ submitting, form, handleSubmit }) =>
        loading ? (
          <div style={{ position: 'absolute', top: '50%', left: '50%' }}>
            <TailSpin arialLabel="loading-indicator" />
          </div>
        ) : (
          <StyledForm
            onSubmit={async (event) => {
              await handleSubmit(event);
              form.reset();
            }}
          >
            <InputWrapper>
              <label htmlFor="name">Название</label>
              <CustomField
                name="name"
                type="text"
                component="input"
                placeholder="Название"
                required
              />
            </InputWrapper>
            <InputWrapper>
              <label htmlFor="price">Стоимость</label>
              <CustomField
                name="price"
                type="text"
                component="input"
```

```
            placeholder="Стоимость"
          />
        </InputWrapper>
        <InputWrapper>
          <label htmlFor="imgSrc">Ссылка на изображение</label>
          <CustomField
            name="imgSrc"
            type="text"
            component="input"
            placeholder="Ссылка"
          />
        </InputWrapper>
        <InputWrapper>
          <label htmlFor="manufacturer">Производитель</label>
          <CustomSelector
            name="manufacturer"
            placeholder="Выберите..."
            component="input"
            options={manfs}
            initialValue={initialManf}
          />
        </InputWrapper>
        <InputWrapper>
          <label htmlFor="weight">Вес</label>
          <CustomField
            name="weight"
            type="text"
            component="input"
            placeholder="Вес"
          />
        </InputWrapper>
        <InputWrapper>
          <label htmlFor="description">Описание</label>
          <CustomField
            name="description"
            type="text"
            component="textarea"
            placeholder="Описание..."
          />
        </InputWrapper>
        <InputWrapper>
          <label htmlFor="quantity">Количество</label>
          <CustomField
            name="quantity"
            type="text"
            component="input"
            placeholder="Количество"
          />
        </InputWrapper>
        <Button type="submit" disabled={submitting}>
          Обновить
```

```
              </Button>
            </StyledForm>
          )
        }
      />
    </>
  );
};
```

PageEditManufacturer/:

PageEditManufacturer.styles.tsx:

```
import { Colors } from 'constants/colors';

import styled from 'styled-components';

export const InputWrapper = styled.div`
  display: flex;
  width: 100%;
  color: ${Colors.TEXT_MAIN_COLOR};
  font-family: Roboto;
  font-size: 14px;
  font-weight: 400;
  justify-content: space-between;
  align-items: center;
`;
```

PageEditManufacturer.tsx:

```
import React, { useCallback, useEffect, useState } from 'react';
import { Form } from 'react-final-form';
import { TailSpin } from 'react-loader-spinner';
import { useParams } from 'react-router';
import { useNavigate } from 'react-router-dom';
import { getManufacturerById, updateManufacturerById } from 'api';

import { Button } from 'components/Button';
import { CustomField } from 'components/fields/CustomField';
import { Navbar } from 'components/Navbar/Navbar';
import { StyledForm } from 'components/StyledForm/StyledForm.styles';
import { sleep } from 'utils/sleep';

import { InputWrapper } from './PageEditManufacturer.styles';

import 'react-loader-spinner/dist/loader/css/react-spinner-loader.css';

export const PageEditCountry = () => {
```

```tsx
const [loading, setLoading] = useState(false);
const [initialValue, setInitialValue] = useState<
  | Record<string, string | number | Record<string, number | string>>
  | undefined
>(undefined);

const { id } = useParams();
const navigate = useNavigate();

useEffect(() => {
  setLoading(true);

  getManufacturerById(parseInt(id || '0')).then(async (data) => {
    if (data) {
      await sleep(200);
      setInitialValue(data);
    }
  });
  setLoading(false);
}, [id]);

const onSubmit = useCallback(
  async (
    values: Record<string, string | Record<string, string | number>>,
  ) => {
    setLoading(true);

    return updateManufacturerById(parseInt(id || '0'), values).then(
      async (value) => {
        if (value) {
          await sleep(2000);
        } else {
          console.error('Error while PUT man fetch');
        }
        setLoading(false);
        navigate('/');
      },
    );
  },
  [],
);

return (
  <>
    <Navbar />
    <Form
      onSubmit={onSubmit}
      initialValues={initialValue}
      validate={(values) => {
        const errors: Record<string, string> = {};
        if (!values.name) {
```

```
          errors.name = 'Required';
        }
        return errors;
      }}
      render={({ submitting, form, handleSubmit }) =>
        loading ? (
          <div style={{ position: 'absolute', top: '50%', left: '50%' }}>
            <TailSpin arialLabel="loading-indicator" />
          </div>
        ) : (
          <StyledForm
            onSubmit={async (event) => {
              await handleSubmit(event);
              form.reset();
            }}
          >
            <InputWrapper>
              <label htmlFor="name">Название</label>
              <CustomField
                name="name"
                type="text"
                component="input"
                placeholder="Название"
                required
              />
            </InputWrapper>
            <InputWrapper>
              <label htmlFor="email">Email</label>
              <CustomField
                name="email"
                type="text"
                component="input"
                placeholder="e-mail"
              />
            </InputWrapper>
            <InputWrapper>
              <label htmlFor="adress">Адрес</label>
              <CustomField
                name="adress"
                type="text"
                component="input"
                placeholder="Адрес"
              />
            </InputWrapper>
            <Button type="submit" disabled={submitting}>
              Обновить
            </Button>
          </StyledForm>
        )
      }
    />
```

```
    </>
  );
};
```

PageProduct/:

PageProduct.tsx:

```
import React, { useEffect, useState } from 'react';
import { useParams } from 'react-router';
import { getProductById } from 'api';
import { ProductType } from 'api/types';

import { Card } from 'components/Card/Card';
import { Navbar } from 'components/Navbar/Navbar';

export const PageProduct = (): JSX.Element => {
  const [product, setProduct] = useState<ProductType>();
  const { id } = useParams();

  useEffect(() => {
    getProductById(parseInt(id || '0')).then((data) => {
      setProduct(data);
    });
  }, [id]);

  if (product) {
    return (
      <>
        <Navbar />
        <Card data={product} detailed />
      </>
    );
  }
  return <></>;
};
```

PageMain/:

PageMain.styles.tsx:

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

export const PageMainWrapper = styled.div`
  display: flex;
  flex-wrap: wrap;
```

```
`;
```

PageMain.tsx:

```tsx
import React, { useCallback, useEffect, useState } from 'react';
import { useSearchParams } from 'react-router-dom';
import { getProducts } from 'api';
import { ProductType } from 'api/types';

import { Card } from 'components/Card/Card';
import { Navbar } from 'components/Navbar/Navbar';

import { PageMainWrapper } from './PageMain.styles';

export const PageMain = (): JSX.Element => {
  const [products, setProducts] = useState<ProductType[]>();
  const [params, setParams] = useSearchParams();

  useEffect(() => {
    getProducts().then((data) => {
      if (params.get('manf')) {
        const parsedData = data?.filter((elem) => {
          if (params.get('manf')) {
            return (
              elem.id_manufacturer === parseInt(params.get('manf') || '-1')
            );
          } else {
            return true;
          }
        });
        setProducts(parsedData);
      } else {
        if (params.get('productName')) {
          const parsedData = data?.filter((elem) => {
            if (params.get('productName')) {
              return elem.name.includes(params.get('productName') || '');
            } else {
              return true;
            }
          });
          setProducts(parsedData);
        } else {
          setProducts(data);
        }
      }
    });
  }, [params]);

  const onRemoveClick = useCallback((id: number) => {
    setProducts((prev) => prev?.filter((value) => value.pk !== id));
```

```
  }, []);

  return (
    <PageMainWrapper>
      <Navbar detailed onChange={setParams} />
      {products?.map((product) => (
        <Card
          key={product.pk}
          data={product}
          onRemove={() => onRemoveClick(product.pk)}
        />
      ))}
    </PageMainWrapper>
  );
};
```

utils/:

sleep.ts

```
export const sleep = async (time?: number) =>
  await new Promise((r) => setTimeout(r, time || 1000));
```

App.tsx

```
import React from 'react';
import { BrowserRouter, Navigate, Route, Routes } from 'react-router-dom';
import { PageAddCountry } from 'pages/PageAddManufacturer';
import { PageAddHotel } from 'pages/PageAddProduct';
import { PageCountries } from 'pages/PageManufacturers';
import { PageCountry } from 'pages/PageManufacturer';
import { PageEdit } from 'pages/PageEdit';
import { PageEditCountry } from 'pages/PageEditManufacturer';
import { PageMain } from 'pages/PageMain';
import { PageProduct } from 'pages/PageProduct';

import './App.css';

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Navigate to="/product" />} />

        <Route path="/product" element={<PageMain />} />
        <Route path="/product/add" element={<PageAddHotel />} />
        <Route path="/product/:id/edit" element={<PageEdit />} />
        <Route path="/product/:id" element={<PageProduct />} />
```

```
        <Route path="/manufacturer" element={<PageCountries />} />
        <Route path="/manufacturer/add" element={<PageAddCountry />} />
        <Route path="/manufacturer/:id/edit" element={<PageEditCountry />} />
        <Route path="/manufacturer/:id" element={<PageCountry />} />
      </Routes>
    </BrowserRouter>
  );
}

export default App;
```

index.tsx

```
import React from 'react';
import ReactDOM from 'react-dom';

import App from './App';

import './index.css';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root'),
);
```

server/

productModel/

models.py

```
from django.db import models

class Manufacturer(models.Model):
    name = models.CharField(max_length=100)
    adress = models.CharField(max_length=999, blank=True, null=True)
    email = models.CharField(max_length=100, blank=True, null=True)

    class Meta:
        managed = True
        db_table = 'manufacturer'


class Product(models.Model):
    name = models.CharField(max_length=255, blank=True, null=True)
    price = models.IntegerField(blank=True, null=True)
    quantity = models.IntegerField(blank=True, null=True)
```

```
    id_manufacturer = models.ForeignKey(Manufacturer, on_delete=models.CASCADE)
    weight = models.FloatField(blank=True, null=True)
    description = models.CharField(max_length=255, blank=True, null=True)
    imgSrc = models.CharField(max_length=1000, blank=True, null=True)

    class Meta:
        managed = True
        db_table = 'product'
```

serializers.py

```
from productModel.models import Manufacturer, Product
from rest_framework import serializers

class ManufacturerSerializer(serializers.ModelSerializer):
    class Meta:
        model = Manufacturer
        fields = ("pk", "name", "adress", "email")

class ProductSerializer(serializers.ModelSerializer):
    def create(self, validated_data):
        return Product.objects.create(**validated_data)
    manufacturer = ManufacturerSerializer(source='id_manufacturer', required=False)
    class Meta:
        model = Product
        fields = ["pk", "name", "price", "quantity", "manufacturer",
"id_manufacturer", "weight", "description", "imgSrc"]
```

views.py

```
from productModel.models import Manufacturer, Product
from productModel.serializers import ManufacturerSerializer, ProductSerializer
from rest_framework.generics import ListCreateAPIView, ListAPIView,
RetrieveUpdateDestroyAPIView
from rest_framework.response import Response

class ManufacturerView(ListCreateAPIView):
    queryset = Manufacturer.objects.all()
    serializer_class = ManufacturerSerializer

class SingleManufacturerView(RetrieveUpdateDestroyAPIView):
    queryset = Manufacturer.objects.all()
    serializer_class = ManufacturerSerializer

class ProductView(ListAPIView):
    queryset = Product.objects.all()
    serializer_class = ProductSerializer
```

```
    def post(self, request):
        manufacturer = request.data
        serializer = ProductSerializer(data=manufacturer)
        if serializer.is_valid(raise_exception=True):
            article_saved = serializer.save()
        return Response({"success": "Product '{}' created
successfully".format(article_saved.name)})

class SingleProductView(RetrieveUpdateDestroyAPIView):
    queryset = Product.objects.all()
    serializer_class = ProductSerializer
```

products/

urls.py

```
from django.contrib import admin
from django.urls import include, path
from productModel import views as productModel

urlpatterns = [
    path('manufacturer/', productModel.ManufacturerView.as_view()),
    path('manufacturer/<int:pk>', productModel.SingleManufacturerView.as_view()),

    path('product/', productModel.ProductView.as_view()),
    path('product/<int:pk>', productModel.SingleProductView.as_view()),

    path('api-auth/', include('rest_framework.urls', namespace='rest_framework')),
    path('admin/', admin.site.urls)
]
```

settings.py

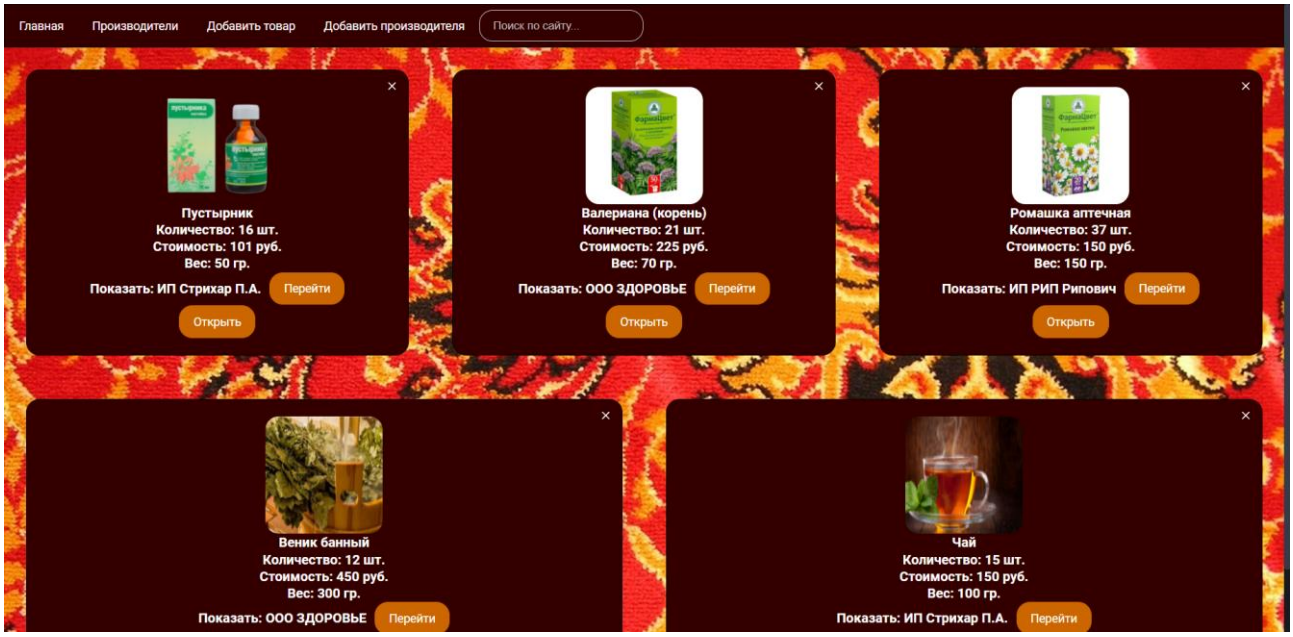
```
"""
Django settings for hotels project.

Generated by 'django-admin startproject' using Django 3.1.1.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.1/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
```

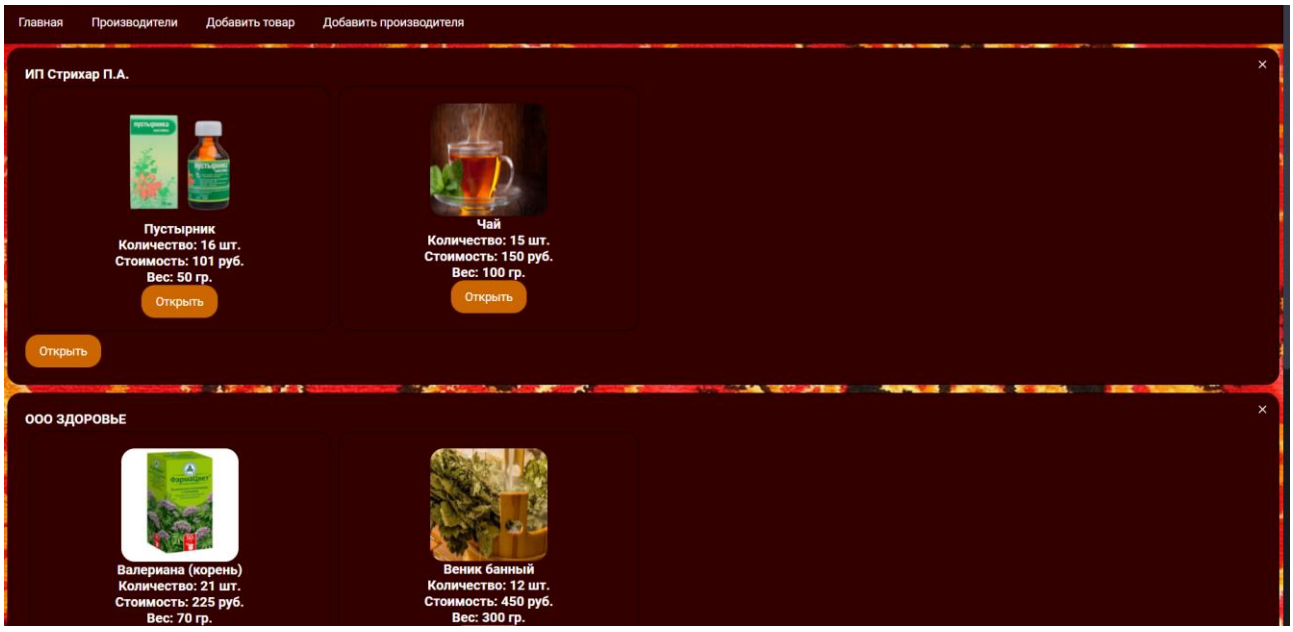
```
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'g5=o&_*8a!sv%x-ac_%@4kgrb75w56_wzrw4+&d4d+ew4i+x+9'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'corsheaders',
    'rest_framework',

    'productModel'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'products.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
```

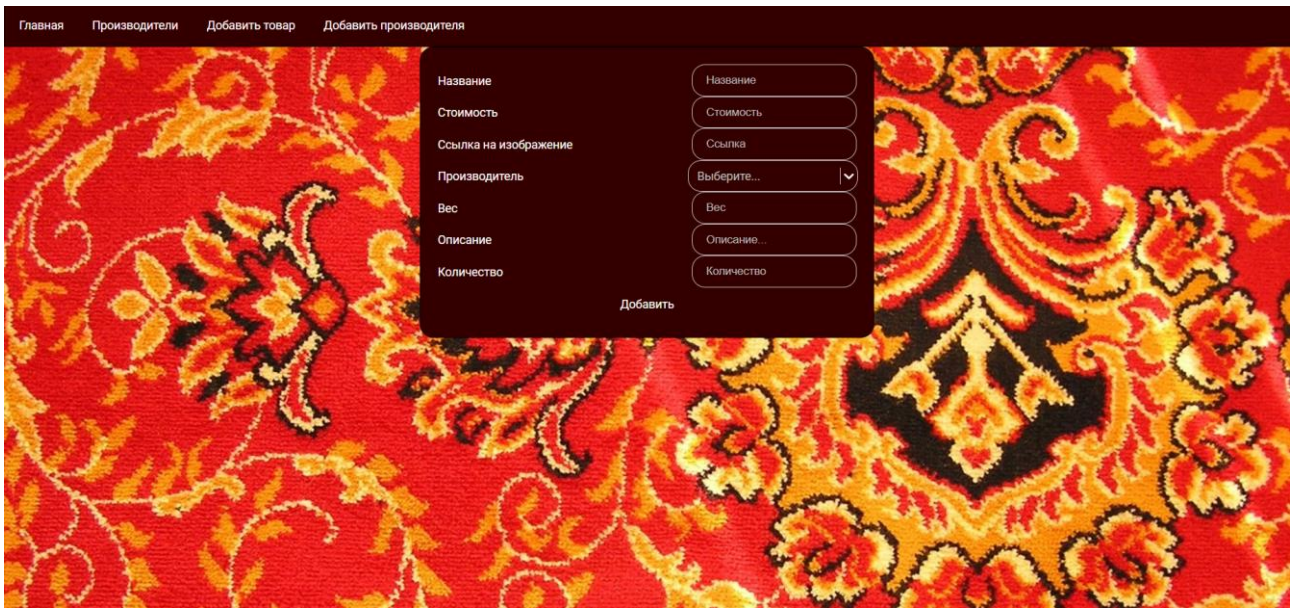
```
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'products.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'products',
        'USER': 'django',
        'PASSWORD': 'root',
        'HOST': 'db',
        'PORT': 3306,
        'OPTIONS': {
          'init_command': 'SET NAMES utf8mb4',
          'charset': 'utf8mb4'
        },
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.1/topics/i18n/
```

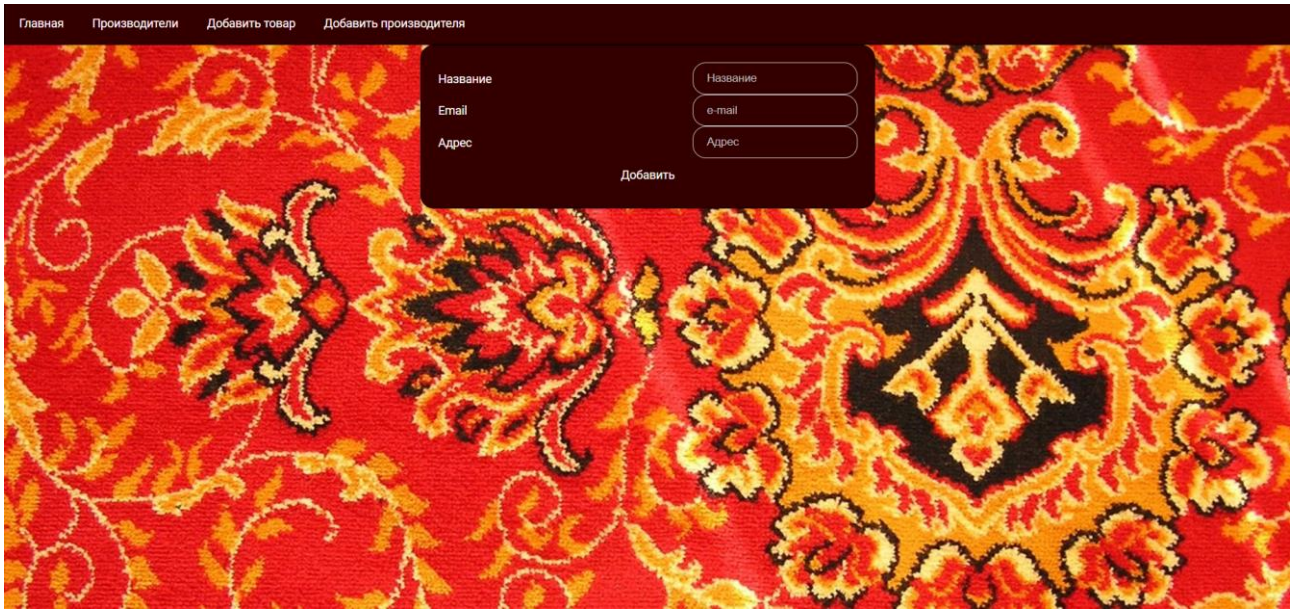
```
LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Europe/Moscow'

USE_I18N = True

USE_L10N = True

USE_TZ = True

UNICODE_JSON = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/

STATIC_URL = '/static/'

CORS_ALLOW_ALL_ORIGINS = True # If this is used then `CORS_ALLOWED_ORIGINS` will
not have any effect
CORS_ALLOW_CREDENTIALS = True
CORS_ALLOWED_ORIGINS = [
    'http://localhost:3000',
] # If this is used, then not need to use `CORS_ALLOW_ALL_ORIGINS = True`
CORS_ALLOWED_ORIGIN_REGEXES = [
    'http://localhost:3000',
]
```

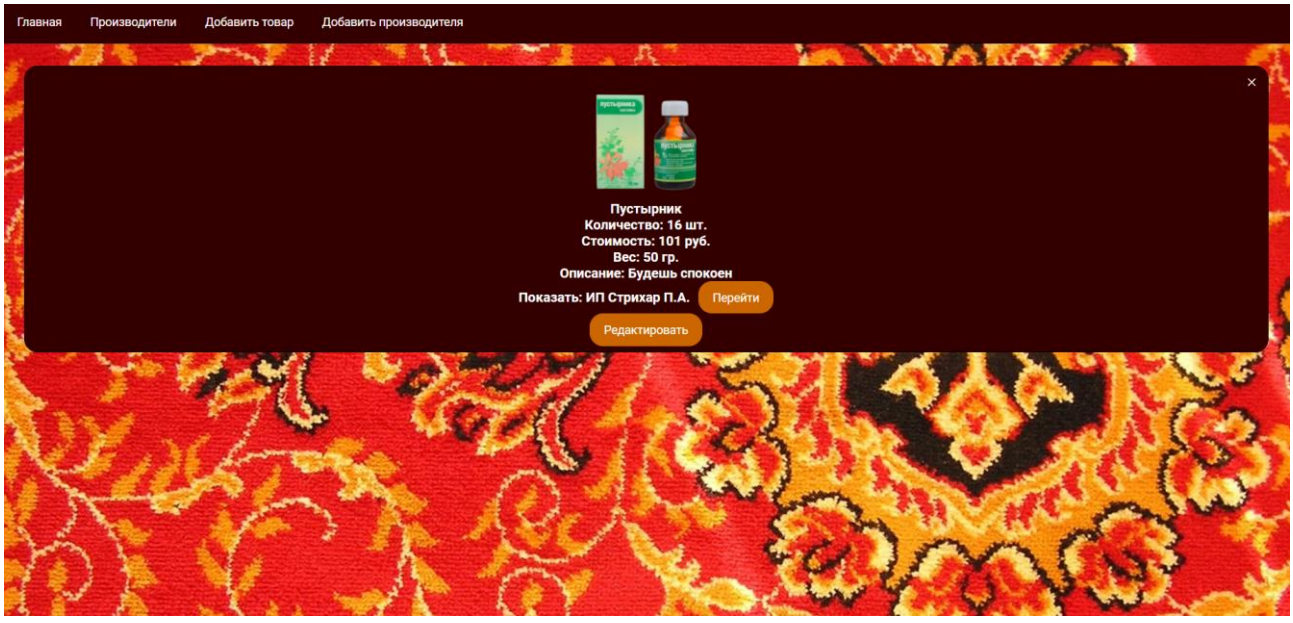
Экранные формы с примерами выполнения программы:






Главная
Производители
Добавить товар
Добавить производителя
Название
Название
Стоимость
Стоимость
Ссылка на изображение
Ссылка
Производитель
Выберите...
Вес
Вес
Описание
Описание...
Количество
Количество
Добавить


Главная
Производители
Добавить товар
Добавить производителя
Название
Название
Email
e-mail
Адрес
Адрес
Добавить


Главная
Производители
Добавить товар
Добавить производителя
Пустырник
Количество: 16 шт.
Стоимость: 101 руб.
Вес: 50 гр.
Описание: Будешь спокоен
Показать: ИП Стрихар П.А.
Перейти
Редактировать