	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 Бондаренко Иван Геннадьевич_____
фамилия, имя, отчество

Группа ИУ5-51Б

Студент 10.01.2021 **Бондаренко И.Г.**
подпись, дата фамилия, и.о.

Преподаватель 15.01.2021 **Гапанюк Ю.Е.**
подпись, дата фамилия, и.о.

2021г.

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/:

HotelDataType.ts:

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

export type HotelDataType = {
  pk: number;
  name: string;
  url: string;
  imageSrc: string;
  stars: number;
  country_code: number;
  country?: CountryDataType;
};
```

CountryDataType.ts:

```
export type CountryDataType = {
  pk: number;
  name: string;
};
```

api/hotels/:

addHotel.ts:

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

export const addHotel = async (values: Partial<HotelDataType>) => {
  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/hotel/', {
    method: 'POST',
    body: body,
  });
  if (data.ok) {
    return true;
```

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

getHotelById.ts:

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

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

getHotels.ts:

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

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

removeHotelById.ts:

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

```
    method: 'DELETE',
    headers: {
      'Content-Type': 'json/application',
    },
  });
  if (data.ok) {
    return true;
  } else {
    return false;
  }
};
```

updateHotelById.ts:

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

api/country/:
addCountry.ts

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

export const addCountry = async (values: Partial<CountryDataType>) => {
  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/country/', {
    method: 'POST',
    body: body,
  });
  if (data.ok) {
    return true;
  } else {
    return false;
  }
```

```
};
```

getCountries.ts:

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

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

getCountryById.ts:

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

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

removeCountryById.ts:

```
export const removeCountryById = async (id: number) => {
  const data = await fetch(`http://127.0.0.1:8000/country/${id}`, {
    method: 'DELETE',
    headers: {
      'Content-Type': 'json/application',
    },
```

```
  });
  if (data.ok) {
    return true;
  } else {
    return false;
  }
};
```

updateCountryById.ts:

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

export const updateCountryById = async (
  id: number,
  values: Partial<CountryDataType>,
) => {
  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/country/${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 21%;
  position: relative;
  height: 300px;
  border: 1px solid ${Colors.BORDER_MAIN};
  border-radius: 16px;
  margin: 5px;
  padding: 20px;
  flex-direction: column;
  justify-content: space-between;
  align-items: center;
  background-color: ${Colors.CARD_MAIN};
`;
```

Card.tsx:

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

import React, { useCallback } from 'react';
import { useNavigate } from 'react-router';
import { removeHotelById } 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 removeHotelById(data.pk);
    if (onRemove) {
      onRemove();
    }
  }, []);

  const navigator = useNavigate();

  const handleCountryClick = useCallback(() => {
    const params = new URLSearchParams();
    params.append('country', data.country_code.toString());
    navigator({ search: params.toString() });
  }, []);

  return (
    <CardWrapper>
      <CardNavbar>
        <Button round onClick={handleRemoveCard}>
          <IcClose width={16} height={16} />
        </Button>
      </CardNavbar>
      <a href={data.url}>
        <Image src={data.imageSrc}></Image>
      </a>
      <Text bold>{data.name}</Text>
      <Text>Количество звезд: {data.stars}</Text>
      <div style={{ display: 'flex', alignItems: 'center' }}>
        <Text style={{ paddingRight: '10px' }}>
          Страна: {data.country?.name}
        </Text>
        <Button
          backgroundColor={Colors.MAIN}
```

```
          backgroundHoverColor={Colors.MAIN_HOVERED}
          onClick={handleCountryClick}
        >
          Перейти
        </Button>
      </div>
      {!detailed ? (
        <StyledLink to={`/hotel/${data.pk}`}>
          <Button
            backgroundColor={Colors.MAIN}
            backgroundHoverColor={Colors.MAIN_HOVERED}
          >
            Открыть
          </Button>
        </StyledLink>
      ) : (
        <StyledLink to={`/hotel/${data.pk}/edit`}>
          <Button
            backgroundColor={Colors.MAIN}
            backgroundHoverColor={Colors.MAIN_HOVERED}
          >
            Редактировать
          </Button>
        </StyledLink>
      )}
    </CardWrapper>
  );
};
```

Card.types.tsx:

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

export type CardProps = {
  data: HotelDataType;
  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

```
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:

```
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';
```

```tsx
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: space-evenly;
  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 { getHotels } from 'api';
import { HotelDataType } from 'api/types';

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({ hotelName: event.target.value });
  }, []);

  return (
    <NavbarWrapper>
      <StyledLink to="/hotel">
        <Button>Home</Button>
      </StyledLink>
      <StyledLink to="/country">
        <Button>Countries</Button>
      </StyledLink>
      <StyledLink to="/hotel/add">
        <Button>Add hotel</Button>
      </StyledLink>
      <StyledLink to="/country/add">
        <Button>Add country</Button>
      </StyledLink>
      {detailed && (
        <div style={{ display: 'flex', justifyContent: 'center ' }}>
          <Form
            onSubmit={() => {
              //
            }}
            render={() => (
              <Field name="name">
                {({ input }) => (
                  <CustomInput
                    {...input}
                    placeholder="Найти"
                    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: 50%;
  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;
  height: 250px;
`;
```

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: 14px;
  font-weight: ${({ bold }) => (bold ? '700' : '400')};

  margin: 0;
`;
```

Text.types.tsx:

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

constants/:

colors.ts:

```
export enum Colors {
  MAIN = '#24a0ed',
  MAIN_HOVERED = '#1597DD',

  TEXT_MAIN_COLOR = 'white',

  BODY_MAIN = '#181a1b',
  CARD_MAIN = '#181a1b',
  BORDER_MAIN = '#C8C8C8',

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

pages/:

PageAddHotel/:

PageAddHotel.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;
`;
```

PageAddHotel.tsx:

```tsx
import React, { useCallback, useEffect, useState } from 'react';
import { Form } from 'react-final-form';
import { TailSpin } from 'react-loader-spinner';
import { getCountries } from 'api/country';
import { addHotel } from 'api/hotel';
import { HotelDataType } 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 './PageAddHotel.styles';

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

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

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

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

  const onSubmit = useCallback(
    async (
      values: Record<string, string | Record<string, string | number>>,
```

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

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

    const stars = values['stars'] as Record<string, number>;
    const country = values['country'] as Record<string, number>;
    const newValues: Partial<HotelDataType> = {
      ...values,
      stars: stars['value'],
      country_code: country['value'],
    };

    return addHotel(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="url">Ссылка на отель</label>
              <CustomField
                name="url"
                type="text"
                component="input"
                placeholder="Ссылка"
              />
            </InputWrapper>
            <InputWrapper>
              <label htmlFor="imageSrc">Ссылка на изображение отеля</label>
              <CustomField
                name="imageSrc"
                type="text"
                component="input"
                placeholder="Ссылка"
              />
            </InputWrapper>
            <InputWrapper>
              <label htmlFor="stars">Количество звезд</label>
              <CustomSelector
                name="stars"
                placeholder="Выберите..."
                options={[
                  { value: 1, label: '☆' },
                  { value: 2, label: '☆☆' },
                  { value: 3, label: '☆☆☆' },
                  { value: 4, label: '☆☆☆☆' },
                  { value: 5, label: '☆☆☆☆☆' },
                ]}
              />
            </InputWrapper>
            <InputWrapper>
              <label htmlFor="country">Страна</label>
              <CustomSelector
                name="country"
                placeholder="Выберите..."
                component="input"
                options={countries}
              />
            </InputWrapper>
```

```
              <Button type="submit" disabled={submitting}>
                Добавить
              </Button>
            </StyledForm>
          )
        }
      />
    </>
  );
};
```

PageAddCountry/:

PageAddCountry.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;
`;
```

PageAddCountry.tsx:

```
import React, { useCallback, useState } from 'react';
import { Form } from 'react-final-form';
import { TailSpin } from 'react-loader-spinner';
import { addCountry } 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 './PageAddCountry.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 addCountry(values).then(async (value) => {
      if (value) {
        await sleep(2000);
      } else {
        console.error('Error while POST country 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>
            <Button type="submit" disabled={submitting}>
              Добавить
            </Button>
          </StyledForm>
        )
      }
    />
  </>
);
};
```

PageCountries/:

PageCountries.styles.tsx:

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

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

PageCountries.tsx:

```
import React, { useCallback, useEffect, useState } from 'react';
import { getCountries, getHotels } from 'api';
import { CountryDataType } from 'api/types';

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

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

export const PageCountries = (): JSX.Element => {
  const [countries, setCountries] = useState<CountryDataType[]>();

  useEffect(() => {
    getCountries().then((data) => {
      getHotels().then((hotels) => {
        const parsedData = data?.filter(
          (elem) =>
            hotels?.filter((hotel) => hotel.country_code === elem.pk).length,
        );
        setCountries(parsedData);
      });
    });
  }, []);
```

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

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

PageCountry/:

PageCountry.styles.tsx:

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

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

PageCountry.tsx:

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

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

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

export const PageCountry = (): JSX.Element => {
  const [country, setCountry] = useState<CountryDataType>();
  const { id } = useParams();

  useEffect(() => {
    getCountryById(parseInt(id || '0')).then((data) => {
      setCountry(data);
```

```
    });
  }, [id]);

  if (country) {
    return (
      <PageCoutnryWrapper>
        <Navbar />
        <CountryCard data={country} 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;
  align-items: center;
`;
```

PageEdit.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 { getHotelById, updateHotelById } from 'api';
import { getCountries } from 'api/country';
import { HotelDataType } 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';
```

```tsx
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 [countries, setCountries] = useState<Option[] | undefined>(undefined);
  const [initialValue, setInitialValue] = useState<
    | Record<string, string | number | Record<string, number | string>>
    | undefined
  >(undefined);

  const initialCountry = useMemo(
    () => initialValue?.country as Option,
    [initialValue],
  );

  const initialStars = useMemo(
    () => initialValue?.stars as Option,
    [initialValue],
  );

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

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

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

        getHotelById(parseInt(id || '0')).then(async (data) => {
          if (data) {
            await sleep(200);
            const formData = data as Record<
              string,
              string | number | Record<string, number | string>
            >;
            formData['stars'] = {
              value: data?.stars,
              label: '☆'.repeat(data?.stars),
            } as Option;
```

```
          formData['country'] = {
            value: data.country?.pk || -1,
            label: data.country?.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 stars = values['stars'] as Record<string, number>;
    const country = values['country'] as Record<string, number>;
    const newValues: Partial<HotelDataType> = {
      ...values,
      stars: stars['value'],
      country_code: country['value'],
    };

    return updateHotelById(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="url">Ссылка на отель</label>
                <CustomField
                  name="url"
                  type="text"
                  component="input"
                  placeholder="Ссылка"
                />
              </InputWrapper>
              <InputWrapper>
                <label htmlFor="imageSrc">Ссылка на изображение отеля</label>
                <CustomField
                  name="imageSrc"
                  type="text"
                  component="input"
                  placeholder="Ссылка"
                />
              </InputWrapper>
              <InputWrapper>
                <label htmlFor="stars">Количество звезд</label>
                <CustomSelector
                  name="stars"
```

```
                placeholder="Выберите..."
                initialValue={initialStars}
                options={[
                  { value: 1, label: '☆' },
                  { value: 2, label: '☆☆' },
                  { value: 3, label: '☆☆☆' },
                  { value: 4, label: '☆☆☆☆' },
                  { value: 5, label: '☆☆☆☆☆' },
                ]}
              />
            </InputWrapper>
            <InputWrapper>
              <label htmlFor="country">Страна</label>
              <CustomSelector
                name="country"
                placeholder="Выберите..."
                options={countries}
                initialValue={initialCountry}
              />
            </InputWrapper>
            <Button type="submit" disabled={submitting}>
              Обновить
            </Button>
          </StyledForm>
        )
      }
    />
  </>
 );
};
```

PageEditCountry/:

PageEditCountry.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;
`;
```

PageEditCountry.tsx:

```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 { getCountryById, updateCountryById } 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 './PageEditCountry.styles';

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

export const PageEditCountry = () => {
  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);

    getCountryById(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 updateCountryById(parseInt(id || '0'), values).then(
        async (value) => {
          if (value) {
            await sleep(2000);
```

```
        } else {
          console.error('Error while PUT country 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>
            <Button type="submit" disabled={submitting}>
              Обновить
            </Button>
          </StyledForm>
        )
```

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

PageHotel/:

PageHotel.tsx:

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

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

export const PageHotel = (): JSX.Element => {
  const [hotel, setHotel] = useState<HotelDataType>();
  const { id } = useParams();

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

  if (hotel) {
    return (
      <>
        <Navbar />
        <Card data={hotel} 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 { getHotels } from 'api';
import { HotelDataType } 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 [hotels, setHotels] = useState<HotelDataType[]>();
  const [params, setParams] = useSearchParams();

  useEffect(() => {
    getHotels().then((data) => {
      if (params.get('country')) {
        const parsedData = data?.filter((elem) => {
          if (params.get('country')) {
            return (
              elem.country_code === parseInt(params.get('country') || '-1')
            );
          } else {
            return true;
          }
        });
        setHotels(parsedData);
      } else {
        if (params.get('hotelName')) {
          const parsedData = data?.filter((elem) => {
            if (params.get('hotelName')) {
              return elem.name.includes(params.get('hotelName') || '');
            } else {
              return true;
            }
          });
          setHotels(parsedData);
        } else {
          setHotels(data);
        }
      }
    });
  }, [params]);

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

```
  }, []);

  return (
    <PageMainWrapper>
      <Navbar detailed onChange={setParams} />
      {hotels?.map((hotel) => (
        <Card
          key={hotel.pk}
          data={hotel}
          onRemove={() => onRemoveClick(hotel.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/PageAddCountry';
import { PageAddHotel } from 'pages/PageAddHotel';
import { PageCountries } from 'pages/PageCountries';
import { PageCountry } from 'pages/PageCountry';
import { PageEdit } from 'pages/PageEdit';
import { PageEditCountry } from 'pages/PageEditCountry';
import { PageHotel } from 'pages/PageHotel';
import { PageMain } from 'pages/PageMain';

import './App.css';

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

        {/* Hotel block */}
        <Route path="/hotel" element={<PageMain />} />
        <Route path="/hotel/add" element={<PageAddHotel />} />
        <Route path="/hotel/:id/edit" element={<PageEdit />} />
        <Route path="/hotel/:id" element={<PageHotel />} />
```

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

export default App;
```

Dockerfile

```
FROM node:13.12.0-alpine

WORKDIR /usr/src/frontend

COPY package.json .

RUN npm install typescript -g
RUN npm install --silent
RUN npm link typescript

COPY . .

CMD ["npm", "start"]
```

webpack.config.js

```
const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');
const webpack = require('webpack');

module.exports = {
    mode: 'none',
    entry: {
        app: path.join(__dirname, 'src', 'index.tsx')
    },
    target: 'web',
    resolve: {
        extensions: ['.ts', '.tsx', '.js'],
        alias: {
            pages: path.resolve(__dirname, 'src/pages/'),
            components: path.resolve(__dirname, 'src/components/'),
            api: path.resolve(__dirname, 'src/api/'),
            utils: path.resolve(__dirname, 'src/utils/'),
            assets: path.resolve(__dirname, 'src/assets/'),
```

```
            constants: path.resolve(__dirname, 'src/constants/'),

        },
    },
    module: {
        rules: [{
                test: /\.tsx?$/,
                use: 'ts-loader',
                exclude: '/node_modules/'
            },
            {
                test: /\.css$/i,
                use: ["style-loader", "css-loader"],
            },
            {
                test: /\.svg$/,
                use: ['@svgr/webpack'],
            },
            {
                test: /\.(png|svg|jpg|jpeg|gif|ico)$/,
                exclude: /node_modules/,
                use: ['file-loader?name=[name].[ext]']
            }
        ],
    },
    devServer: {
        static: {
            directory: path.join(__dirname, 'public'),
        },
        port: 3000,
        host: '0.0.0.0'
    },
    watchOptions: {
        aggregateTimeout: 500,
        ignored: [path.posix.resolve(__dirname, './node_modules')],
        poll: 1000
    },
    output: {
        filename: '[name].js',
        path: path.resolve(__dirname, 'dist')
    },
    plugins: [
        new webpack.DefinePlugin({
            'process.env.NODE_ENV': JSON.stringify(process.env.NODE_ENV),
            'process.env.MY_ENV': JSON.stringify(process.env.MY_ENV),
        }),
        new HtmlWebpackPlugin({
            template: './public/index.html',
            filename: './index.html',
            favicon: './public/favicon.ico'
        })
```

```
    ]
}
```

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/

hotel_model/

models.py

```
from django.db import models

class Country(models.Model):
    name = models.CharField(max_length=100)

    class Meta:
        managed = True
        db_table = 'Country'


class Hotel(models.Model):
    name = models.CharField(max_length=255, blank=True, null=True)
    url = models.CharField(max_length=999, blank=True, null=True)
    imageSrc = models.CharField(max_length=999, blank=True, null=True)
    stars = models.IntegerField(blank=True, null=True)
    country_code = models.ForeignKey(Country, on_delete=models.CASCADE)

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

serializers.py

```
from hotel_model.models import Country, Hotel
from rest_framework import serializers
```

```
class CountrySerializer(serializers.ModelSerializer):
    class Meta:
        model = Country
        fields = ("pk", "name")


class HotelSerializer(serializers.ModelSerializer):
    def create(self, validated_data):
        return Hotel.objects.create(**validated_data)
    country = CountrySerializer(source='country_code', required=False)
    class Meta:
        model = Hotel
        fields = ("pk", "name", "url", "imageSrc", "stars", "country_code",
"country")
```

views.py

```
from hotel_model.models import Country, Hotel
from hotel_model.serializers import CountrySerializer, HotelSerializer
from rest_framework.generics import ListCreateAPIView, ListAPIView,
RetrieveUpdateDestroyAPIView
from rest_framework.generics import get_object_or_404
from rest_framework.response import Response

class CountryView(ListCreateAPIView):
    queryset = Country.objects.all()
    serializer_class = CountrySerializer

class SingleCountryView(RetrieveUpdateDestroyAPIView):
    queryset = Country.objects.all()
    serializer_class = CountrySerializer

class HotelView(ListAPIView):
    queryset = Hotel.objects.all()
    serializer_class = HotelSerializer

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

class SingleHotelView(RetrieveUpdateDestroyAPIView):
    queryset = Hotel.objects.all()
    serializer_class = HotelSerializer
```

hotels/

urls.py

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

urlpatterns = [
    path('country/', hotel_views.CountryView.as_view()),
    path('country/<int:pk>', hotel_views.SingleCountryView.as_view()),

    path('hotel/', hotel_views.HotelView.as_view()),
    path('hotel/<int:pk>', hotel_views.SingleHotelView.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',

    'hotel_model'
]

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 = 'hotels.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 = 'hotels.wsgi.application'


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

DATABASES = {
```

```
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'hotels',
        'USER': 'django',
        'PASSWORD': 'root',
        'HOST': 'db',
        'PORT': 3306,
        'OPTIONS': {'charset': 'utf8'},
        'TEST_CHARSET': 'utf8',
    }
}


# 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


# 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',
]
```

Docker.py

```
FROM python:3.8
ENV PYTHONUNBUFFERED 1
RUN mkdir /backend
WORKDIR /backend
COPY requirements.txt /backend/
RUN pip install --upgrade pip && pip install -r requirements.txt
ADD . /backend/
```

requirements.txt

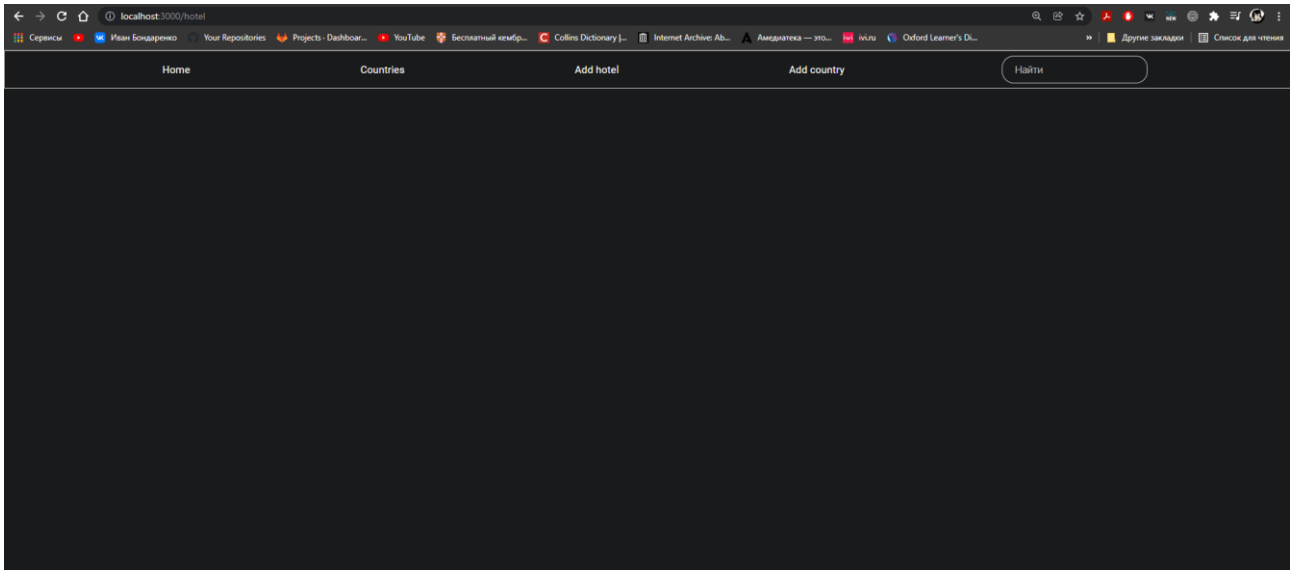
```
Django==3.1.1
mysqlclient==2.0.1
django-mysql==3.8.1
djangorestframework==3.13.1
django-cors-headers==3.10.1
drf-writable-nested==0.6.3
```

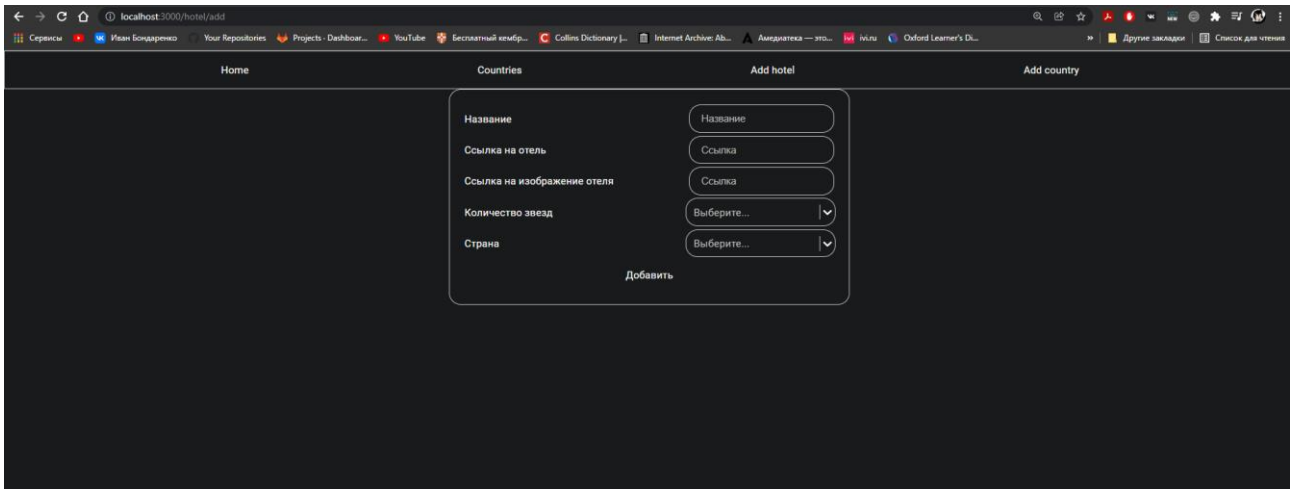
./docker-compose.yml

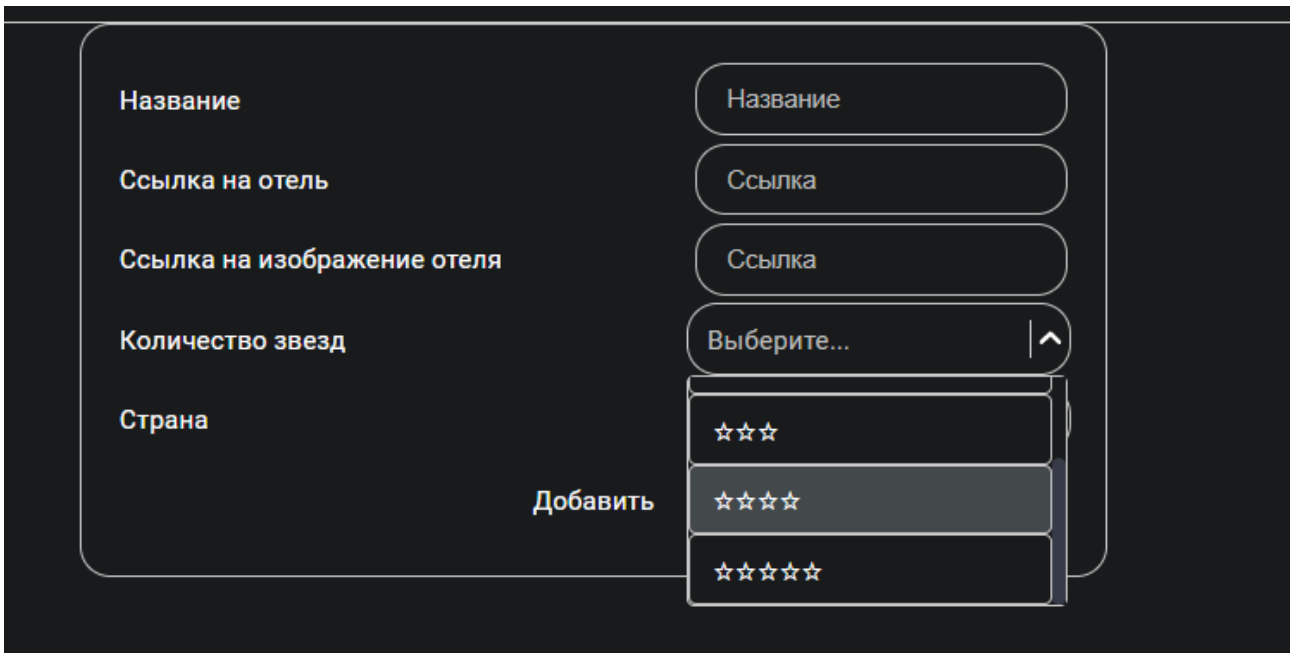
```
version: '3'

services:
  db:
    image: mysql:5.7
    ports:
      - '3306:3306'
    environment:
      MYSQL_DATABASE: 'hotels'
      MYSQL_USER: 'django'
      MYSQL_PASSWORD: 'root'
```

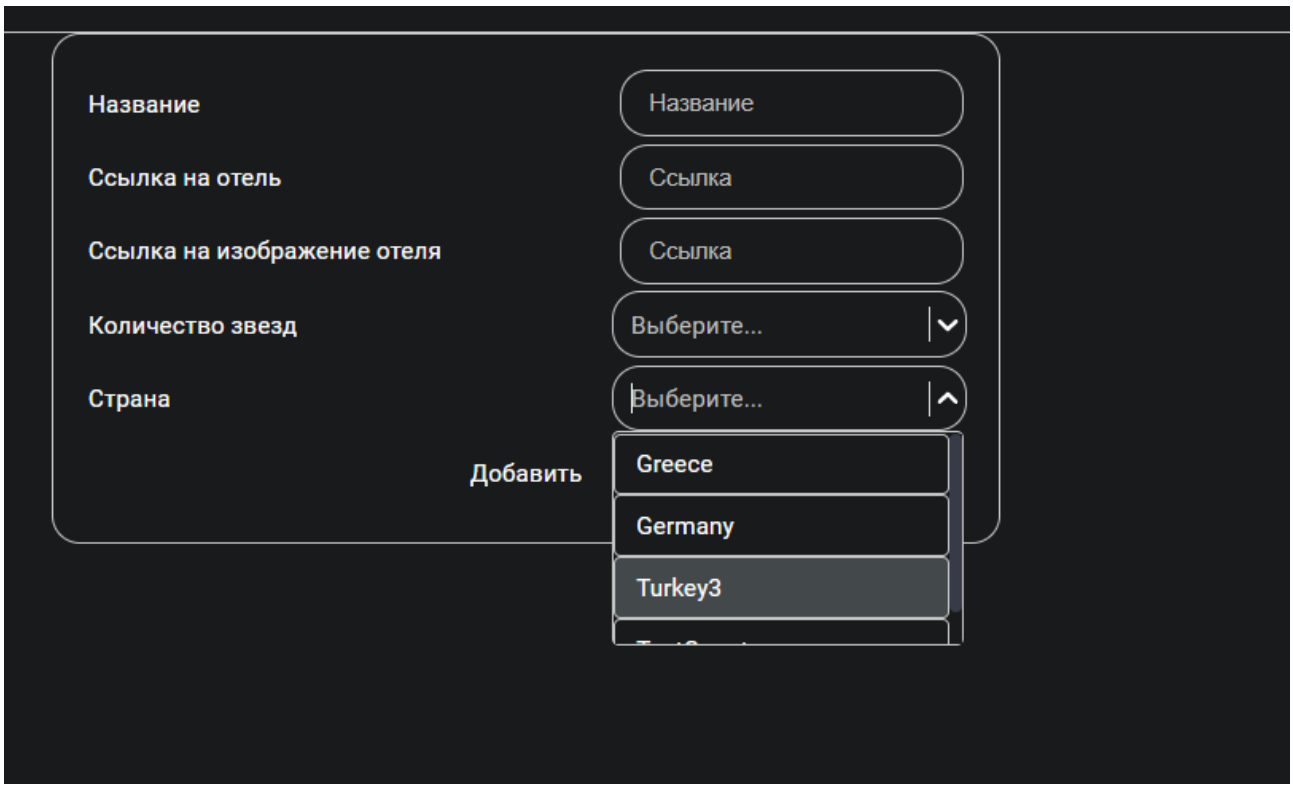
```
    MYSQL_ROOT_PASSWORD: 'root'
    MYSQL_RANDOM_ROOT_PASSWORD: 'yes'
  restart: always
  volumes:
    - ./db_django:/var/lib/mysql
frontend:
  build: ./client
  environment:
    CHOKIDAR_USEPOLLING: "true"
  volumes:
    - ./client:/usr/src/frontend
  ports:
    - '3000:3000'
  depends_on:
    - backend
backend:
  build: ./server
  environment:
    CHOKIDAR_USEPOLLING: "true"
  command: python manage.py runserver 0.0.0.0:8000
  volumes:
    - ./server:/backend
  ports:
    - '8000:8000'
  depends_on:
    - db
```

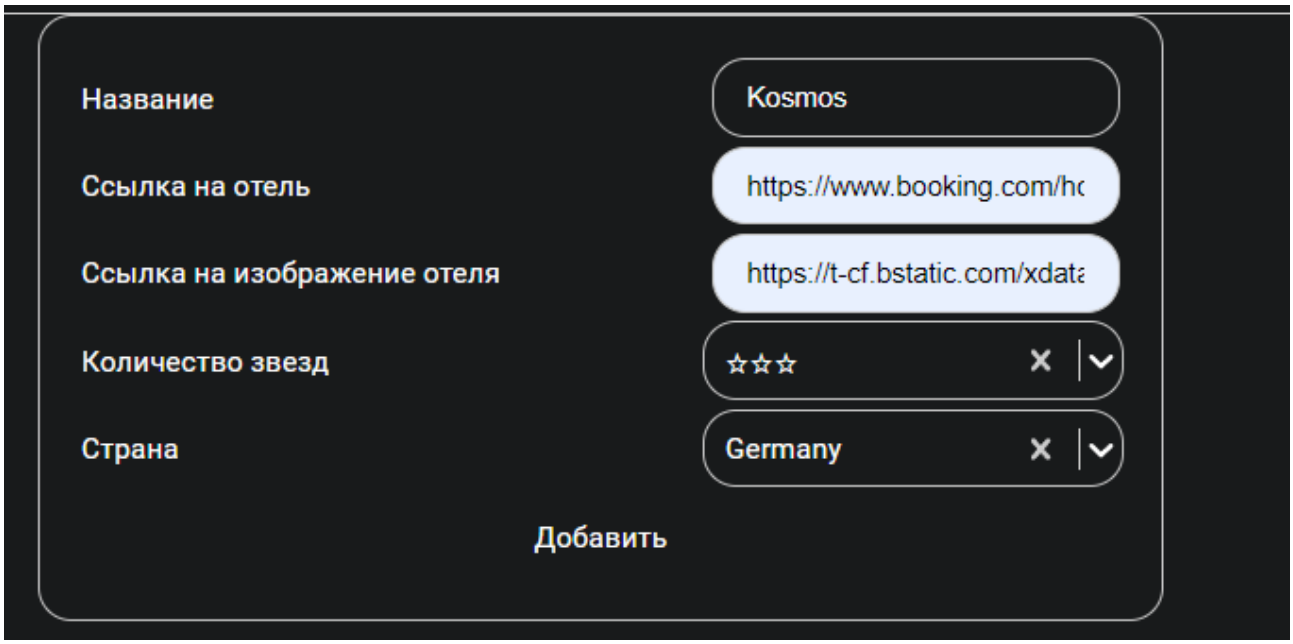
Экранные формы с примерами выполнения программы:

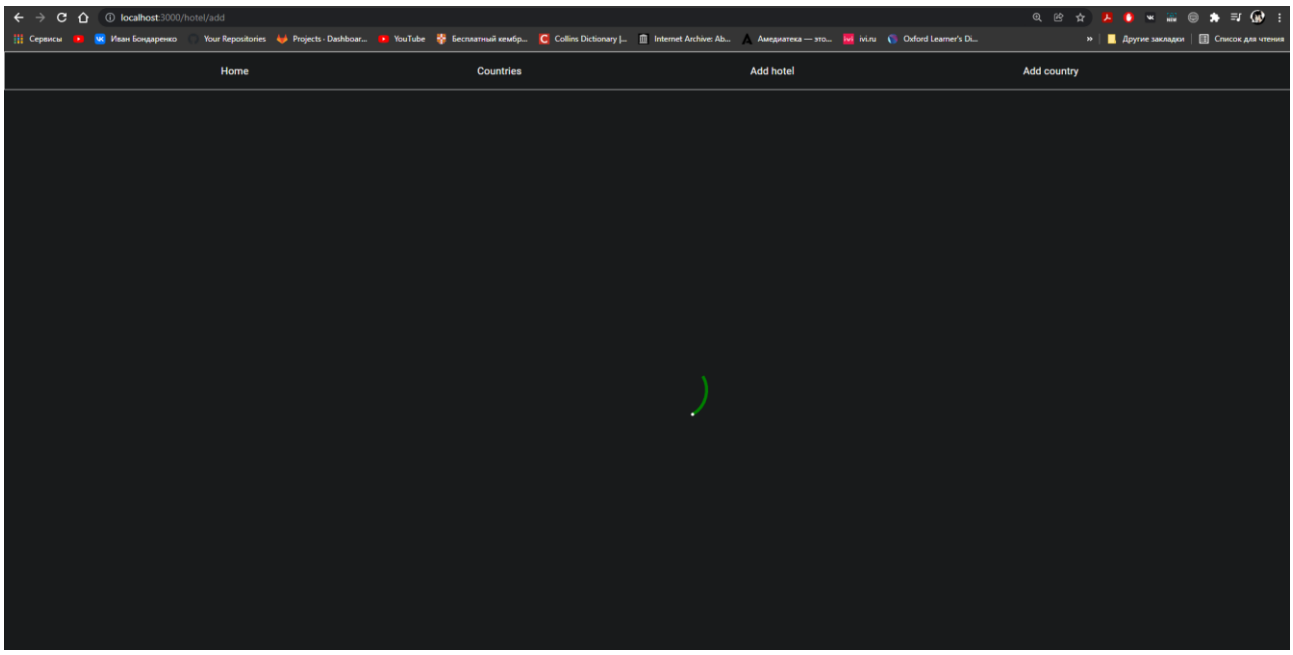


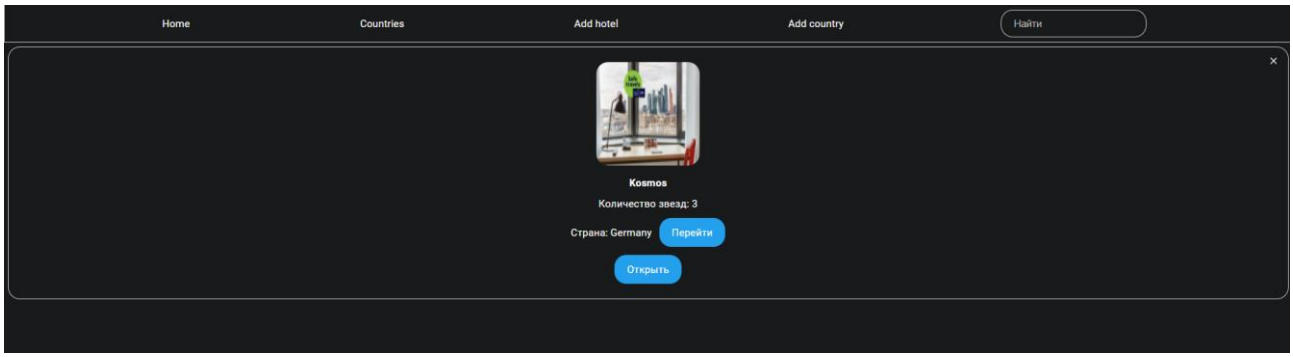


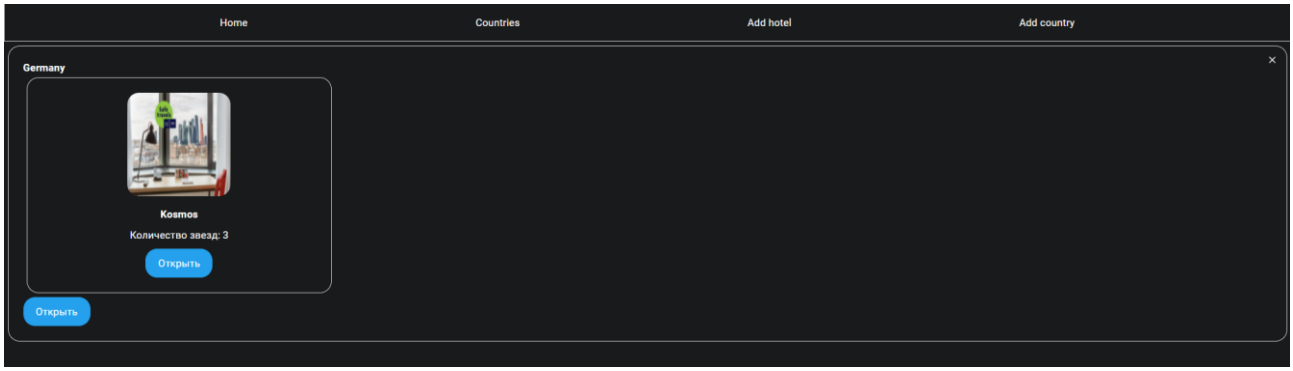


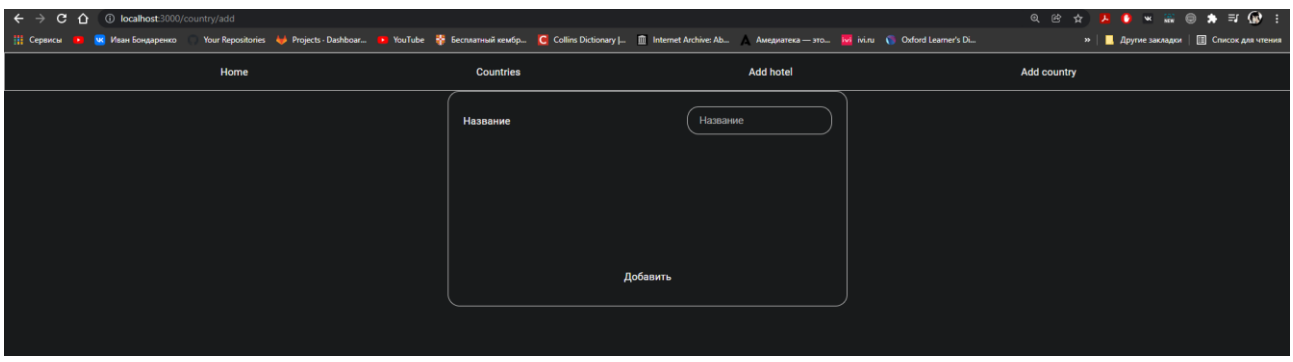

Название
Название
Ссылка на отель
Ссылка
Ссылка на изображение отеля
Ссылка
Количество звезд
Выберите...
Страна
Выберите...
Добавить
Greece
Germany
Turkey3

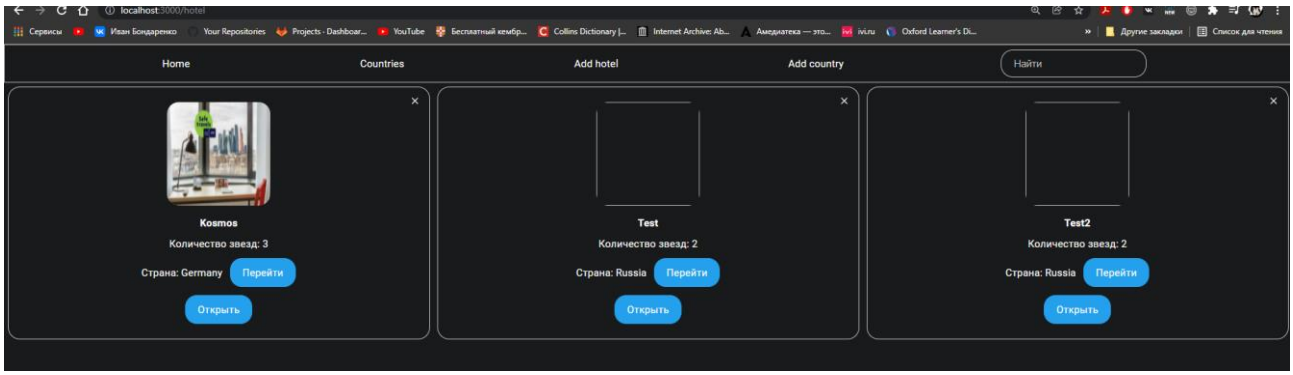

Название
Kosmos
Ссылка на отель
https://www.booking.com/hc
Ссылка на изображение отеля
https://t-cf.bstatic.com/xdata
Количество звезд
☆☆☆
Страна
Germany
Добавить

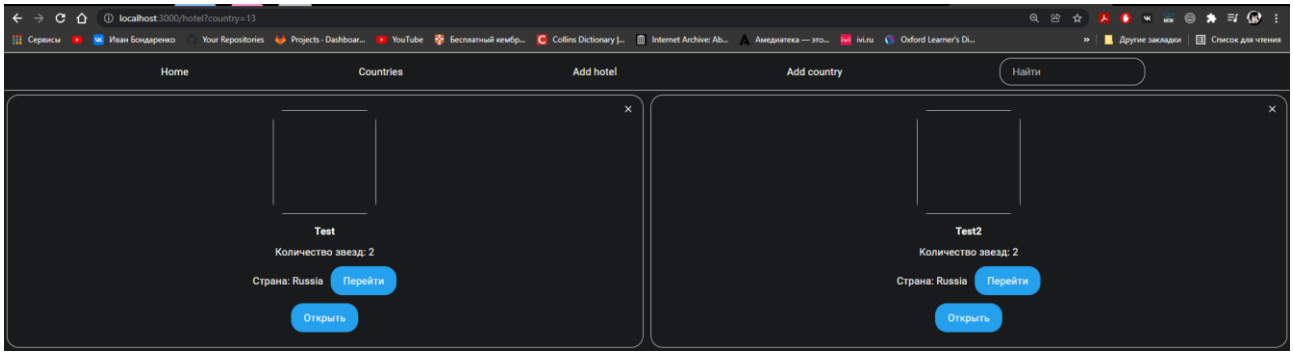

localhost:3000/hotel/add
Home
Countries
Add hotel
Add country

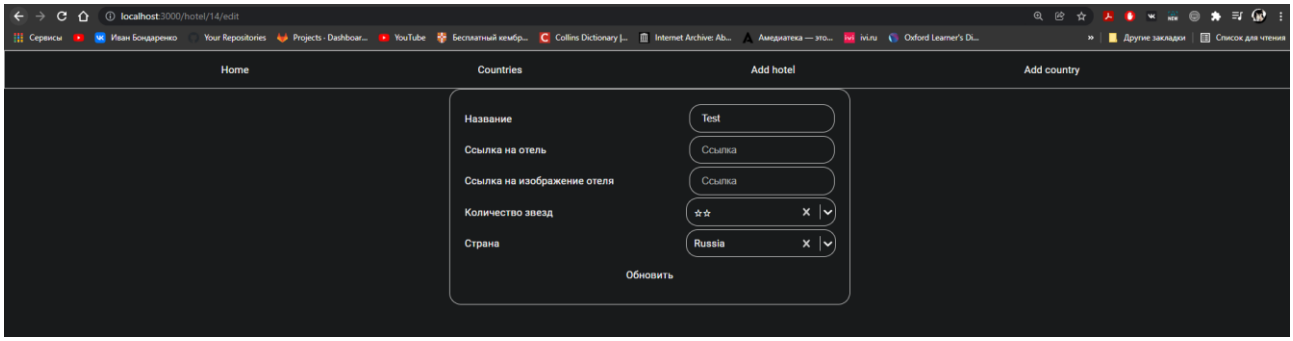

Home
Countries
Add hotel
Add country
Найти
Kosmos
Количество звезд: 3
Страна: Germany
Перейти
Открыть


Home
Countries
Add hotel
Add country
Germany
Kosmos
Количество звезд: 3
Открыть
Открыть


localhost:3000/country/add
Home
Countries
Add hotel
Add country
Название
Название
Добавить


localhost:3000/hotel
Home
Countries
Add hotel
Add country
Найти
Kosmos
Количество звезд: 3
Страна: Germany
Перейти
Открыть
Test
Количество звезд: 2
Страна: Russia
Перейти
Открыть
Test2
Количество звезд: 2
Страна: Russia
Перейти
Открыть


localhost:3000/hotel?country=13
Home
Countries
Add hotel
Add country
Найти
Test
Количество звезд: 2
Страна: Russia
Перейти
Открыть
Test2
Количество звезд: 2
Страна: Russia
Перейти
Открыть


localhost:3000/hotel/14/edit
Home
Countries
Add hotel
Add country
Название
Test
Ссылка на отель
Ссылка
Ссылка на изображение отеля
Ссылка
Количество звезд
Страна
Russia
Обновить