Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет программной инженерии и компьютерной техники

ВЫПУСКНАЯ АТТЕСТАЦИОННАЯ РАБОТА

ВЕБ-СЕРВИС ДЛЯ ПОИСКА ТУРИСТИЧЕСКОГО НАПРАВЛЕНИЯ

Автор: Зюзина И. М.
Направление подготовки:
«Веб-разработчик»

Руководитель: Сивинский С. А.

Санкт-Петербург 2025 г.

Содержание

ВВЕДЕНИЕ

Целью дипломной работы является создание веб-сервиса для поиска идеи для путешествия (географического места на карте, соответствующего пожеланиям пользователя). В период расширения возможностей передвижения по миру, оптимизации транспортных систем, активного использования глобальной сети и автоматизированных систем поиска это имеет большое значение для тех, кто хочет выбрать вариант отдыха под индивидуальные запросы. В российской сети интернет присутствует огромное количество порталов и сервисов, предлагающих бронирование туров, отелей, билетов на все виды транспорта и пр. Но практически отсутствуют сайты, предлагающие подбор самой идеи для путешествия и выбора туристического направления.

Из наиболее близких к этому туристических порталов известны следующие:

- www.turizm.ru. Помимо стандартной линейки поиска туров, отелей и билетов здесь реализована система поиска места для путешествия с выбором страны или вида отдыха, а также типа отдыха - «красивые виды», «великолепная архитектура», «путешествие с друзьями», «вкусная еда», «погружение в историю», «семейный отдых», «романтическое путешествие», «шопинг», «пешие прогулки», «загородный отдых» и пр. Результат представлен в виде страницы с описанием предлагаемого объекта и его особенностей, начальной стоимостью отелей, курсом валюты, визовыми требованиями, количеством часов для перелета в место назначения, погодой, отзывами. Также вниманию посетителя сайта в результатах поиска предлагается подборка отелей, список достопримечательностей, статьи и помощь консультанта.
- travel.ru. Представлены модули поиска отелей и билетов, новости и путеводитель для туриста. А также раздел «Куда поехать отдыхать?», где реализован поиск направления для отдыха с фильтрацией по месяцам года. В наличии 12 страниц («январь», «февраль», «март» и т.д), на каждой из которых информация представлена в следующих блоках: открытые страны, пляжные курорты, экскурсионные туры, с детьми, по России, активный отдых, круизы, праздники и фестивали и пр. Весь текст размещен на одной странице, переходы по

разделам осуществляются по «якорным» ссылкам.

- russia.travel. Есть подборки направлений для путешествий, советы. Система поиска - по регионам России, географическим направлениям, типам отдыха.
- tourdom.ru. Новости. Подборки популярных направлений.
- needtravel.ru. Системы поиска авиабилетов, отелей, туров. Подбор направления отсутствует.
- youtravel.me. Сайт с авторскими турами. Есть подбор направления («Куда хотите») с фильтрацией по датам и странам. А также подборки с предложениями.

Список туристических сайтов можно продолжать до бесконечности, но стандартный набор для поиска - туры, отели, билеты (плюс страховки, трансферы и прочие дополнительные услуги). Однако уровень и объем туристического траффика за последние годы выросли до такой степени, когда стандартные, легкодоступные и популярные направления отдыха перестали удовлетворять запросам многих туристов.

Практическое применение нашего веб-сервиса возможно на туристических сайтах и порталах и состоит в удовлетворении потребностей в поиске информации для тех, кто ищет нестандартный или наиболее подходящий под текущие запросы и возможности вариант отдыха. При этом сервис должен в качестве результатов выдавать не только саму идею направления или места, куда можно отправиться, но и информацию о минимальной стоимости перелёта до пункта назначения, о климатической зоне, часовом поясе, валюте, описание места назначения и пр. Фильтрация и подбор может проходить по следующим критериям:

- тип отдыха (основной критерий)
- город вылета (необязательный параметр).

В ходе работы появляется возможность демонстрации навыков работы в следующих технологиях:

(1) HTML, CSS (в данном проекте использован SCSS),

(2) Javascript (включая ES6),

(3) сборщик модулей Vite,

(4) транспилятор Babel,

(5) Фреймворк React,

(6) TypeScript,

(7) Node.js и фреймворк Express,

(8) библиотека Axios,

(9) СУБД PostgreSQL и PGAdmin4

(10) ORM Sequelize

(11) механизм Cross-Origin Resource Sharing (CORS)

(12) контейнеризация с помощью Docker.

Проект размещён на сервисе github и доступен по адресу https://github.com/Irina1781/TravelFinal

Ход выполнения работы

Основные функции

Функционал сервиса позволяет пользователю осуществлять работу с массивом записей в базе данных, который необходимо фильтровать по определенным параметрам (полям).

В рамках веб-сервиса пользователь может производить фильтрацию по полям: «Тип отдыха» (type) и «Город вылета» (city). Основной упор в данной работе сделан на функционале фильтрации записей с возможностями отбора по одному или нескольким тэгам.

Для каждого объекта базы данных предусмотрено заполнение следующих полей:

- название, краткое описание, географическое расположение
- фото
- **тип отдыха** (пляжный, семейный, гастрономический, событийный, экскурсионный, оздоровительный, треккинг и пешие маршруты и пр.). Поле type доступно для фильтрации
- краткое описание
- географическое расположение
- валюта
- климат. Выбор из тропического, умеренного, полярного и экваториального и др. видов.
- часовой пояс
- минимальная стоимость перелёта
- доступные города вылета.

Клиентская часть

Клиентская часть выполнена с помощью библиотеки (фреймворка) React и языка TypeScript.

В клиентской части предусмотрены 2 основные части:

1. «Интерфейс поиска». Модуль, отображаемый при загрузке и фильтр по параметрам. Под ним - кнопка запуска поиска «Найти» и кнопка «Очистить».

2. «Результаты поиска» - часть интерфейса с выводом результатов фильтрации в кратком представлении.

Реализация результатов в виде ленты, в которой для каждого объекта указаны:

- Краткое описание, гео и фото
- Климат
- Часовой пояс
- Валюта
- Стоимость перелёта туда-обратно от... руб.
- Подробнее - ссылка на полную информацию.

Интерфейс поиска

В верхней части страницы располагается фильтр, с помощью которого можно настроить параметры поиска. Варианты по каждому фильтру предлагаются в виде кнопки с выпадающим списком. Отбор данных после выбора пользователя ведётся либо по одному (обязательному) параметру с множественным выбором («Тип отдыха»), либо по нему, а также по полю «Город вылета» (необязательному) одновременно.

Предусмотрены кнопки «Найти», «Очистить».

В фильтр встроены механизмы валидации. Например:

- если не выбран ни один параметр, то кнопка «Найти» останется неактивной,
- если не выбран параметр в обязательных полях поиска по типам отдыха, то кнопка «Найти» останется неактивной, а пользователь увидит уведомление «Ничего не найдено».

Компонентная структура интерфейса поиска

Все компоненты интерфейса расположены в папке /.front. Интерфейс поиска включает в себя: заголовки H1, H2, H6, кнопки «Найти» (Button) и «Очистить» (Button), а также два выпадающих кастимозированных списка Dropdown. Один - с выбором единичного (SINGLE) варианта города вылета, второй - с возможностью мультивыбора (MULTIPLE) из предложенных вариантов типов отдыха. Они описаны в файлах Dropdown.tsx, Menuitem.tsx и resstype.tsx и доступны по адресу ./src/components. Основное содержимое страницы формируется в компоненте TravelSearch на странице TravelSearch.tsx.

Чтобы содержимое меню не обрезалось границами модального окна, используется *порталирование*. Для позиционирования меню вводится тип Coord, где хранчтся текущие координаты меню, и переменная controlRef для элемента Control. Для вычисления этих координат создана функция getCoords. Она вычисляет координаты и, в случае, если не удалось это сделать, возвращает значение null. Если координаты присутствуют, они устанавливаются в useEffect в случае, если меню открыто.

В стилях константы Menu, имеющей позицию absolute, эти координаты также используются:

```
const Menu = styled.menu<{ coords: Coords }>`
  position: absolute;
  left: ${(p) => `${p.coords.left}px`};
  top: ${(p) => `${p.coords.top}px`};
  min-width: ${(p) => `${Math.max(100, p.coords.width)}px`};
`;
```

Для закрытия модального окна при клике за его пределами используется Backdrop onClick={() => setOpen(false)}

По сути это элемент div размерами на всю ширину и высоту окна.

```
const Backdrop = styled.div`
  position: fixed;
  inset: 0;
`;
```

При клике по нему происходит закрытие меню.

Dropdown

Выпадающее меню - один из основных элементов интерфейса. Показывается он только в случае, если состояние isOpen равно true.

```
const handleOpen = () => setOpen(true);
```

За пункты меню отвечает файл MenuItem.tsx, где основной компонент обладает свойствами:

active: *показывает, активен ли элемент*

disabled: *запрещает выбор элемента, в коде по умолчанию неактивен*

value: *параметр*

onClick?(): *метод, описывающий события при клике по элементу*.

Константа Root служит, по сути, контейнером для компонента.

За возможность выбора сразу нескольких вариантов из предложенного списка отвечает компонент enum Behavior. В нем указаны возможности выбора как одного, так и нескольких элементов. Для негов типах прописаны как общие свойства, так и свойства для выбора одного (SingleProps) и нескольких (MultipleProps) вариантов. В последних двух указываем behaviour и соответствующие варианты onChange и value.

Свойства label и children отвечают за содержимое элемента соответственно до момента открытия и после выбора пунктов выпадающего меню.

Сначала формируется массив с индексами, в котором каждый элемент проверяется на валидность с помощью isValidElement (является ли элементов

React), не заблокирован ли он в свойстве disabled и является ли он MenuItem. В случае соответствия всем требованиям его индекс добавляется в массив indexes.

```
const indexes = useMemo(
  () =>
    items.reduce<Array<number>>((result, item, index) => {
      if (React.isValidElement(item)) {
        if (item.props.disabled !== true && item.type === MenuItem) {
          result.push(index);
        }
      }
```

Свойство highlightedIndex обозначает позицию элемента в массиве, где хранятся валидные индексы элементов.

Для каждого элемента задается ref для хранения списка элементов:

```
const elements = useRef<Record<number, HTMLDivElement>>({});
..
{Children.map(children, (child, index) => {
            if (isValidElement(child)) {
              return cloneElement(child, {
                active: index === indexes[highlightedIndex],
                onMouseEnter: () =>
                  setHighlightedIndex(indexes.indexOf(index)),
                onClick: (ev: MouseEvent) => {
                  ev.stopPropagation();
                  !child.props.disabled && handleChange(child.props.value);
                },
                ref: (node: HTMLDivElement) =>
                  (elements.current[index] = node),
              });
```

Для управления состоянием формы используется метод handleChange, использующий хук useCallback для предотвращения повторного рендеринга компонентов. Он срабатывает при выборе пользователем новых входных данных и подтверждении выбора с помощью клика мышкой или нажатием Enter.

За работу с клавиатурой отвечает функция handleKeyDown. Она предполагает выбор и подсветку элементов при перемещении стрелок вдоль списка вниз и вверх, а также действия при нажатии клавиши Enter - запуск метода handleChange. Свойство length используется с целью при достижении конца списка перехода на первый элемент, при достижении начала - на последний.

В функции также настроен скроллинг на случай, если фокусированный элемент окажется за границами меню:

```
elements.current[indexes[index]]?.scrollIntoView({
block: "nearest",
```

При выборе элементов на клавиатуре срабатывают функции с добавлением и удалением «слушателя событий»:

```
 useEffect(() => {
   if (isOpen) {
     document.addEventListener("keydown", handleKeyDown, true);
   }

   return () => {
     document.removeEventListener("keydown", handleKeyDown, true);
   };
 }, [isOpen, handleKeyDown]);
```

За динамическую работу мышкой с элементами списка отвечает функция Children.map. Выбранный пользователем элемент копируется с помощью cloneElement и помечается свойством active, а также подсвечивается при выборе мышкой и при клике.

Filter

Компонент Filter отвечает за рендеринг компонентов поиска и полученных данных после выбора пользователем нужных ему вариантов. Здесь прописаны возможные варианты выбора городов и типа отдыха из списков выпадающего меню (файл constants.tx), типы данных, по которым ведётся поиск (types.ts), интерфейс и настройки поиска и фильтрации данных, в том числе работа кнопок «Найти» (метод handleUpdate) и «Очистить» (метод handleReset). Основная часть методов, отвечающих за работу фильтра, прописана в файле Filter.tsx.

Из файла Dropdown.tsx сюда импортируется перечисление Behaviour и метод Dropdown с полученным от пользователя индексом выбранного элемента.

Интерфейс FilterInterface определяет структуру объекта со свойствами handleSearch, climateList и timezoneList. Последние два (список значений таблиц climate и timezone) принимают значения, импортируемые из файла types.ts в компоненте TravelSearch.

В методе Filter константам присваиваются значения выбранных пользователем элементов из списков city и type. В событии handleUpdate, которое «висит» на кнопке «Найти», с помощью асинхронной функции async/await идёт обращение к базе данных, производится поиск по параметрам типа отдыха и возвращается константа handleSearch со значениями city и type. Кнопке «Очистить» соответствует событие handleReset, обнуляющее при нажатии на кнопку значения констант setCity и setType.

Results

За вывод результатов поиска отвечает **компонент Results**. Здесь формируется интерфейс с выводом списка значений базы данных, соответствующих параметрам выбора.

В файле List.tsx создаётся лента результатов поиска в виде карточек с указанием названия предлагаемого места отдыха, возможными городами вылета, кратким описанием, географическим расположением, указанием типа климата,

часового пояса в формате UTM, местной валюты и минимальной стоимости перелёта от Москвы в обе стороны.

Так как текстовые варианты для полей city, timezone и climate находятся не в основной, а в связанных таблицах, а в основной таблице их значения представлены числовым типом, потребовалось прописать константы getClimateName, getTimezoneName и getCityName, в которых принимается id табличной записи, а на выходе возвращается ее текстовое значение.

Сам метод List с параметрами results, climate, timezone и cityList выводит в качестве результата Listcontainer с итогами мэппинга, сформированными в виде карточек (CardContainer).

В файле Results.tsx формируется интерфейс ResultInterface с теми же параметрами из List.

TravelSearch

Папка pages предназначена для хранения кода страниц созданного сайта. Пока здесь находится единственная страница TravelSearch.

В файле TravelSearch.tsx импортируется константа Results и Filter из соответствующих компонентов. С помощью хука UseEffect и функции async/await константам getClimateList, getCityList и getTimezoneList после запроса к таблицам базы данных передаются значения списков городов вылета, часовых поясов и климатических зон.

Сама страница отвечает за рендеринг всех компонентов интерфейса с последовательным выводом:

- заголовков H1 и H2
- компонента Filter с отрисовкой интерфейса поиска с выпадающими списками и двумя кнопками
- компонента Results с выводом результатов поиска.

Управление стилями веб-сервиса

Дизайн приложения не являлся фокусом данной работы. Основной упор был сделан на алгоритмическом функционале. Для оформления стилей использовалась стандартная таблица стилей styles.css. Стили также частично прописаны в сопутствующих файлах компонентов. Дополнительно использовался модуль styled-components.

Транспиляция клиентской части кода

Клиентская часть кода использует синтаксис TSX, SCSS. С целью транспиляции кода в браузерный Javascript был использованы инструменты Babel, Vite.

Серверная часть

Серверная часть написана на Node.js (Express) и следует паттерну MVC. Для взаимодействия с базой данных используется Docker Compose.

База данных

Для хранения записей и тэгов используется объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом PostgreSQL.

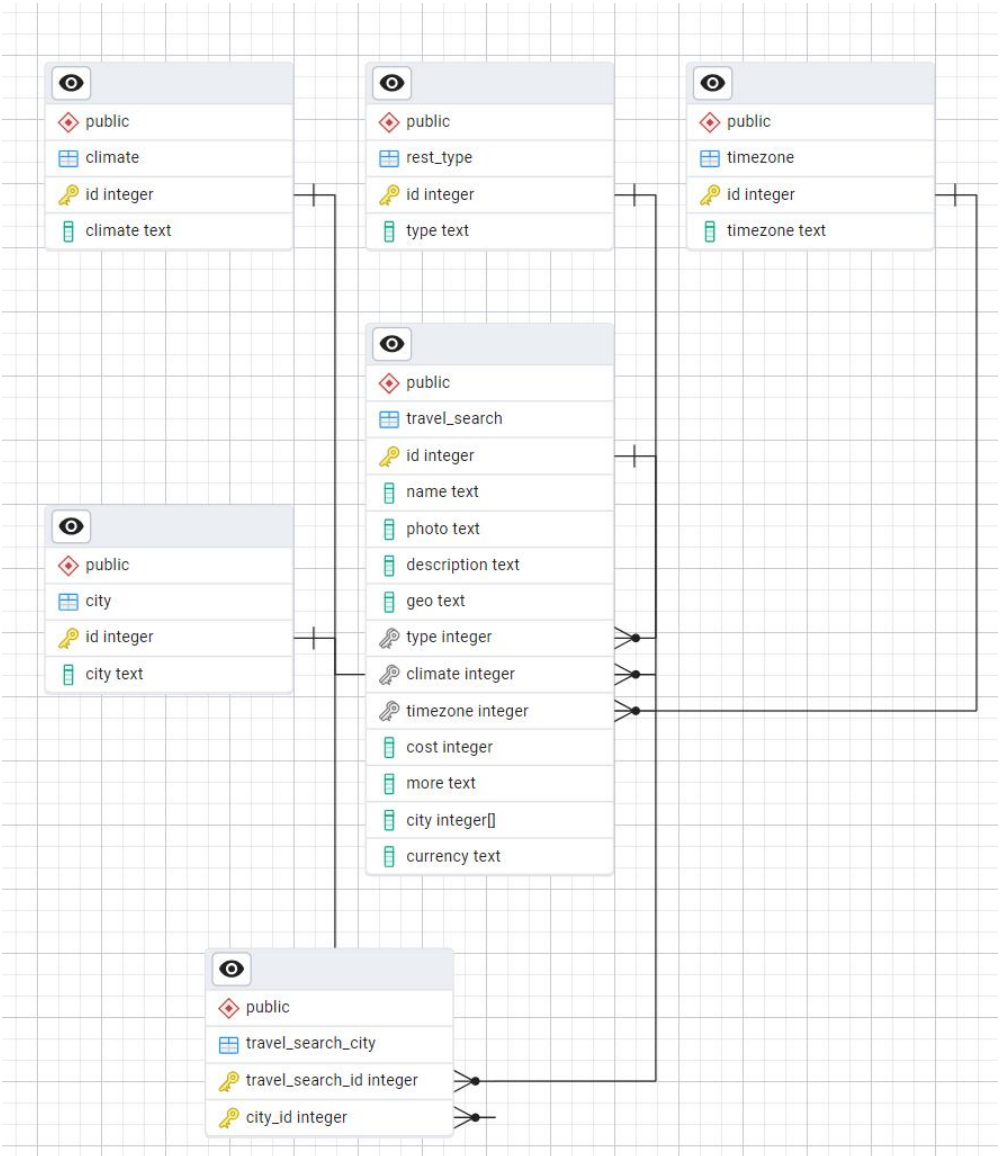
Данные хранятся в пяти таблицах:

(1) основная таблица с записями: **travel_search**,

(2) связанные вспомогательные таблицы:

- city (город вылета)
- climate (климат)
- rest_type (тип отдыха)
- timezone (часовой пояс)
- travel_search_city (для сопоставления типов данных между travel_search и city).

Таблицы связаны между собой по полю id. Связи отображены в следующей схеме:



Развертывание

Для целей демонстрации приложение можно развернуть с помощью docker-compose.

Конфигурационные файлы для создания образов и контейнеров - Dockerfile и docker-compose.yml. Последний расположен в корневой части проекта и содержит все инструкции для создания и запуска всех контейнеров.

Два dockerfiles добавлены в бэкенд и фронтенд и содержат перечень инструкций для развертывания соответствующих компонентов. Для серверной и пользовательской частей инструкция предполагает установку Node версии 19, установку зависимостей через npm install, обновление файлов package.json и

package-lock.json и запуск основного скрипта при запуске контейнера.

В процессе развёртывания предполагается создание и запуск нескольких контейнеров:

- project_postgres (конфигурация для запуска контейнера на основе образа PostgreDQL 17.2 с docker hub, к которой «подтягивается» дамп БД migrate.sql)
- project_pgadmin (с перенаправлением docker-хоста 8080 на локальный порт 80)
- project_back (на основе postgres_db с назначением порта 3000)
- project_front (с зависимостью от backend и развёртыванием интерфейса на локальном порту 8180).

Формирование образов, создание и запуск контейнеров производится командой **docker-compose up -d**. Или вариант с флагами **docker-compose up -d --force-recreate**.

Таким образом при развёртывании приложения не происходит установки PostgreSQL на локальный ПК. Используется образ, скачанный с docker-hub и запущенный в контейнере. Так же как и само приложение запускается не локально, а в контейнере, перенаправленном на локальный хост с портом 8180.

Архив для развёртывания также размещён на сервисе GitHub по ссылке https://github.com/Irina1781/TravelRep

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта подкреплены дополнительно на практике навыки работы по следующему диапазону технологий:

(1) HTML, CSS

(2) Javascript (использовался React, TypeScrypt), что позволило отработать широко применяемую на практике библиотеку React

(3) Babel, Vite

(4) Node.js (фреймворк Express.js)

(5) PostgreSQL(PGAdmin4).

СПИСОК ИСПОЛЬЗОВАННЫХ РЕСУРСОВ

1. Библиотека React. https://reactjs.org/.
2. Язык TypeScript https://www.typescriptlang.org/
3. Фреймворк Express Node.js. https://expressjs.com/ru/.
4. База данных PostgreSQL. https://www.postgresql.org/
5. Транспилятор Babel. https://babeljs.io/.
6. Бандлер Vite. https://vitejs.ru/
7. ORM Sequelize. https://sequelize.org/
8. Платформа Docker. https://www.docker.com/