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 **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

БАКАЛАВРСКАЯ ПРОГРАММА **09.03.01/03 Вычислительные машины, комплексы, системы и сети**

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Тип практики Эксплуатационная

Название предприятия НУК ИУ МГТУ им. Н.Э. Баумана

Студент ИУ6-63Б — 09.09.2024 (Подпись, дата) — Ю.В. Лазарева (И.О. Фамилия)

Руководитель практики от МГТУ им. Н.Э.Баумана — 09.09.2024 (Подпись, дата) — С.С Данилюк (И.О. Фамилия)

Оценка Отлично

2024 г.

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)

З А Д А Н И Е
на производственную практику

по теме ПРИЛОЖЕНИЕ ПОЗВОЛЯЮЩЕЕ ПОЛЬЗОВАТЕЛЮ САЙТА ВЫБРАТЬ ИНТЕРЕСУЮЩИЕ ЕГО КАТЕГОРИИ МАГАЗИНОВ

Студент группы ИУ6-63Б

Лазарева Юлия Валерьевна
(Фамилия, имя, отчество)

Направление подготовки 09.03.01 Информатика и вычислительная техника

Бакалаврская программа 09.03.01/03 Вычислительные машины, комплексы, системы и сети

Тип практики Эксплуатационная практика

Название предприятия НУК ИУ МГТУ им. Н.Э. Баумана

Техническое задание

Создать одностраничное (SPA) приложение, которое позволяет посетителю сайта выбрать интересующие его категории магазинов (список категорий придумать самостоятельно) и отобразить эти магазины на карте рядом с посетителем, используя геолокацию

Оформление отчета по практике:

Отчет на 15-25 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
нет

Дата выдачи задания « 28 » июня 2024 г.

Руководитель практики от МГТУ им. Н.Э.Баумана	28.06.2024 (Подпись, дата)	С.С. Данилюк (И.О. Фамилия)
Студент	28.06.2024 (Подпись, дата)	Ю.В. Лазарева (И.О. Фамилия)

Примечание: Задание оформляется в двух экземплярах.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель практики заключалась в приобретении навыков и знаний, необходимых для разработки одностраничного приложения (SPA) [1], способного осуществлять поиск магазинов, интересующих пользователей, таких как рестораны, аптеки, супермаркеты и другие, вблизи их текущего местоположения, и отображение их на карте. В частности, для достижения этой цели были поставлены следующие задачи:

– изучение технологий для создания SPA-приложений;

– определение требований к разрабатываемому приложению;

– выбор стека технологий для поставленной цели;

– создание функционального и интуитивного веб-интерфейса (обеспечение пользователям возможности выбора категорий магазинов);

– использование веб-технологий для определения геолокации пользователя (интеграция нескольких API, таких как геолокационные сервисы);

– отображение результатов поиска на интерактивной карте (осуществление интеграции картографических сервисов для визуализации данных);

– обеспечение адаптивности приложения (возможности комфортного использования на различных устройствах).

В целом можно сказать, что одной из задач данной практики являлось опытным путём понять, как современные веб-технологии могут быть эффективно использованы для создания полезных и удобных сервисов, способных значительно улучшить повседневную жизнь пользователей.

1 Реализация задуманного функционала

В рамках практики были определены и реализованы несколько ключевых функций, обеспечивающих взаимодействие пользователя с приложением. Эти функции включают получение текущей геопозиции пользователя, выбор категорий магазинов, поиск и отображение магазинов на карте, а также динамическое обновление карты. Ниже приведено более детальное описание каждого из этих пунктов, включая примеры кода на JavaScript.

1.1 Получение текущей геопозиции пользователя

Одной из первых задач приложения является получение текущего местоположения пользователя. Это достигается с помощью встроенного API геолокации, который запрашивает разрешение на доступ к геоданным при загрузке страницы.

Листинг 1 – Пример кода для инициализации карты

```
function getUserLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(
            (position) => {
                const latitude = position.coords.latitude;
                const longitude = position.coords.longitude;
                initializeMap(latitude, longitude);
            },
            (error) => {
                console.error('Error getting location:',
error);
                alert('Не удалось получить ваше местоположение.
Пожалуйста, разрешите доступ к геоданным.');
            }
        );
    } else {
        alert('Ваш браузер не поддерживает геолокацию.');
    }
}
```

В этом коде navigator.geolocation.getCurrentPosition используется для получения координат пользователя. При успешном определении

местоположения функция initializeMap инициализирует карту с фокусом на текущей точке.

1.2 Выбор категорий магазинов

Была реализована возможность выбирать категории магазинов, которые пользователь желает увидеть на карте. Это было сделано при помощи пользовательского интерфейса, в котором представляются различные категории, такие как рестораны, аптеки, супермаркеты и т.д.

Листинг 2 – Пример кода для выбора категорий

```
const categories = ['restaurants', 'pharmacies',
'supermarkets'];
const selectedCategories = [];

function toggleCategory(category) {
    const index = selectedCategories.indexOf(category);
    if (index > -1) {
        selectedCategories.splice(index, 1);
    } else {
        selectedCategories.push(category);
    }
    updateMapWithSelectedCategories();
}
```

Здесь selectedCategories хранит выбранные пользователем категории. Функция toggleCategory добавляет или удаляет категории из этого списка в зависимости от выбора пользователя. Затем updateMapWithSelectedCategories обновляет отображение карты в соответствии с выбранными категориями. Выпадающий список для выбора категорий изображён на рисунке 1.

1.3 Поиск и отображение магазинов

После того, как пользователь выбрал интересующие его категории, приложение выполняет поиск соответствующих магазинов вблизи текущего местоположения пользователя и отображает их на карте.

Листинг 3 – Пример кода для поиска и отображения магазинов

```
function fetchStoresByCategory(category, latitude, longitude) {
    fetch(apiUrl)
        .then((response) => response.json())
        .then((data) => {
```

Продолжение листинга 3

```
            data.stores.forEach((store) => {
                addMarkerToMap(store);
            });
        })
        .catch((error) => {
            console.error('Error fetching stores:', error);
        });
}

function updateMapWithSelectedCategories() {
    const { latitude, longitude } = getCurrentMapCenter();
    selectedCategories.forEach((category) => {
        fetchStoresByCategory(category, latitude, longitude);
    });

}
```

В этом коде функция fetchStoresByCategory запрашивает данные о магазинах по выбранной категории и текущим координатам. Затем магазины отображаются на карте с помощью addMarkerToMap.

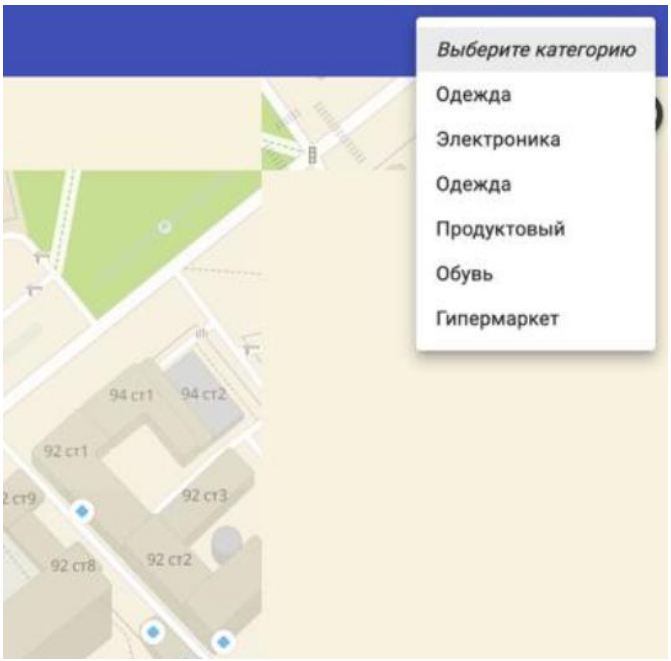

Рисунок 1 – Снимок экрана выбора категорий

1.4 Отображение карты и установка маркеров

Карта приложения динамически обновляется в зависимости от выбранных категорий и местоположения пользователя. Каждый найденный

магазин отображается на карте с помощью маркера, который обозначает его расположение.

Листинг 4 – Пример кода для отображения карты

```
let map;
function initializeMap(latitude, longitude) {
    map = new DG.Map(document.getElementById('map'), {
        center: { lat: latitude, lng: longitude },
        zoom: 14,
    });
    new DG.marker({
        position: { lat: latitude, lng: longitude },
        map: map,
        title: 'Ваше местоположение',
    });
}

function addMarkerToMap(store) {
    const marker = new DG.marker({
        position: { lat: store.latitude, lng:
store.longitude },
        map: map,
        title: store.name,
    });
…
}
```

Функция initializeMap создаёт карту, центрированную на текущем местоположении пользователя. Маркер для обозначения текущей позиции также добавляется. Функция addMarkerToMap отвечает за добавление маркеров для каждого найденного магазина, отображая их на карте.

2 API для определения и отображения геопозиции

Для реализации функционала определения и отображения геопозиции пользователя в приложении использовались несколько API [4]. Каждое из них играло свою роль в создании интерактивной карты и предоставлении пользователю информации о ближайших магазинах.

2.1 2GIS Maps API

2GIS Maps API был использован для отображения карты, установки маркеров и работы с геолокацией пользователя. Этот API предоставляет инструменты для работы с картографическими данными и взаимодействия с ними [2]. Функция addMarker добавляет маркер на карту, указывая местоположение магазина или пользователя. Функция updateMapCenter обновляет центр карты в зависимости от текущего местоположения пользователя.

Карта, отображенная с помощью 2GIS Maps API, представлена на рисунке 2.

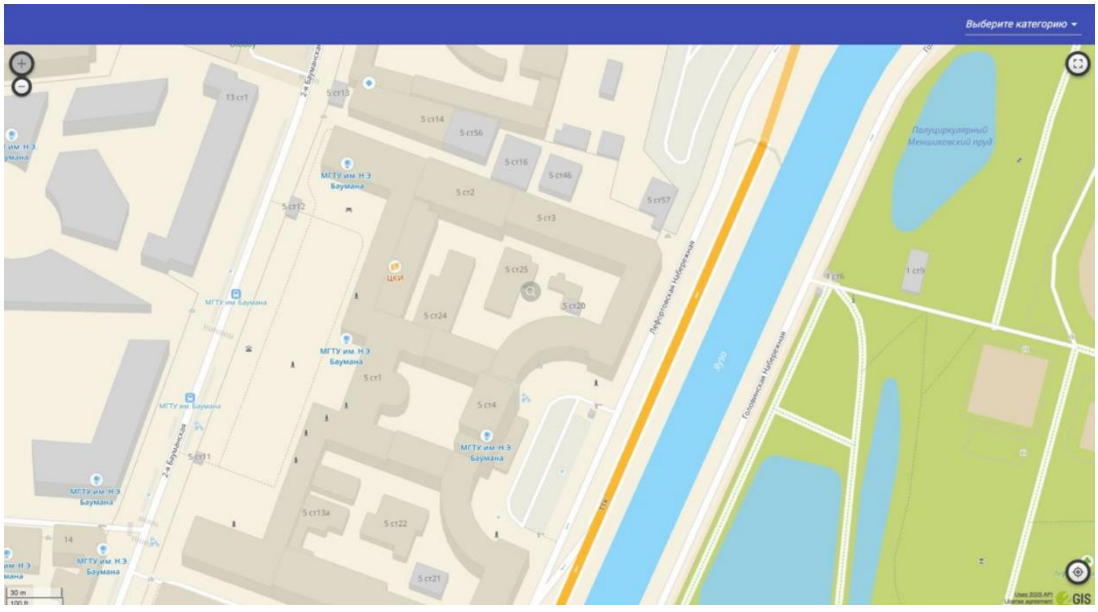

Рисунок 2 – Снимок экрана интерфейса разработанного приложения

2.2 Geolocation API браузера

Изначально для определения местоположения использовался стандартный Geolocation API браузера, который позволяет получать геопозицию устройства. Однако этот API имеет ограничения в некоторых

ситуациях, например, когда пользователь отказывает в доступе к геоданным или использует устройство без GPS.

2.3 IPStack API

IPStack API использовался как резервный механизм для определения местоположения пользователя по IP-адресу в случае отказа от использования Geolocation API. Это решение эффективно, когда пользователь блокирует доступ к геоданным, но доступен интернет.

Листинг 5 – Пример кода для использования IPStack API

```
function useFallbackLocation() {

fetch('http://api.ipstack.com/check?access_key=YOUR_ACCESS_KEY'
)
        .then((response) => response.json())
        .then((data) => {
            const { latitude, longitude, city } = data;
            updateMapCenter(latitude, longitude);
            addMarker(latitude, longitude, `Ваше местоположение
(по IP): ${city}`);
        })
        .catch((error) => {
            console.error('Ошибка при получении геолокации
через IPStack:', error);
        });
}
```

В данном примере API ipstack.com возвращает данные о местоположении пользователя на основе его IP-адреса. Затем карта центрируется на полученных координатах, и на карту добавляется маркер с указанием города.

2.4 API для получения данных о магазинах

Для получения данных о магазинах и их отображения на карте использовался 2GIS Catalog API. Этот API предоставляет информацию о различных предприятиях и позволяет искать магазины в выбранной категории, используя координаты пользователя.

Листинг 6 – Пример кода для получения данных о магазинах с использованием 2GIS Catalog API

```
function fetchStores(category, latitude, longitude) {
    const apiUrl =
`https://catalog.api.2gis.com/2.0/catalog/branch/search?key=YOU
R_API_KEY&q=${category}&point=${longitude},${latitude}&radius=1
000`;

    fetch(apiUrl)
        .then((response) => response.json())
        .then((data) => {
            data.result.items.forEach((store) => {
                const { lat, lon, name } = store.point;
                addMarker(lat, lon, name);
            });
        })
        .catch((error) => {
            console.error('Ошибка при получении данных о
магазинах:', error);
        });
}
```

Этот код отправляет запрос к 2GIS Catalog API, чтобы получить список магазинов по выбранной категории. Для каждого магазина, найденного API, добавляется маркер на карту с использованием координат магазина и его названия.

3 Выбор стека технологий разработки

Для реализации приложения были использованы современные веб-технологии, которые обеспечили высокую производительность и удобство разработки. В частности, были применены React, Redux, и Material-UI. Ниже описаны их основные роли в проекте, а также представлены примеры кода, иллюстрирующие их использование.

3.1 React

React был выбран в качестве основной библиотеки для построения пользовательского интерфейса. Он обеспечил возможность разбиения приложения на отдельные компоненты, что упростило управление кодом и его повторное использование. Основные элементы приложения, такие как карта и панель навигации, были реализованы в виде отдельных React-компонентов.

В данном примере MapComponent использует React Hook useEffect, чтобы инициализировать карту при первом рендере и обновлять ее при изменении местоположения или маркеров. Это позволяет компоненту быть динамическим и реагировать на изменения состояния.

3.2 Redux

Redux был использован для централизованного управления состоянием приложения. Он позволил эффективно управлять данными, которые поступали из различных API, и координировать взаимодействие между компонентами. Redux обеспечил хранение состояний местоположения пользователя, выбранных категорий магазинов и других фильтров [5].

Листинг 7 – Пример кода для создания Redux-хранилища и редуктора

```
import { createStore } from 'redux';

const initialState = {
    location: { latitude: null, longitude: null },
    selectedCategory: null,
    markers: [],
};

function appReducer(state = initialState, action) {
    switch (action.type) {
```

Продолжение листинга 7

```
case 'SET_LOCATION':
            return { ...state, location: action.payload };
        case 'SET_CATEGORY':
            return { ...state, selectedCategory:
action.payload };
        case 'SET_MARKERS':
            return { ...state, markers: action.payload };
        default:
            return state;
    }
}

const store = createStore(appReducer);

export default store;
```

Здесь appReducer обрабатывает различные действия, такие как SET_LOCATION для обновления местоположения пользователя, SET_CATEGORY для выбора категории магазинов и SET_MARKERS для установки маркеров на карте. Хранилище store обеспечивает централизованное управление этими состояниями.

3.3 Material-UI

Material-UI была использована для оформления пользовательского интерфейса. Эта библиотека стилей предоставляет готовые компоненты и темы, что позволило быстро создать красивый и интуитивно понятный интерфейс, соответствующий современным стандартам дизайна [3].

Листинг 8 – Пример кода для создания панели навигации с использованием Material-UI

```
const Navbar = ({ selectedCategory, onCategoryChange }) => {
    return (
        <AppBar position="static">
            <Toolbar>
                <Typography variant="h6" style={{ flexGrow:
1 }}>
                    Поиск Магазинов
                </Typography>
                <Select
                    value={selectedCategory}
```

Продолжение листинга 8

```
                    onChange={onCategoryChange}
                    variant="outlined"
                    style={{ color: 'white' }}
                >
                    <MenuItem
value="restaurant">Рестораны</MenuItem>
                    <MenuItem
value="pharmacy">Аптеки</MenuItem>
                    <MenuItem
value="supermarket">Супермаркеты</MenuItem>
                </Select>
            </Toolbar>
        </AppBar>
    );
};
```

Этот код создает панель навигации с заголовком и выпадающим списком категорий, который позволяет пользователю выбирать интересующую его категорию магазинов. Компоненты AppBar, Toolbar, и Select из Material-UI обеспечивают современный и функциональный интерфейс.

4 Краткое описание использования программы

Разработанная программа представляет собой одностраничное веб-приложение, которое позволяет пользователю находить магазины определенных категорий вблизи его текущего местоположения. Ниже представлена инструкция для работы с приложением.

Шаг 1. Откройте веб-приложение по указанной ссылке.

Сначала необходимо перейти по ссылке на веб-приложение. Приложение загружается в браузере и автоматически начинает инициализацию, готовясь к работе с геолокацией.

Шаг 2. Разрешите использование геоданных в браузере.

При первом запуске браузер запросит у вас разрешение на использование данных о вашем местоположении, так как приложение использует вашу текущую геопозицию для отображения ближайших магазинов. Без этого шага приложение не сможет корректно работать и показывать результаты на карте.

Шаг 3. Выберите интересующую категорию магазина в меню, расположенном в правом верхнем углу.

После того, как ваше местоположение будет определено, вы сможете выбрать категорию магазинов, которые хотите найти. Меню выбора категорий доступно в правом верхнем углу экрана. Варианты категорий включают такие пункты, как "Рестораны", "Аптеки", "Супермаркеты" и другие. Выберите нужную категорию, и приложение начнет поиск магазинов, соответствующих вашему запросу.

Шаг 4. Нажмите на кнопку в правом нижнем углу экрана для отображения текущего местоположения и доступных магазинов на карте.

Для удобства взаимодействия с картой в правом нижнем углу экрана расположена кнопка, которая позволяет центрировать карту на вашем текущем местоположении. После нажатия на эту кнопку на карте отобразятся маркеры, указывающие расположение магазинов выбранной категории в радиусе от вас. Каждому магазину соответствует отдельный маркер, при нажатии на который можно получить дополнительную информацию.

ЗАКЛЮЧЕНИЕ

В рамках эксплуатационной практики были изучены технологии для создания SPA-приложений: React, Redux, Material-UI, а также приобретены навыки интеграции с API для определения геопозиции. Готовый продукт представляет собой удобный инструмент для пользователей, позволяющий быстро находить магазины различных категорий вблизи их текущего местоположения. Приложение полезно для широкого круга пользователей, от туристов до местных жителей, а также в дальнейшем может быть расширено и адаптировано под различные нужды, такие как доставка товаров и маркетинг, поиск услуг, туристических достопримечательностей или иных категорий объектов.

Существующие возможности приложения также могут быть легко расширены за счет добавления новых категорий магазинов, улучшения алгоритмов поиска и фильтрации, а также интеграции с дополнительными сервисами и базами данных, что делает его перспективным инструментом для решения ряда повседневных задач.

По окончании практики был написан отчёт, определение содержания и оформление которого осуществлялись с учетом требований ГОСТ 7.32.2017.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Guide To Single Page Applications [Электронный ресурс]. – Режим доступа: https://radixweb.com/blog/guide-to-single-page-applications (дата обращения: 02.07.2024).

2. 2GIS Documentation [Электронный ресурс]. – Режим доступа: https://docs.2gis.com/en (дата обращения: 10.07.2024).

3. Learning Resources – Material UI [Электронный ресурс]. – Режим доступа: https://mui.com/material-ui/getting-started/learn/ (дата обращения: 11.07.2024).

4. How To Consume REST APIs In React [Электронный ресурс]. – Режим доступа: https://www.freecodecamp.org/news/how-to-consume-rest-apis-in-react/ (дата обращения: 10.07.2024).

5. Redux Fundamentals [Электронный ресурс]. – Режим доступа: https://redux.js.org/tutorials/fundamentals/part-1-overview (дата обращения: 05.07.2024).