**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ГЕОДЕЗИИ И КАРТОГРАФИИ (МИИГАиК)**

Факультет: *Геоинформфционных систем и информационной безопасности*

Направление: *Информационные системы*

*и технологии*

**Курсовая работа**

по дисциплине «Методы и средства проектирования информационных систем и технологий»

**Выполнил**:

студент ФПКиФ,

группы ИСиТ 4-1Б,

Корякин В. А.

2021

Оглавление

[Введение 3](#__RefHeading___Toc801_1377500199)

[Веб-сервис 4](#__RefHeading___Toc803_1377500199)

[Преимущества и недостатки 4](#__RefHeading___Toc805_1377500199)

[Преимущества: 4](#__RefHeading___Toc807_1377500199)

[Недостатки: 4](#__RefHeading___Toc809_1377500199)

[Примеры 5](#__RefHeading___Toc811_1377500199)

[Технологии реализации 5](#__RefHeading___Toc813_1377500199)

[Архитектура: 5](#__RefHeading___Toc815_1377500199)

[Методы разработки: 6](#__RefHeading___Toc817_1377500199)

[Средства разработки веб-сервиса 7](#__RefHeading___Toc819_1377500199)

[Сервер 7](#__RefHeading___Toc821_1377500199)

[Среда разработки 9](#__RefHeading___Toc823_1377500199)

[Используемые технологии 9](#__RefHeading___Toc825_1377500199)

[HTML 9](#__RefHeading___Toc827_1377500199)

[CSS 9](#__RefHeading___Toc829_1377500199)

[JavaScript 10](#__RefHeading___Toc831_1377500199)

[Bootstrap 10](#__RefHeading___Toc833_1377500199)

[Leaflet 10](#__RefHeading___Toc835_1377500199)

[MapBox 11](#__RefHeading___Toc837_1377500199)

[Разработка веб-сервиса 12](#__RefHeading___Toc839_1377500199)

[Создание скелета основной веб-страницы 12](#__RefHeading___Toc841_1377500199)

[Динамическая загрузка содержимого с помощью JavaScript 12](#__RefHeading___Toc843_1377500199)

[Создание основы карты 12](#__RefHeading___Toc845_1377500199)

[Панель контроля 14](#__RefHeading___Toc847_1377500199)

[Хранение данных, формат JSON 15](#__RefHeading___Toc849_1377500199)

[Запросы на сервер 15](#__RefHeading___Toc851_1377500199)

[Создание маркеров отелей Leaflet 16](#__RefHeading___Toc853_1377500199)

[Создание Popup 18](#__RefHeading___Toc855_1377500199)

[Изменение иконки маркера 19](#__RefHeading___Toc857_1377500199)

[Создание левой части сервиса, список отелей 19](#__RefHeading___Toc859_1377500199)

[Создание функций и хендлеров 20](#__RefHeading___Toc861_1377500199)

[Окно информации 21](#__RefHeading___Toc863_1377500199)

[Создание маркеров пиков гор 23](#__RefHeading___Toc865_1377500199)

[Нанесение на карту информации о горнолыжных подъемниках 24](#__RefHeading___Toc867_1377500199)

[Заключение 26](#__RefHeading___Toc869_1377500199)

[Источники 27](#__RefHeading___Toc871_1377500199)

# Введение

**Цель курсовой работы**: создать сайт-сервис для территории Приэльбрусья

Для выполнения курсовой работы планируется изучить задачи согласно следующему плану:

* Понятие веб-сервиса;
* Практическое применение веб-сервиса;
* Изучить программные средства для разработки веб-сервиса;
* Изучить средства программирования веб-сервиса, в частности языки HTML, CSS, JavaScript.
* Использование картографической информации для расширения возможностей веб-сервиса;
* Изучить дополнительные возможности использования картографической информации в разрабатываемом сайте-сервисе;
* Изучить открытые источники данных картографической информации;
* Изучить возможности программной среды по использованию дополнительных библиотек, плагинов, фреймворков для представления картографической информации.

# **Веб-сервис**

**Веб-служба**, веб-сервис (англ. web service) — идентифицируемая уникальным веб-адресом (URL-адресом) программная система со стандартизированными интерфейсами.

Веб-службы могут взаимодействовать друг с другом и со сторонними приложениями посредством сообщений, основанных на определённых протоколах (SOAP, XML-RPC и т. д.) и соглашениях (REST). Веб-служба является единицей модульности при использовании сервис-ориентированной архитектуры приложения.

В обиходе веб-сервисами называют услуги, оказываемые в Интернете. В этом употреблении термин требует уточнения, идёт ли речь о поиске, веб-почте, хранении документов, файлов, закладок и т. п. Такими веб-сервисами можно пользоваться независимо от компьютера, браузера или места доступа в Интернет.

## Преимущества и недостатки

### Преимущества:

* Веб-службы обеспечивают взаимодействие программных систем независимо от платформы. Например, Windows-C#-клиент может обмениваться данными с Java-сервером, работающим под Linux.
* Веб-службы основаны на базе открытых стандартов и протоколов. Благодаря использованию XML достигается простота разработки и отладки веб-служб.
* Использование интернет-протокола обеспечивает HTTP-взаимодействие программных систем через межсетевой экран. Это значительное преимущество, по сравнению с такими технологиями, как CORBA, DCOM или Java RMI. С другой стороны, веб-службы не привязаны намертво к HTTP — могут использоваться и другие протоколы.

### Недостатки:

* Меньшая производительность и больший размер сетевого трафика по сравнению с технологиями RMI, CORBA, DCOM за счёт использования текстовых XML-сообщений. Однако на некоторых веб-серверах возможна настройка сжатия сетевого трафика.
* Аспекты безопасности. Ответственные веб-службы должны использовать кодирование, возможно — требовать аутентификации пользователя. Достаточно ли здесь применения HTTPS, или предпочтительны такие решения, как XML Signature, XML Encryption или SAML — должно быть решено разработчиком.

## Примеры

* **Магазин приложений** — интернет-магазин, позволяющий сторонним компаниям-разработчикам ПО предлагать владельцам мобильных (обычно) устройств устанавливать и приобретать различные приложения и игры, а производителям прочего контента — также книги, музыку и фильмы.
  + Магазин приложений Windows (Microsoft Store)
  + Магазин приложений Windows Phone (ранее Windows Phone Marketplace)
  + Магазин приложений Google/Android (Google Play)
  + Магазин приложений Chrome
  + Магазин приложений Apple (App Store)
  + Mac App Store
  + Opera Mobile Store
  + Samsung Galaxy Store
  + Магазин приложений Huawei (AppGallery)
  + Магазин приложений Яндекс (Яндекс.Store)
* **Маркетплейс –** предоставление коммерсантам торговой веб-площадки для торговли онлайн.
* **Стриминг –** предоставляют услуги по стримингу (выдаче потокового мультимедиа).
  + стриминг аудио (Spotify, Tidal, Roon, Qobuz , Muzlab и пр.);
  + стриминг видео (Netflix, Megogo, Disney+ и пр.).

## Технологии реализации

### Архитектура:

Можно выделить три инстанции, взаимодействующие в рамках веб-службы. Переведём их названия как:

* заказчик (service requester);
* исполнитель (service provider);
* каталог (service broker).

Когда служба разработана, исполнитель регистрирует её в каталоге, где её могут найти потенциальные заказчики. Заказчик, найдя в каталоге подходящую службу, импортирует оттуда её WSDL-спецификацию и разрабатывает в соответствии с ней своё программное обеспечение. WSDL описывает формат запросов и ответов, которыми обмениваются заказчик и исполнитель в процессе работы. Для обеспечения взаимодействия используются следующие стандарты:

* XML: Расширяемый язык разметки, предназначенный для хранения и передачи структурированных данных;
* SOAP: Протокол обмена сообщениями на базе XML;
* WSDL: Язык определения внешних интерфейсов, который определяет взаимодействие (контракт) между потребителем и веб-службами SOAP. Написан на базе XML;
* UDDI: Универсальный интерфейс распознавания, описания и интеграции (Universal Discovery, Description and Integration). Каталог веб-служб и сведений о компаниях, предоставляющих веб-службы во всеобщее пользование или конкретным компаниям. Пока UDDI существуют, однако, только в небольших фирменных сетях и ещё не нашли широкого распространения в открытом интернете;
* JSON: Более эффективный язык разметки, ставший массовым в 2010х годах.

### Методы разработки:

Существуют средства автоматизации разработки веб-служб, разделяющиеся на две основные группы. При разработке снизу-вверх сначала пишутся имплементирующие классы, а из их исходного текста генерируются WSDL-файлы, документирующие службу. Недостатком этого метода является подверженность Java-классов частым изменениям. При подходе "сверху вниз" сначала подготавливается WSDL, а из него генерируется скелет Java-класса, имплементирующего службу. Этот путь считается более трудным, зато приводит к более чистым и лучше защищенным от изменений решениям. Пока формат сообщений, которыми обмениваются заказчик и исполнитель, не меняется, изменения в каждом из них не нарушают взаимодействия. Эта техника называется иногда «contract first», так как исходной точкой является WSDL («договор» между заказчиком и исполнителем).

# Средства разработки веб-сервиса

## Сервер

Для того, чтобы удобнее работать с нашим сайтом топографического веб-сервиса было принято решение установить локальный сервер на операционной системе Linux Mint 20.1. В качестве программного средства было выбрано **Apache2** для настройки виртуального хоста для HTTP-сервера. Apache - это популярнейший свободный веб-сервер. Состоянием на 2020 год он используется на 33% всех сайтов интернета, а это приблизительно 304 миллиарда сайтов.

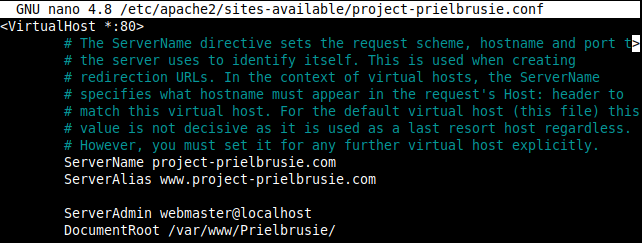
HTTP-сервер на Apache настраивается довольно просто:

* Устанавливаем программное средство с помощью консоли:  
  sudo apt install apache2
* После установки добавим веб-сервер в автозагрузку, чтобы не запускать его вручную после перезагрузки компьютера:  
  sudo systemctl enable apache2

Было бы не совсем удобно, если на одной физической машине можно было размещать только один сайт. Apache может поддерживать сотни сайтов на одном компьютере и выдавать для каждого из них правильное содержимое. Для этого используются виртуальные хосты. Сервер определяет к какому домену приходит запрос и отдает нужное содержимое из папки этого домена.

Настройки хостов Apache расположены в папке /etc/apache2/sites-available/. Для создания нового хоста достаточно создать файл с любым именем (лучше кончено с именем хоста) и заполнить его нужными данными. Обернуть все эти параметры нужно в директиву VirtualHost. Здесь будут использоваться такие:

* ServerName - основное имя домена;
* ServerAlias - дополнительное имя, по которому будет доступен сайт;
* ServerAdmin - электронная почта администратора;
* DocumentRoot - папка с документами для этого домена.



После создания файла конфигурации для нашего веб-сервера, необходимо его активировать:  
 sudo a2ensite project-prielbrusie.com

Настройка виртуальных хостов Apache завершена и на публичном сервере это все бы уже работало, но если вам нужна настройка Apache на домашней машине, то увы ваш новый сайт не откроется в браузере. Браузер не знает такого сайта. И откуда ему знать? DNS службы не могут ничего сообщить об этом доменном имени. Но в системе Linux мы можем сами указать ip адреса для доменных имен в файле /etc/hosts. Поэтому добавляем в конец файла такие строки:  
 nano /etc/hosts



## Среда разработки

В качестве среды разработки был выбран редактор исходного кода Visual Studio Code от компании Microsoft.

**Visual Studio Code** (VS Code) — редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

## Используемые технологии

### HTML

**HTML** (от англ. *HyperText Markup Language* — «язык гипертекстовой разметки») — стандартизированный язык разметки документов для просмотра веб-страниц в браузере. Веб-браузеры получают HTML документ от сервера по протоколам HTTP/HTTPS или открывают с локального диска, далее интерпретируют код в интерфейс, который будет отображаться на экране монитора.

Элементы HTML являются строительными блоками HTML страниц. С помощью HTML разные конструкции, изображения и другие объекты такие как интеракивная веб-формы могут быть встроены в отображаемую страницу. HTML предоставляет средства для создания заголовков, абзацев, списков, ссылок, цитат и других элементов. Элементы HTML выделяются тегами, записанными с использованием угловых скобок. Такие теги, как <img /> и <input />, напрямую вводят контент на страницу. Другие теги, такие как <p>, окружают и оформляют текст внутри себя и могут включать другие теги в качестве подэлементов. Браузеры не отображают HTML-теги, но используют их для интерпретации содержимого страницы.

В HTML можно встроить программый код на языке программирования JavaScript, для управления поведением и содержанием веб-страниц. Также включение CSS в HTML описывает внешний вид и макет страницы.

### CSS

**CSS** (англ. *Cascading Style Sheets* «каскадные таблицы стилей») — формальный язык описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки (чаще всего HTML или XHTML). Также может применяться к любым XML-документам, например, к SVG или XUL.

CSS используется создателями веб-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось отделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими шрифт Брайля.

### JavaScript

**JavaScript** (аббр. *JS*) — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

JavaScript является объектно-ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам — функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания — что придаёт языку дополнительную гибкость.

### Bootstrap

**Bootstrap** (также известен как Twitter Bootstrap) — свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

Bootstrap использует современные наработки в области CSS и HTML, поэтому необходимо быть внимательным при поддержке старых браузеров.

### Leaflet

**Leaflet** — библиотека с открытым исходным кодом, написанная на JavaScript, предназначенная для отображения карт на веб-сайтах. Поддерживает большинство мобильных и стационарных платформ из числа тех, что поддерживают HTML5 и CSS3.

Наряду с OpenLayers и Google Maps API — одна из наиболее популярных картографических JavaScript-библиотек, использующаяся на таких крупных сайтах, как Flickr, Foursquare, Craigslist, Data.gov, IGN, проектах Викимедиа, OpenStreetMap, Meetup, WSJ, MapBox, CloudMade, CartoDB и других.

Leaflet поддерживает слои Web Map Service (WMS), GeoJSON, векторные и тайловые слои. Многие другие типы слоёв поддерживаются дополнительными модулями.

Как и в других картографических веб-библиотеках, в Leaflet реализована следующая модель: отображается базовая карта с, возможно, растровыми и векторными слоями, накладываемыми поверх неё.

Основные типы объектов Leaflet:

* Растровые типы (TileLayer и ImageOverlay)
* Векторные типы (Path, Polygon и специфические типы, такие как Circle)
* Групповые типы (LayerGroup, FeatureGroup и GeoJSON)
* Управляющие элементы (Zoom, Layers и т. д.)

### MapBox

**Mapbox** является американским поставщиком пользовательских онлайн-карт для веб-сайтов и приложений, таких как Foursquare, Lonely Planet, Facebook, Financial Times, The Weather Channel и Snapchat. С 2010 года он быстро расширил нишу пользовательских карт, в ответ на ограниченный выбор, предложенный поставщиками карт, такими как Google Maps. Mapbox является создателем или активным участником некоторых открытых картографических библиотек и приложений, включая библиотеку Mapbox GL-JS JavaScript, спецификацию MBTiles, картографическую IDE TileMill, библиотеку Leaflet JavaScript, а также язык стилистики карт и парсер CartoCSS.

Данные берутся из открытых источников, таких как OpenStreetMap и NASA, а также из приобретенных собственных источников данных, таких как DigitalGlobe. Технология основана на Node.js, Mapnik, GDAL и Leaflet.

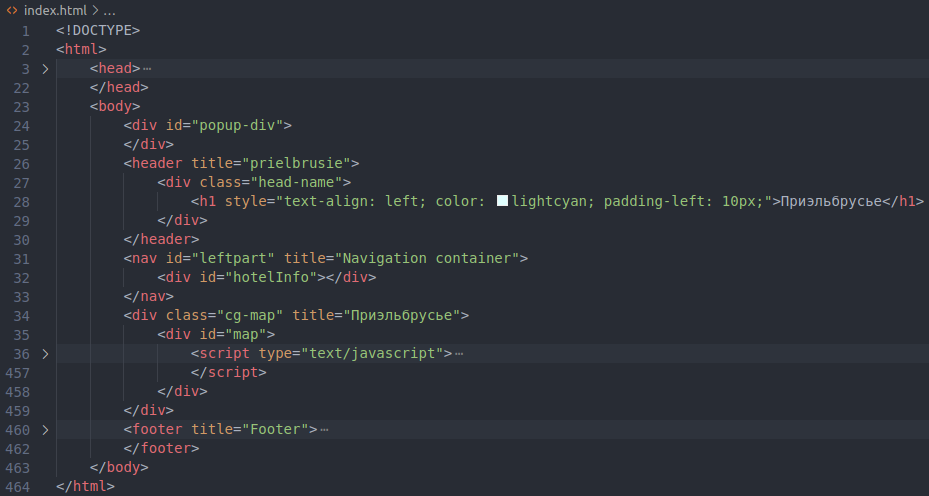
Mapbox использует анонимизированные данные из телеметрических пингов, таких как Strava и RunKeeper, для идентификации вероятных отсутствующих данных в OpenStreetMap с помощью автоматических методов, затем вручную применяет исправления или сообщает о проблеме разработчикам OSM.

# Разработка веб-сервиса

## Создание скелета основной веб-страницы

Для создания основы страницы веб-сервиса использовался HTML. Чтобы в дальнейшем не усложнять себе задачу был создан HTML-скелет будущей страницы сервиса, которая в дальнейшем будет дополняться файлами CSS-стилей: самописных файлов и заимствованных из фреймворков и библиотек (таких как Bootstrap и Leaflet). Основное содержание страницы будет заполняться посредством выполнения JavaScript скрипта, встроенного в страницу и динамически подгружающегося при загрузке этой страницы.

Пример оформления скелета представлен на рисунке.



## Динамическая загрузка содержимого с помощью JavaScript

### Создание основы карты

Для отображения карты на будущем веб-сервисе была использована библиотека **Leaflet**. В описании используемых технологий эта библиотека уже была представлена. На основе этой библиотеке мы и будем строить работу с картой. На карте будут использоваться несколько слоев листов (tile layer). Эти слои листов будут подгружаться из источника пользовательских карт MapBox, данные которых берутся из открытых источников, таких как OpenStreetMaps (OSM) и NASA.

Перед тем как добавлять карту на наш сайт, нам необходимо в области «head» скелета сайта указать ссылки на CSS-файл Leaflet и вставить JavaScript-скрипт от Leaflet, для того, чтобы наша карта смогла загрузиться и правильно отобразиться.





В части кода, расположенного чуть ниже, добавляются несколько слоев листов MapBox: слой со спутниковыми снимками (satelliteLayer) и слой с нарисованной векторной картой(outdoorLayer). Затем с использованием данных слоев создается карта, которая привязывается к контейнеру «div» с id=«map» на скелете страницы.



После этих операций карта уже отображается, и с ней можно работать.

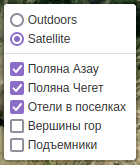


### Панель контроля

Мы добавили несколько слоев карты, но пока не можем переключаться между ними. Для этого нам необходимо настроить панель контроля слоев (Layer Control). Заодно добавим наши следующие слои, чтобы в будущем больше к ним не возвращаться.



Теперь мы можем переключаться между слоями карты, а еще нам доступны созданные нам пустые слои, которые в будущем будут отображать нам информацию об отелях в на различных участках Приэльбрусья, горнолыжных подъемниках, вершинах гор.



Теперь можно добавлять данные на карту.

### Хранение данных, формат JSON

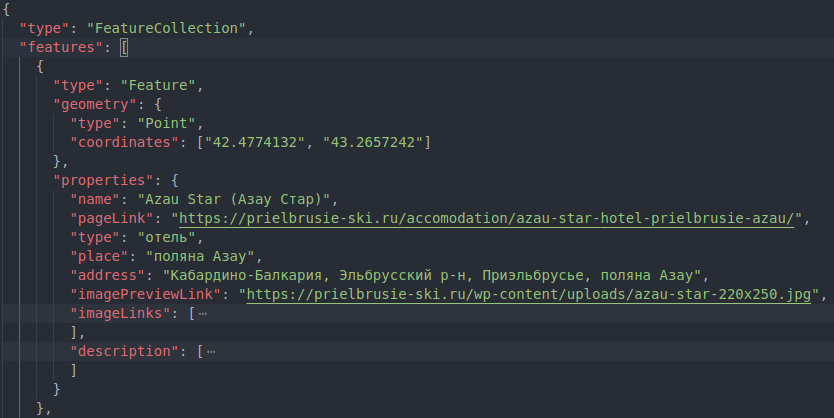
Для хранения больших постоянно изменяющихся данных обычно используют специальные базы данных, такие как PostgreSQL или MySQL. Для наших целей пока подойдет обычный JSON-файл.

**JSON** (англ. J*avaScript Object Notation*, обычно произносится как JAY-sən) — текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми.

Несмотря на происхождение от JavaScript (точнее, от подмножества языка стандарта ECMA-262 1999 года), формат считается независимым от языка и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON.

Обычно JSON-файл представляет из себя коллекцию объектов, которые записываются в формате «ключ» - «значение». Значением может быть как обычная строка, так и массив, или другой объект сколь угодно вложенности. Пары ключ-значение разделяются между собой запятой.

Для наших целей был подготовленн JSON-файл с данными об отелях Приэльбрусья. Конечно, здесь собраны не все данные, но на начальном этапе этого должно хватить.

 После того, как у мы имеем структурированные данные об отелях, мы можем их использовать в своих целях. Для начала прочитаем эти данные, чтобы в скрипте нашего сервера мы могли с ними работать. Для этого напишем запрос с помощью XMLHTTPRequest.

### Запросы на сервер

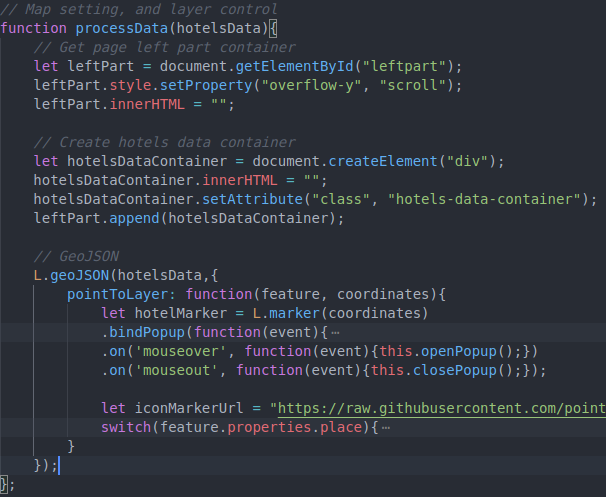
**XMLHttpRequest** (XMLHTTP, XHR) — API, доступный в скриптовых языках браузеров, таких как JavaScript. Использует запросы HTTP или HTTPS напрямую к веб-серверу и загружает данные ответа сервера напрямую в вызывающий скрипт. Информация может передаваться в любом текстовом формате, например, в XML, HTML или JSON. Позволяет осуществлять HTTP-запросы к серверу без перезагрузки страницы.

XHR использует запросы напрямую к веб-серверу, если бы у нас не было своего виртуального хоста Apache, у нас не получилось бы прочитать данные на нашем компьютере.



### **С**оздание маркеров отелей Leaflet

Теперь, когда данные получены и передаются в нужную нам функцию, мы можем уже добавлять их на карту. Для этого снова воспользуемся функционалом библиотеки Leaflet для облегчения нашей задачи.



Кроме работы с JSON-файлом данных, тут описаны еще несколько инструкций, относящихся к отображению и функционалу HTML-составляющей нашего сайта. Она понадобится нам позже.

А пока что стоит рассмотреть функцию, которая позволяет нам работать непосредственно с массивом JSON-данных. До этого не было сказано, но данные нашего JSON-файла представлены в формате **geoJSON**, который позволяет нам работать с различными объектами (точка, кривая, полигон, окружность). Для этого в geoJSON-файле прописывается определенная структура, описанная в спецификации. Коротко говоря, если внимательно посмотреть на фрагмент geoJSON-файла представленного выше, то мы можем понять его структуру. GeoJSON — это объект, имеющий два свойства. Первое указывает на то, что это коллекция элементов (feature collection). Второе является массивом из этих самых элементов. Каждый элемент в свою очередь имеет тип, свою геометрию (точка, кривая, полигон, ломанная), и набор координат для представления этого объекта на карте.

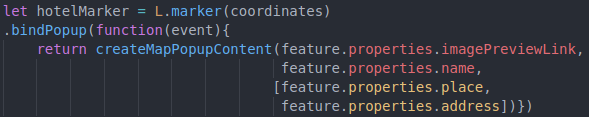
Возвращаясь к функции, стоит отметить, что она принимает на вход JSON-объект, и самостоятельно находит там коллекцию элементов, распознает их, и позволяет нам работать с ними как с отдельными объектами (feature), а так же находит их координаты и присваивает их переменной coordinates для более удобной работы с объектами. Внутри этой функции, вызывается другая функция для отрисовки маркеров на карте. А также вызываются метод bindPopup(), который привязывает к каждому методу всплывающее окна с информацией, которую мы заполним позже. Кроме этого настраивается условие открытия и сокрытия всплывающего окна с помощью метода «.on».

После того, как мы создали маркеры, отсортируем их по месту их расположения. В нашем случае разделим маркеры на 3 слоя: поляна Азау, поляна Чегет, отели в поселках. И для каждого маркера в слое поставим иконку со своим цветом. Для этих целей применяется блок выбора case (switch).

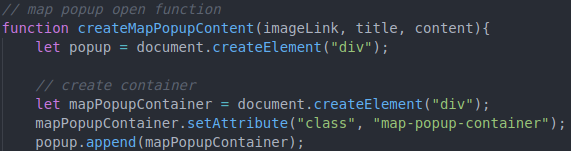


### Создание Popup

Всплывающее окно (popup) может быть добавлено к каждому маркеру, чтобы выводить информацию по каждому маркеру. Это окно может содержать простой текст, который описывает маркер, например, название места, координаты, адрес и т.д. Однако на этом функционал всплывающего окна не заканчивается. Метод «.bindPopup» принимает на вход строку текста, которая может содержать элементы HTML, при этом вложенная структура может быть довольно сложной и содержать элементы любой вложенности. Практически, мы можем вставить в это окно целую html-страницу. Происходит это потому, что само окно по строению является контейнером (типа div). Чтобы не писать большую сложную строку внутри метода, можно реализовать функцию, которая будет возвращать строку, html-структуру. Для размещения необходимой информации мы и воспользуемся функцией, которая будет возвращать методу фрагмент html-страницы, содержащий информацию об отеле для каждого отдельного маркера.



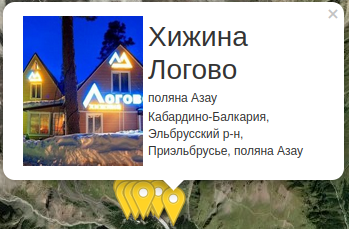
Внутрь метода мы помещаем функцию, которая будет вызывать функцию для создания контента нашего отеля, и возвращать результат как аргумент метода. Функция создания контента всплывающего окна принимает на вход 3 аргумента, ссылку на изображение отеля, имя отеля, массив данных об отеле (в нашем случае место расположения и адрес).



Функция состоит из методов создания контейнеров, элементов страницы. Этот код особо ничем не отличается, от примера на изображении выше, поэтому приводить весь код здесь не имеет смысла. Важно отметить, что результат функции представляется в виде html-структуры и возвращается обратно в метод bindPopup.

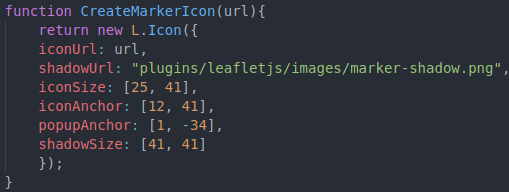


В результате выполнения функции мы получим следующее всплывающее окно, отображающее основную информацию об отеле. Оно будет всплывать при наведении на него курсора мыши.



### Изменение иконки маркера

В этом блоке для отелей в поселках настраивается маркер синего цвета, для отелей на поляне Азау – маркер золотого цвета, на поляне Чегет – зеленого. Учитывая особенности изменения иконки маркера в Leaflet (стандартные иконки библиотеки являются изображениями и не поддерживают настройку цвета программно), иконки маркеров подгружаются с Github из открытого репозитория. Затем используя метод «.setIcon» вызывается функция смены иконки маркера «CreateMarkerIcon», которая возвращает уже готовую иконку.

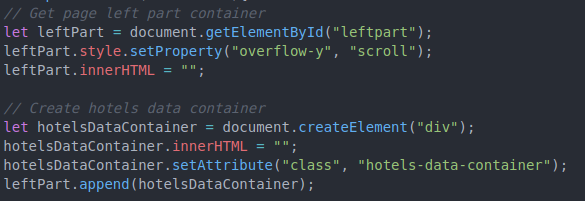


В данной функции присутствуют и другие атрибуты настройки иконки маркера, но они для нас сейчас не особо важны. К тому же они были взяты из того же репозитория с изображениями маркеров.

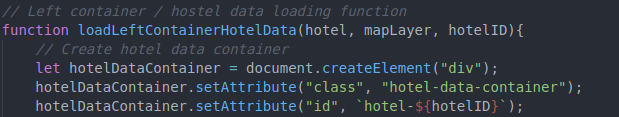
В этом же блоке каждый маркер добавляется на свой слой с использованием метода «.addTo». Теперь мы можем видеть наши маркеры на карте, так как маркеры созданы, добавлены в соответствующие слои, а эти слои мы уже добавили на карту.

### Создание левой части сервиса, список отелей

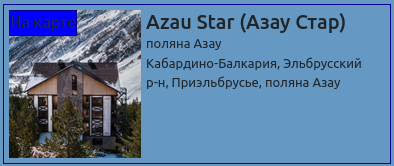
Теперь стоит упомянуть тот код, который был описан перед добавлением маркеров на карту.



В этом коде мы получаем элемент левой панели скелета нашего сайта, чтобы затем загрузить в него нужную информацию. Здесь же настраивается прокрутка контейнера и создается общий внутренний контейнер для данных «hotelsDataContainer», который вставляется в левую панель методом «.append». В следующих частях кода этот метод будет часто использоваться для динамической загрузки нужной информации на наш веб-сервис. Эта информация будет подгружаться на сайт с помощью отдельной функции, вызов которой вы уже могли видеть в блоке «switch». Эта функция «loadLeftContainerHotelData()», которая получает на вход 3 аргумента: данные об отеле, слой маркеров, содержащий маркер данного отеля, и id маркера.

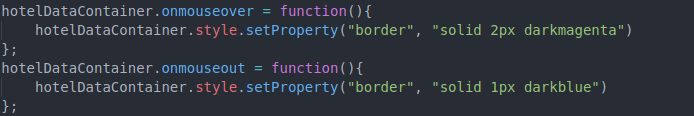


В этой функции большая часть кода отражает создание контейнеров и элементов, для вывода нужных данных об отелях. Таких как название, местоположение, адрес и фотография отеля, чтобы сформировать у пользователя первое представление о каждом отеле. Готовый вариант такой плитки для каждого отеля представлен ниже.



### Создание функций и хендлеров

Следует уделить внимание интерактивной составляющей, которая выражена в виде следующих функций:

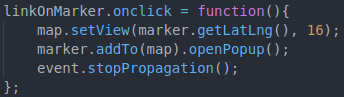


Эти функции-хендлеры (handler) позволяют пользователю лучше взаимодействовать с контентом веб-сервиса. Они служат для выделения плитки с информацией об отеле путем изменения рамки вокруг плитки при наведении мыши, чтобы пользователю было понятно, что с данным контентом можно взаимодействовать. Вторая функция меняет рамку на обычную при потере фокуса курсора над ней.

В этой же функции создания плиток отелей присутствуют и другие хендлеры, которые связывают список плиток с информацией об отелях и маркеры на карте.



Этот хендлер при клике по маркеру на карте пролистывает список отелей до нужной плитки с информацией и подсвечивает её на 1 секунду. Сделано это для удобства использования, чтобы пользователь не терялся в большом списке. У каждой плитки так же есть кнопка, которая позволяет выделить местоположения отеля на карте и центрирует положение карты на нём, показывая маркер, если он скрыт.



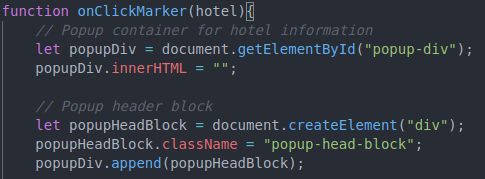
Простенькие функции визуализации, для облегчения взаимодействия с контентом. Но в последних двух есть загвоздка, чтобы они работали, нужно передать маркер или его id в вызываемую функцию. Такой подход не является самым надежным и предпочтительным, но на данном этапе подойдет и такая реализация.

### Окно информации

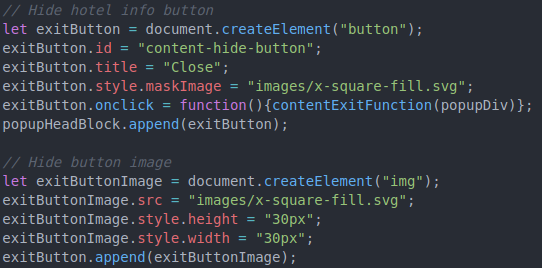
Для отображения более подробной информации об отелях было принято решение сделать отдельное окно, в которое будет подгружаться вся информация об отеле. Для того, чтобы вызвать данное окно с подробной информацией об отеле будет использоваться свойство «onclick» контейнера плитки отеля. Этому свойству присваивается функция, которая вызывает функцию-конструктор окна информации и принимает на вход объект «hotel» со всей доступной информацией.



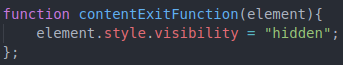
В самой функции создания окна информации мало чего интересного: большую часть кода занимает создание контейнеров, тегов, помещение их в родительские элементы, и, конечно, помещение информации об отеле в эти контейнеры. Фрагмент функции создания окна описания отеля представлен ниже.



Из более важных частей этой функции можно выделить создание кнопки закрытия окна контента, а также замена иконки этой кнопки. Фрагмент кода описания кнопки показан ниже.

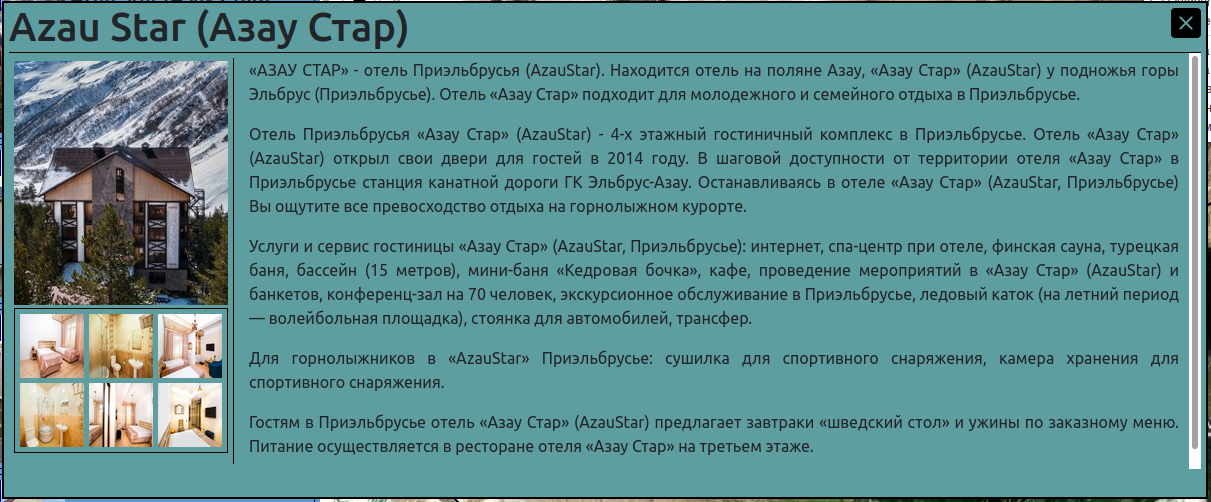


На фрагменте кода можно увидеть описание кнопки закрытия контента. В описании атрибутов тегов «button» и «img», есть атрибут «onclick» которому присваивается функция, которая вызывает функцию закрытия окна контента, принимающую в себя аргумент в виде html-тега.



Функция закрытия выражена в общем виде, что позволяет ей принимать на вход любой HTML-тег, имеющий свойство стиля оформления «visibility» (видимость). В данном случае мы передаем функции корневой элемент DOM-дерева этого окна — основной контейнер, который включает в себя весь контент.

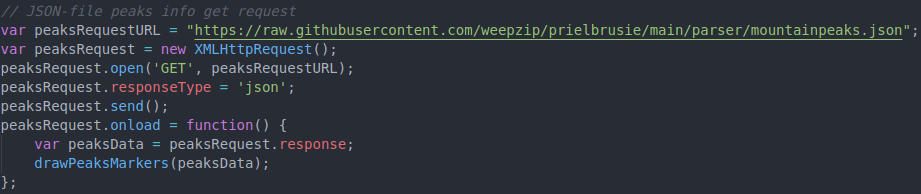
В конечном результате мы получаем окно контента, в которое загружаются данные для нужного нам отеля. Внешний вид этого окна показан на изображении ниже.



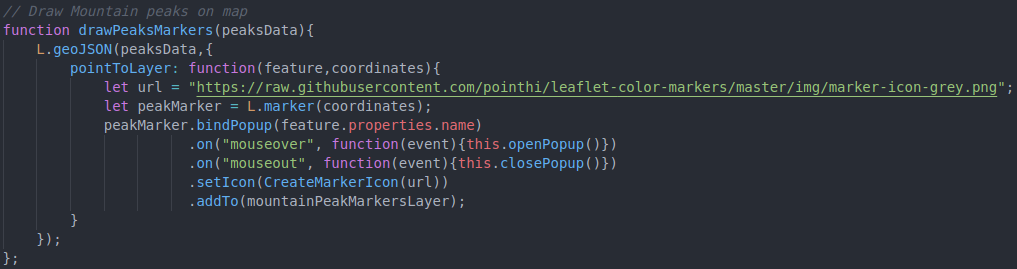
### Создание маркеров пиков гор

Поскольку алгоритм создания маркеров, всплывающих окон для них, изменения цвета, добавление на карту, и прочие возможности были описаны выше. Для следующих частей кода, описывающих создание маркеров и операции с ними, не будет подробного описания действий. Просто предоставим выкладку для демонстрации реализации возможностей веб-сервиса.

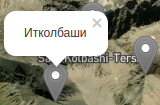
Для того, чтобы отобразить пики гор на карте нам нужно получить данные об этих пиках, поэтому был написан запрос на сервер, код которого представлен ниже.



После получения ответа от сервера, вызывается функция создания маркеров.

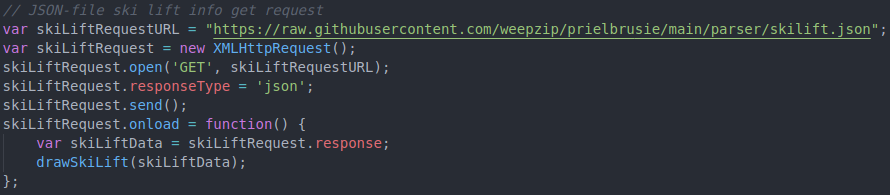


В этой функции данные читаются как geoJSON, и для каждого элемента коллекции создается маркер со всплывающим окном. Определяются методы, декларирующие открытие и закрытие окна по получению и потере фокуса курсора мыши, изменения иконки отображения маркера и добавления на слой маркеров пиков гор. Функцию создания иконки маркера мы уже рассмотрели выше, поэтому не будем ее рассматривать здесь. В результате мы получим следующее.



### Нанесение на карту информации о горнолыжных подъемниках

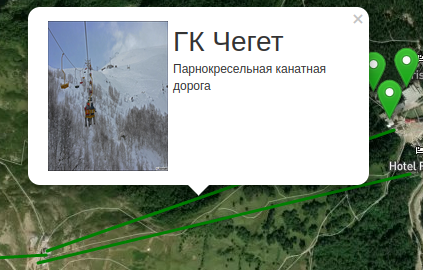
Для отрисовки маршрутов подъемников, и для вывода данных о каждом из подъемников, нужно получить данные. Делаем запрос HTTP-запрос.



После получения данных вызывается функция отрисовки данных на карту.



В этой функции для каждого подъемника выбирается цвет, исходя из его местоположения. Затем координаты представляются в виде «latlng», нужном для библиотеки Leaflet. После чего создаются полилинии, которые добавляются на слой с подъемниками. Для каждого подъемника добавляется всплывающее окно с информацией. Контент всплывающего окна запрашивается из функции «createMapPopupContent()», с которой мы уже работали. Результат этой функции мы можем видеть на изображении ниже.



# Заключение

Во время курсовой работы я познакомился с понятием веб-сервиса, спецификой его применения в современных условиях для реальных задач. Были приобретены знания работы с программным средством Visual Studio Code для разработки веб-сервиса. Я изучил средства верстки, программирования и визуального оформления веб-сервисов, в частности мною были получены знания языка гиперразметки — HTML, языка оформления содержимого CSS, языка программирования JavaScript. Так же для расширения возможностей веб-сервиса были использованы средства визуального представления картографической информации такие как: библиотека Leaflet для языка JavaScript, сервис MapBox, предоставляющий картографическую информацию из открытых источников и проприетарные карты, которыми можно бесплатно воспользоваться.

В результате работы получилось создать узконаправленный картографический веб-сервис, позволяющий пользователям получить нужную им информацию с помощью взаимодействия с сервисом. Были добавлены дополнительные элементы визуализации и добавлена интерактивная составляющая для лучшего взаимодействия с сервисом.

# Источники

1. Каргашин Павел Евгеньевич «Основы цифровой картографии» 2019г.
2. Молочко А.В., Хворостухин Д.П. «Геоинформационное картографирование в экономической и социальной географии» 2019г.
3. Молочко А.В., Федоров А.В. «Основы геоинформационного картографирования. Методические указания по выполнению лабораторных работ» 2015г.
4. Молочко А.В. «Интерактивный подход к созданию наглядных картографических способов изображения в рамках языковой концепции современной картографии» 2014г.
5. «Геоинформатика. Толковый словарь основных терминов» 1999г.
6. «Геоинформатика: в 2-х кн.: учебник для студентов вузов» под ред. В.С. Тикунова – 3 изд. 2010г.
7. Документация Github: <https://docs.github.com/en>
8. Документация HTML: <https://developer.mozilla.org/ru/docs/Web/HTML>
9. Документация CSS: <https://developer.mozilla.org/ru/docs/Web/CSS>
10. Документация HTTP: <https://developer.mozilla.org/ru/docs/Web/HTTP>
11. Документация JavaScript: <https://developer.mozilla.org/ru/docs/Web/JavaScript>
12. Документация API: <https://developer.mozilla.org/ru/docs/Web/API>
13. Документация DOM: <https://developer.mozilla.org/ru/docs/Web/API/Document_Object_Model>
14. Документация событий DOM: <https://developer.mozilla.org/ru/docs/Web/Events>
15. Документация по библиотеке Leaflet: <https://leafletjs.com/reference-1.7.1.html>
16. Документация Mapbox: <https://docs.mapbox.com/>
17. Документация Bootstrap: <https://getbootstrap.com/docs/5.1/getting-started/introduction/>