ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ФАКУЛЬТЕТ МАТЕМАТИКИ И ЕСТЕСТВЕННОНАУЧНОГО ОБРАЗОВАНИЯ
КАФЕДРА ИНФОРМАТИКИ, ЕСТЕСТВЕННОНАУЧНЫХ ДИСЦИПЛИН И МЕТОДИК ПРЕПОДАВАНИЯ

Создание интерактивной карты НИУ «БелГУ»

Выпускная квалификационная работа
обучающегося по направлению подготовки 44.03.05 Информатика и иностранный язык (английский)
очной формы обучения, группы 02041205
Федорова Антона Васильевича

Научный руководитель
к.ф.-м. н., доцент
Беляева И.Н.

БЕЛГОРОД 2017

Оглавление

ВВЕДЕНИЕ 4

1 ВВЕДЕНИЕ В ТЕХНОЛОГИИ СОЗДАНИЯ ИНТЕРАКТИВНЫХ КАРТ 6

1.1 Понятие интерактивной карты 6

1.2 Здание НИУ «БелГУ» 7

1.3 Обзор существующих технологий 8

1.4 Библиотека Leaflet 11

1.5 Библиотека OpenLayers 12

2 РАЗРАБОТКА ИНТЕРАКТИВНОЙ КАРТЫ НИУ «БЕЛГУ» 14

2.1 Инструменты и технологии разработки 14

2.2 Описание кода приложения 15

2.3 Создание векторного слоя интерактивной карты из растрового изображения плана здания 33

3 ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ КАРТЫ НИУ «БЕЛГУ» 38

3.1 Работа карты 38

3.2 Использование Material design icons 41

ЗАКЛЮЧЕНИЕ 44

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 45

ВВЕДЕНИЕ

Интернет-услуги в области геоданных постоянно расширяются и технологически совершенствуются, затрагивая все более глубокие пласты геоинформационной деятельности: производство и распространен ие цифровых геоданных, их стандартизацию и классификацию, создание ГИС (геоинформационные системы) с возможностями удаленного доступа для широкого круга пользователей посредством "открытых" сетей (то есть не требующих создания особых информационно-технологических инфраструктур), осуществление комплексных научно-исследовательских ГИС-проектов, подготовку профессиональных кадров в области ГИС. Можно говорить о формировании в сети Интернет мощного геоинформационного "пласта", который уже сейчас оказывает существенное влияние на развитие ГИС и геоинформационных наук в мире.

Веб-картография ознаменовала собой демократизацию доступа широких слоев компьютерных пользователей к географическим данным. С момента своего зарождения в середине 90-х гг. и до настоящего времени данные технологии прошли значительный путь развития. Сегодняшнего пользователя Интернета невозможно представить без использования картографических веб-сервисов (webmaps). В сети каждый день растет число сайтов, использующих технологию картографических сервисов (интерактивных карт) для распространения и популяризации данных и сопутствующей информации, а также предлагающих за плату разнообразные электронные карты вместе с наборами данных.

Данные технологии развивались на протежении длительного времени и на данном этапе существует возможность их использования не только для навигации по городам и в путешествиях, но и для создания на их основе различных приложений и сервисов для ориентации в помещениях.

Такая разработка является актуальной для различных зданий, учебных заведений, в том числе и НИУ «БелГУ». Университет является огромным комплексом из множества корпусов этажей и аудиторий среди которых легко потерять дорогу. Поэтому мы и задались целью создать интерактивную карту университета. Карту, которая выполняла бы функцию не только навигации по корпусам университета, но также являлась бы удобным графическим представлением информации об учебных аудиторяих, кабинетах, служебных помещениях и других объетах университета.

Еще одним неотъемлимым качеством, обуславливающим актуальность данной разработки, является то, что карта будет доступна со всех устройств, в которых есть доступ в интернет. Ее простота, легковесность и адаптивный дизайн позволят пользоватся картой именно в тот момент, когда она будет нужна, то есть на ходу и с мобильных устройств.

Цель выпускной квалификационной работы - создание интерактивной карты НИУ «БелГУ».

В соответствии с целью были сформированы следующие *задачи*:

- Выявить основные потребности потенциальных пользователей;
- Изучить основные средства создания интерактивных карт;
- Подготовить материалы на основе технических планов;
- Разработка интерактивной карты.

Объект выпускной квалификационной работы - здание НИУ «БелГУ».

Предмет выпускной квалификационной работы - интерактивная карта.

Научная новизна состоит в разработке наглядной и удобной системы навигации по корпусам НИУ «БелГУ» с возможностью использования мобильных устройств, а также в разработке способов представления информации в этой системе.

Выпускная квалификационная работа сосотоит из введения, трех глав, заключения и списка использованных источников.

1 ВВЕДЕНИЕ В ТЕХНОЛОГИИ СОЗДАНИЯ ИНТЕРАКТИВНЫХ КАРТ

1.1 Понятие интерактивной карты

Недавно на смену бумажным картам и атласам пришли электронные интерактивные, карты, позволяющие находить любые объекты моментально: достаточно ввести название объекта, щелкнуть на соответствующей кнопке, и нужный фрагмент карты окажется перед вами. Но дело не только в скорости: в электронных картах, в отличие от обычных, реализована интерактивность, то есть карты реагируют на действия пользователя. Онлайн проекты, объединяемые под термином «интерактивная карта», появились в сети недавно, но с каждым днем их количество все больше. Не все страны и города на интерактивных картах представлены одинаково подробно, также как не все сайты одинаково удобны. Тем не менее, эта информация очень полезна.

Изначально интерактивные карты отображали чисто картографическую информацию, которая описывала некоторую область земной поверхности, и выполняли в основном функции справочника. Теперь электронные карты стоит рассматривать не только как картографическое произведение, но и как интерфейс доступа к данным, связанным с определенной территорией.

Электронные карты, представленные в Интернете и включающие элементы интерактивности, называют электронными интерактивными картами [10].

1.2 Здание НИУ «БелГУ»

В ходе разработки будут использованы планы корпусов НИУ «БелГУ», на основании которых будет создана карта (см. Рисунок 1). Планируется их обработка, а также обновление в соответствии с реальными зданиями и их изменениями с момента составления планов.

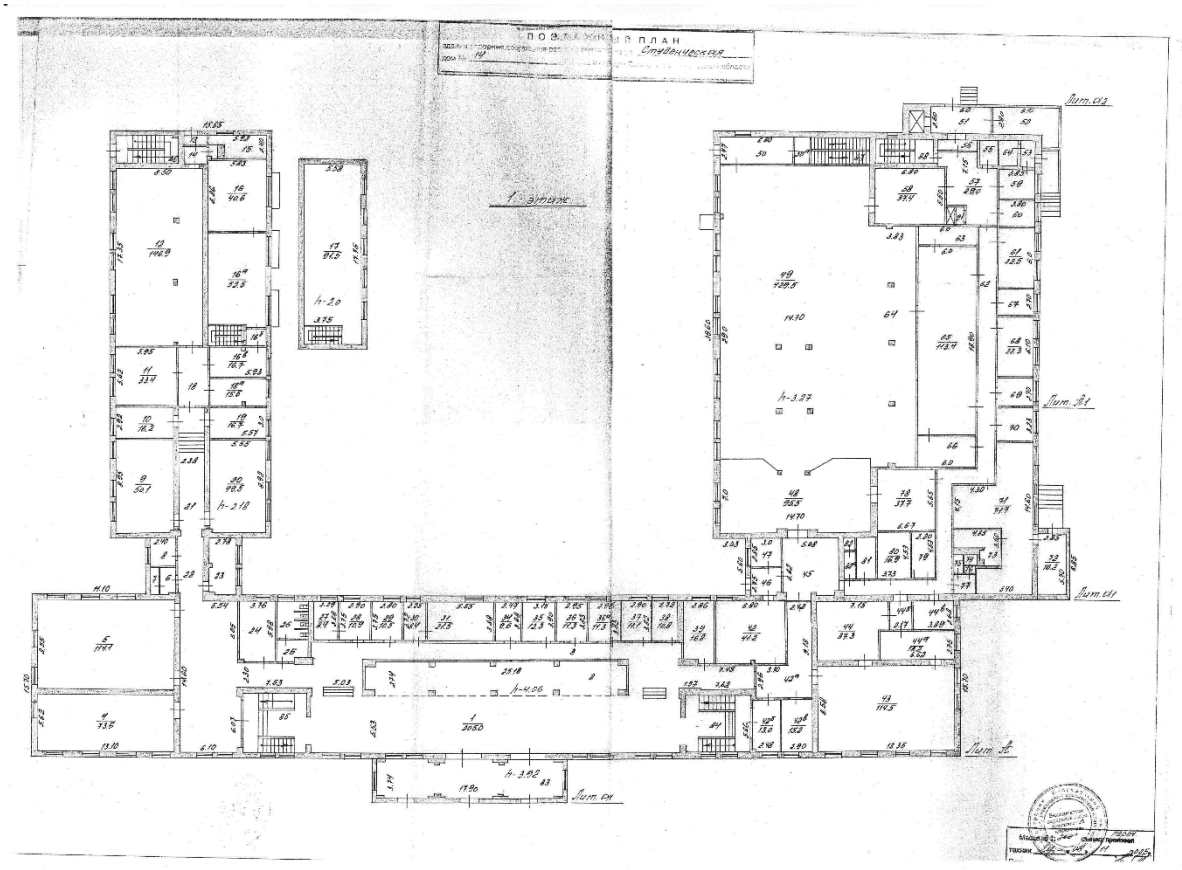

Рисунок 1 – Часть планов корпуса НИУ «БелГУ»

Также проведена работа по нанесению на планы расположения кабинетов и других помещений. Способ представления планов и относящийся к помещениям информации в виде используемых приложением данных описан во второй главе.

1.3 Обзор существующих технологий

Существует два типа современных веб карт: картографические веб-сервисы и коллективные веб карты.

Среди множества таких сервисов можно выделить некоторых лидеров в этой области: YandexMaps, Google Maps, Bing Maps

Эти сервисы выделяются широким спектром предоставляемых возможностей, хорошим покрытием и удобством для пользователей. К числу их сильных сторон можно отнести развитые навигацию по карте и масштабирование (zooming), наличие специализированных информационных ресурсов (карты улиц крупных городов, транспортных потоков в режиме реального времени и прочее).

Все ведущие картографические сервисы поддерживают последние версии популярных браузеров (IE, Mozilla Firefox, Safari, Opera, Chrome). При этом Google Maps доступен для пользователей наибольшего числа версий браузеров, включая уже вышедшие из употребления - благодаря своей давней истории и изначально кросс-браузерному дизайну, заложенному в его клиентское программное обеспечение.

Если говорить об источниках картографических данных, то общей чертой этих сервисов является сотрудничество со специализированными организациями, у которых они закупают данные.

Безусловно, сервисы различаются функциональностью. Например, картографическая служба Microsoft предоставляет навигацию по трехмерным картам улиц крупнейших городов США и некоторых других стран [17], в то время как Google дает уникальный инструмент для разработчиков, желающих разместить ту или иную прикладную информацию на картах, - GoogleMaps API.

Также эти сервисы отличаются по степени покрытия различных районов земного шара и актуальностью картографических данных. Например, карты Яндекса для территорий стран СНГ отвечают последнему требованию. Актуальность карт в "GoogleMaps" и "BingMaps" для разных регионов - 1-3 года.

Каждый сервис имеет свои преимущества и недостатки. В зависимости от поставленной задачи можно применять разные веб-сервисы.

Современные картографические веб-сервисы, несомненно, предоставляют массу возможностей, как специалистам, так и пользователям интернета. Однако у них имеется целый ряд недочетов, связанных с применением коммерческих данных (недостаточное или отсутствующее покрытие в определенных регионах земного шара, устаревшая картографическая информация и так далее).

Именно это послужило толчком для развития альтернативных картографических веб-решений - коллективных веб-карт. Их отличительной чертой является возможность для пользователей самим создавать и обновлять данные на картах. В результате получается достаточно достоверная и совершенно бесплатная карта региона и мира в целом.

Среди наиболее ярких представителей следует выделить проект "OpenStreetMaps".

Он был создан в 2004 г. Стивом Костом после того, как тот разочаровался в качестве и доступности электронных карт Великобритании. С тех пор проект развился и на сегодняшний день превратился в достаточно массовое движение GPS-картографирования (сейчас он насчитывает свыше 50 тысяч зарегистрированных пользователей, из которых около 5 тысяч - активные картографы) [15].

Одна из важных особенностей OSM - развитие набора инструментов для автоматического ввода данных, полученных пользователями с помощью

GPS-картографирования, в централизованное хранилище. Именно благодаря этому в системе появляются и обновляются карты различных участков земного шара.

Популяризации данных OSM способствовало создание основателями OSM компании "Клаудмейд", которая разрабатывает линейку коммерческих продуктов, использующих данные OSM (а именно, картографические Web API и MobileAPI [18].

Другим примером успешной коллективной веб-карты является проект WikiMapia, созданный россиянами Александром Корякиным и Евгением Савельевым в 2006 г. По сути, это надстройка над GoogleMaps, в которой применяется wiki-подобный интерфейс для ввода пользовательских данных [16].

Также следует упомянуть о роли opensource проектов и технологий в развитии веб картографии. Это такие проекты, которые также как и OpenStreetMaps развиваются путем коллективных усилий множества людей. Этот вклад не ограничивается только предоставлением свежих картографических данных определенной местности. Люди напрямую участвуют в развитии технологий путем изучения исходного кода приложений, который, к слову, находится в свободном доступе, и предлагая или внося в него изменения. Такие проекты позволяют создавать свои картографические проекты без ограничений уже упомянутых ранее в тексте сервисов. Если поставленная задача требует обработки уже существующих карт или вам требуется создать свою карту с нуля, то следует обратить внимание на эти технологии.

К ним относятся: OpenLayers, Leaflet и подобные им. Это открытые JavaScript библиотеки, которые позволяют работать с различными картографическими данными.

Так как нашей задачей является создание карты помещения, что предполагает работу с уникальными данными, а также создание интерфейса для работы именно с картой помещения, то нам следует подробнее остановится именно на этих продуктах.

1.4 Библиотека 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. Автор библиотеки – киевлянин Владимир Агафонкин на момент выхода первой версии (2011) был сотрудником CloudMade, с 2013 перешёл в MapBox [13].

Leaflet позволяет разработчику, не знакомому с ГИС, легко отображать растровые карты, состоящие из маленьких фрагментов – тайлов, с, возможно, дополнительными слоями, накладываемыми поверх основного. Слои могут быть интерактивными, например, отображать подсказку при клике по маркеру.

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.

Также Leaflet сравнивают с проприетарной закрытой Google Maps API (впервые вышла в 2005) и Bing Maps API – они обе используют значительную часть на стороне сервера для предоставления таких услуг, как геокодировние, прокладка маршрутов, поиск и интеграция с дополнительным ПО, таким как Google Earth. Google Maps API дают скорость и простоту вместе с гибкостью, однако дают доступ только к сервисам Google Maps. Впрочем, DataLayer – часть Google's API – позволяет использовать внешние данные.

1.5 Библиотека OpenLayers

OpenLayers – библиотека с открытым исходным кодом, написанная на JavaScript, предназначенная для создания карт на основе программного интерфейса (API), подобного GoogleMap API или Bing Maps API. Библиотека включает в себя компоненты из JavaScript-библиотек Rico и Prototype JavaScript Framework [14].

OpenLayers позволяет очень быстро и легко создать web-интерфейс для отображения картографических материалов, представленных в различных форматах и расположенных на различных серверах. Благодаря OpenLayers разработчик имеет возможность создать, к примеру, собственную карту, включающую слои, предоставляемые WMS (и WFS) серверами, такими как Mapserver, ArcIMS или Geoserver, и данными картографических сервисов Google. Библиотека является разработкой с открытым исходным кодом и разрабатывается при спонсорской поддержке проекта MetaCarta, который использует OpenLayers в своих разработках. Тем не менее, OpenLayers является независимым свободно распространяемым продуктом.

Эти платформы часто сравнивают. Обе являются открытым ПО, обе — клиентские библиотека на JavaScript. Leaflet заметно компактнее, содержит

около 7 тысяч строк против 230 тысяч у OpenLayers по состоянию на 2015. Leaflet занимает меньше места, чем OpenLayers (около 123 кБ против 423), не в последнюю очередь благодаря модульной структуре. Код новее и использует свежие возможности JavaScript, HTML5 и CSS3.

2 РАЗРАБОТКА ИНТЕРАКТИВНОЙ КАРТЫ НИУ «БЕЛГУ»

2.1 Инструменты и технологии разработки

Напомним, что целью нашей работы является создание интерактивной карты НИУ «БелГУ». Одним из основных требований к нашей карте является ее интерактивность, то есть возможность взаимодействия пользователя с картой. Это то, что будет отличать нашу карту от обычного изображения карты или плана помещения.

Следующим требованием является ее кроссплатформенность и возможность использования без дополнительных установок или подготовки устройства.

Также нам важна легковесность и быстродействие карты, так как из предыдущих пунктов ясно, что нами поставлена задача разработать такое приложение, которое должно работать на лету и не занимать много времени пользователя при взаимодействии с ним.

Обозначенным требованиям отвечает однозначно только один способ реализации нашей карты. Это веб приложение.

Реализация карты в виде веб приложения позволит нам обеспечить не только требование интерактивности, легковесности, кроссплатформенности и так далее, но и обозначенное выше требование облегчить доступ пользователя к карте, то есть обойтись без установки, как это было бы с мобильным приложением, которое к тому же жестко привязано к платформе разработки.

Следующим шагом становится выбор пути реализации нашей карты в виде веб приложения. Наиболее правильным решением будет воспользоваться одной из упомянутых ранее библиотек. Большинство таких

библиотек, как говорилось в предыдущей главе, являются бесплатными и с открытым исходным кодом.

Нашим выбором будет JavaScript библиотека Leaflet. В первой главе среди прочих уже упоминались ее достоинства. Здесь же отметим только то, почему мы предпочли ее аналогичной библиотеке OpenLayers [14].

Одним из преимуществ является то, что Leaflet гораздо более простая и легкая библиотека (около 123 кБ против 423). В самой идее Leaflet заложена простота. Она содержит только самые необходимые функции. Остальное легко подключается плагинами, коих на данный момент великое множество и при должном желании и навыке каждый может написать плагин и опубликовать его для того, чтобы облегчить другим жизнь и развить библиотеку [13].

Для представления данных мы выбрали формат данных GeoJSON который позволяет представить наше здание как набор геометрических фигур, которые будут описаны с помощью географических координат и иметь различные свойства для дальнейшей обработки их в нашем приложении. Такой формат позволит также легко хранить свойства аудиторий для дальнейшего развития системы. Он поддерживается множеством систем и сервисов, а также легко хранится в базе данных [20].

Перейдем к конкретной реализации нашей системы.

2.2 Описание кода приложения

Приложение основано на библиотеке Leaflet. Это значит, что мы будем подключать набор инструментов через ссылку. В этом разделе мы опишем структуру html документа и код в котором будем использовать инструменты Leaflet, которые будут описаны в следующем разделе.

Сама карта в структуре DOM будет располагаться там, где мы поставим тег

Где mapid - это id нашей карты, с помощью которого мы будем в дальнейшем манипулировать и настраивать карту.

Минимальным необходимым условием для отображения карты будет указание высоты блока с помощью стилей:

```
#mapid { height: 180px; }
```

И следующее объявление в скрипте:

```
var mymap = L.map('mapid').setView([51.505, -0.09],
13);
```

В котором имеются следующие команды

L.map(<String> id, <Map options>options?) Возвращает экземпляр объекта карты с учетом DOM-идентификатора элемента .

setView(<LatLng> center, <Number>zoom, <Zoom/pan options>options?) Устанавливает представление карты (географический центр и масштаб) с заданными параметрами анимации.

Также существует функция добавления слоя тайлов (фрагментов карты):

```
L.tileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/
{y}.png?{foo}', {foo: 'bar'}).addTo(mymap);
```

Вот описание этой команды:

L.tilelayer(<String> urlTemplate,<TileLayer options> options?)

Создает объект слоя тайлов с учетом шаблона URL и, опционально, с объектом опций. В этой команде мы указываем URL, где хранятся тайлы и опции их отображения. В нашей карте мы не будем использовать этот функционал, так как наша карта будет иметь несколько иное представление, которое мы опишем в следующем разделе.

Всю работу по отображению карты выполняет библиотека. Это позволяет сосредоточится при разработке на том, как использовать весь доступный инструментарий именно для решения определенной задачи, а не тратить время на разработку уже готовых решений.

Разберем подробнее наш скрипт, который содержится в теге body вместе с упомянутым ранее тегом div с id нашей карты.

```
<script type='text/javascript'>

// Прокрутка страницы при загрузке для корректного
отображения в браузере Chrome на Android.
window.addEventListener("load",function() {
    // Set a timeout...
    setTimeout(function(){
        // Hide the address bar!
        window.scrollTo(0, 1);
    }, 0);
});

// Объявление объекта карты.
var mymap = L.map('mapid', {
```

```
    maxZoom: 21,
    minZoom: 18,
    layers: level1,
    renderer: L.svg({ padding: 0.5 })
}).setView([50.62147,36.57862], 20);

// Описание стилей корридора.
function corridorStyle(feature) {
   return {
        color: "#247BA0",
        weight: 1,
        fillColor: "#f0f9e8",
        fillOpacity: 1
    }
};

// Описание стилей комнат.
function roomStyle(feature) {
   return {
       fillColor: getColor(feature.properties.type),
       weight: 1,
       opacity: 1,
       color: '#247BA0',
       fillOpacity: 1
   };
}

// Получение цвета заполнения фона в зависимости от
роли помещения.
```

```
function getColor(d) {
       switch (d) {
           case 'inter':   return "#bae4bc";
           case 'stairs':   return "#0868ac";
             case 'tech':   return "#43a2ca";
             case 'stud':   return "#7bccc4";
       }
}

// Объявление конкретных объектов на карте и присвоение
им данных и стилей.
var level1_corridor = L.geoJSON(block1_1, {style:
corridorStyle});
var level2_corridor = L.geoJSON(block1_2, {style:
corridorStyle});
var level3_corridor = L.geoJSON(block1_3, {style:
corridorStyle});
var level4_corridor = L.geoJSON(block1_4, {style:
corridorStyle});

// Создание групп маркеров.
var level1_markers = L.layerGroup([]);
var level2_markers = L.layerGroup([]);
var level3_markers = L.layerGroup([]);
var level4_markers = L.layerGroup([]);

// Функция создания иконки мест интереса или получения
номера аудитории для отображения.
function makeIcon(properties) {
```

```
    if (isNaN(properties)) return '<i class="mdi ' +
properties + ' mdi-24px"></i>'
    else return properties;
}

// Функция создания маркеров.
function onEachFeatureMakingMarkers(feature, layer) {
    var label = L.marker(layer.getBounds().getCenter(),
{
        icon: L.divIcon({
            className: 'my-div-icon',
            html: makeIcon(feature.properties.name),
            iconSize: [50, 30],
            popupAnchor: [0, -10]
            //iconAnchor: [0,0]
makeIcon(feature.properties.name)
        })
    });
    if (feature.properties && feature.properties.popup)
{
        label.bindPopup(feature.properties.popup)
        layer.bindPopup(feature.properties.popup)
    }
    window["level" + feature.properties.level +
"_markers"].addLayer(label);
};

// Описание объектов комнат с присвоением им данных и
стилей
```

```
var level1_rooms = L.geoJSON(rooms1_1, {
    style: roomStyle,
    onEachFeature: onEachFeatureMakingMarkers
});

var level2_rooms = L.geoJSON(rooms1_2, {
    style: roomStyle,
    onEachFeature: onEachFeatureMakingMarkers
});

var level3_rooms = L.geoJSON(rooms1_3, {
    style: roomStyle,
    onEachFeature: onEachFeatureMakingMarkers
});

var level4_rooms = L.geoJSON(rooms1_4, {
    style: roomStyle,
    onEachFeature: onEachFeatureMakingMarkers
});

// Создание слоев комнат.
var level1 = L.layerGroup([level1_corridor,
level1_rooms, level1_markers]);
var level2 = L.layerGroup([level2_corridor,
level2_rooms, level2_markers]);
var level3 = L.layerGroup([level3_corridor,
level3_rooms, level3_markers]);
var level4 = L.layerGroup([level4_corridor,
level4_rooms, level4_markers]);
```

```
// Объявление этажей.
var baseMaps = {
    "4": level4,
    "3": level3,
    "2": level2,
    "1": level1
}

// Добавление интерфейса этажей.
L.control.layers(baseMaps, null, {
    //position: 'bottomright',
    collapsed: false
}).addTo(mymap);

// Функция для корректного отображения маркеров на
разных уровнях увеличения.
function removeMarkersOnZoomend(map, level,
level_markers){

    function removeMarkers(){
    var currentZoom = map.getZoom();
    if (currentZoom < 20 && map.hasLayer(level_markers)
&& map.hasLayer(level)) map.removeLayer(level_markers);
    if (currentZoom >= 20 &&
!map.hasLayer(level_markers) && map.hasLayer(level))
map.addLayer(level_markers);
}
    map.on('zoomend', removeMarkers);
```

```
    map.on('baselayerchange', removeMarkers);

}

// Применение функции отображения маркеров.
removeMarkersOnZoomend(mymap, level1, level1_markers);
removeMarkersOnZoomend(mymap, level2, level2_markers);
removeMarkersOnZoomend(mymap, level3, level3_markers);
removeMarkersOnZoomend(mymap, level4, level4_markers);

</script>
```

Первая функция:

```
window.addEventListener("load",function() {
    // Set a timeout...
    setTimeout(function(){
        // Hide the address bar!
        window.scrollTo(0, 1);
    }, 0);
});
```

Она нужна для корректного отображения нашей карты в браузере Chrome на операционной системе Android и подобных ему. Она позволяет скрыть панель с адресной строкой при загрузке карты для ее полноэкранного отображения. Дело в том, что браузер сам скрывает эту панель при прокрутке страницы, но так как наша карта является блоком, который растянут на всю страницу, то как таковой прокрутки страницы не происходит и поэтому приходится слегка прокручивать страницу при загрузке.

Следом мы объявляем объект карты

```
var mymap = L.map('mapid', {
    maxZoom: 21,
    minZoom: 18,
    layers: level1,
    renderer: L.svg({ padding: 0.5 })
}).setView([50.62147,36.57862], 20);
```

к которому в дальнейшем будем добавлять различные модули и объекты с помощью функции .addTo(mymap).

Здесь, в отличии от описанного ранее примера, мы используем опции:

maxZoom и minZoom - Используются для ограничения максимальных значений увеличения и уменьшения карты.

layers - показывает какие слои следует добавить на карту при загрузке.

renderer - метод отображения векторных слоев.

Функция setView указывает точку по которой карта будет отцентрирована при загрузке страницы, а также уровень увеличения.

Затем следует несколько функций объявления стилей для различных объектов карты:

Первая объявляет стили коридора:

```
function corridorStyle(feature) {
   return {
        color: "#247BA0",
        weight: 1,
        fillColor: "#f0f9e8",
        fillOpacity: 1
```

```
    }
}
```

Она возвращает значения стилей:

color - цвет обводки.

weight - толщина обводки.

fillColor - цвет заливки.

fillOpacity - степень прозрачности заливки.

Следом идет функция стилей комнат:

```
function roomStyle(feature) {
   return {
       fillColor: getColor(feature.properties.type),
       weight: 1,
       opacity: 1,
       color: '#247BA0',
       fillOpacity: 1
   };
}
```

Все параметры те же, что и у предыдущей, за исключением того, что цвет заливки у нее определяется с помощью следующей функции:

```
function getColor(d) {
    switch (d) {
        case 'inter':   return "#bae4bc";
        case 'stairs':   return "#0868ac";
        case 'tech':   return "#43a2ca";
        case 'stud':   return "#7bccc4";
```

```
    }
}
```

Эта функция считывает параметр d и в зависимости от содержащегося в нем значения роли комнаты возвращает определенный цвет.

При вызове getColor() в функции roomStyle() параметр d принимает значение feature.properties.type, которое будет получено из загружаемых данных об объекте.

Следом идет объявление объектов коридора:

```
var level1_corridor = L.geoJSON(block1_1, {style:
corridorStyle});
```

Где используется следующая функция Leaflet:

```
L.geoJSON(<Object>geojson?, <GeoJSON options>options?)
```

Создает слой GeoJSON. Принимает данные GeoJSON и опции отображения. Наши данные находятся в переменной block1_1, которая находится в файле, который в свою очередь подключается в разделе head с помощью ссылки:

```
<script src="geojson/block1_1.js"
type="text/javascript"></script>
```

Таким образом подключаются все данные об объектах, отображаемых на карте. В этих файлах мы присваиваем упомянутым ранее переменным, используемым в нашем коде для обозначения этих объектов, данные в формате GeoJSON:

```
var block1_1 = {
 "type": "FeatureCollection",
 "crs": {
   "type": "name",
   "properties": {
     "name": "urn:ogc:def:crs:OGC:1.3:CRS84"
   }
 },
 "features": [
   {
     "type": "Feature",
     "properties": {
       "id": null,
       "type": "building_outline",
        "level": 1
     },
     "geometry": {
       "type": "Polygon",
       "coordinates": [
         [
           [
             36.578081159846164,
             50.621928209162206
           ],
           ...// Много координат точек.
           [
             36.578081159846164,
             50.621928209162206
```

```
            ]
          ]
        ]
      }
    }
 ]
};
```

Также мы присваиваем опции style функцию определения стиля. В данном случае функцию для коридора.

Следующим шагом идет функция создания маркеров. Маркерами будут обозначаться номера комнат, а также они будут принимать вид иконок для различных мест интереса.

Перед самой функцией мы создаем группы слоев для каждого этажа, в которых будут содержаться созданные с помощью нашей функции маркеры:

```
var level1_markers = L.layerGroup([]);
```

Сама функция имеет вид:

```
function onEachFeatureMakingMarkers(feature, layer) {
    var label = L.marker(layer.getBounds().getCenter(),
{
        icon: L.divIcon({
            className: 'my-div-icon',
            html: makeIcon(feature.properties.name),
            iconSize: [50, 30],
            popupAnchor: [0, -10]
```

```
            //iconAnchor: [0,0]
makeIcon(feature.properties.name)
        })
    });
    if (feature.properties && feature.properties.popup)
{
        label.bindPopup(feature.properties.popup)
        layer.bindPopup(feature.properties.popup)
    }
    window["level" + feature.properties.level +
"_markers"].addLayer(label);
};
```

В ней используется стандартная функция создания маркеров Leaflet:

```
L.marker(<LatLng> latlng,<Marker options> options?)
```

Которая, согласно документации Leaflet, создает объект Marker с учетом географической точки и необязательного объекта опций. Первым параметром принимает географическую точку. В нашем случае мы используем функции layer.getBounds().getCenter(), которые позволяют сначала с помощью getBounds() получить границы объекта, а следом используя их с помощью getCenter() получить его центр. Таким образом мы всегда выставляем маркер по центру комнаты.

В опции icon мы указываем нужную нам иконку. В нашем случае это специальный тип иконок, который предусмотрен библиотекой Leaflet и обозначен:

```
L.divIcon({
```

```
        className: 'my-div-icon',
        html: makeIcon(feature.properties.name),
        iconSize: [50, 30],
        popupAnchor: [0, -10]
})
```

Имеет следующие опции:

className - имя класса, которое будет использоваться при задании стилей иконкам.

html - в данной опции указывается html код, который будет представлять нашу иконку. Собственно в этом и особенность этих иконок. Они представляют собой div блок вместо изображения. Это открывает простор для модернизации и выбора различного представления иконок, к примеру, как в нашем случае это просто текст с номером кабинета или специальный символ, обозначающий места интереса. Про эти символы мы поговорим в одном из следующих разделов.

Также в нашем случае мы используем в этой опции не просто html код, а функцию, которая поможет нам собрать этот код для правильного отображения иконок:

```
function makeIcon(properties) {
    if (isNaN(properties)) return '<i class="mdi ' +
properties + ' mdi-24px"></i>'
    else return properties;
```

Следом мы создаем объекты комнат для разных этажей:

```
var level1_rooms = L.geoJSON(rooms1_1, {
    style: roomStyle,
```

```
    onEachFeature: onEachFeatureMakingMarkers
});
```

Здесь используется уже упоминаемая ранее функция L.geoJSON. Все параметры схожи, за исключением опции onEachFeature. Она вызывается для каждого созданного с помощью функции объекта. Она предназначена для присвоения различных свойств множеству объектов, а также для создания маркеров, для чего мы ее и используем в этом контексте. С ее помощью мы вызываем упоминаемую ранее функцию onEachFeatureMakingMarkers.

Дальнейшим шагом будет создание самого объекта этажа со всеми его частями: коридором, комнатами и маркерами. Это можно наблюдать в следующей части кода:

```
var level1 = L.layerGroup([level1_corridor,
level1_rooms, level1_markers]);
```

Затем мы группируем созданные этажи для дальнейшего их использования:

```
var baseMaps = {
    "4": level4,
    "3": level3,
    "2": level2,
    "1": level1
}
```

Теперь мы используем список этажей для создания переключателя этажей:

```
L.control.layers(baseMaps, null, {
    //position: 'bottomright',
    collapsed: false
}).addTo(mymap);
```

Это простой интерфейс, предоставляемый библиотекой. Список слоев в первом параметре будет переключатся через radio button. Второй список через checkbox. Нам нужно однозначно выбирать один из этажей, поэтому мы используем первый параметр, а во втором ставим null. Опция collapsed обозначает, будет ли сворачиваться интерфейс. Мы ставим значение false, так как хотим, чтобы он был всегда на виду. При большом количестве этажей может быть полезным свернуть его.

Следующей функцией мы организуем адаптивное отображение маркеров на нашей карте. При низких уровнях увеличения некоторые маркеры не будут отображаться. Это позволит избежать слишком нагруженного их отображения и пресечения друг с другом.

```
function removeMarkersOnZoomend(map, level,
level_markers){
    function removeMarkers(){
        var currentZoom = map.getZoom();
        if (currentZoom < 20 &&
map.hasLayer(level_markers) && map.hasLayer(level))
map.removeLayer(level_markers);
        if (currentZoom >= 20 &&
!map.hasLayer(level_markers) && map.hasLayer(level))
map.addLayer(level_markers);
    }
```

```
    map.on('zoomend', removeMarkers);
    map.on('baselayerchange', removeMarkers);
}
```

Внутри нее организован перехват событий, происходящих с картой. Мы обрабатываем события смены уровня увеличения и смену этажа. При возникновении этих событий будет вызвана другая функция, которая описана внутри - removeMarkers(). Она проверяет текущий уровень увеличения, после указанных событий и в зависимости от него либо убирает маркеры, либо добавляет их на место.

2.3 Создание векторного слоя интерактивной карты из растрового изображения плана здания

В данном разделе описывается способ создания векторного слоя карты или плана, используя различные растровые источники картографических данных. В данном конкретном примере будет создаваться векторный слой в формате GeoJSON. В дальнейшем этот формат будет использован нами в качестве основного носителя картографических данных. Изначальным источником картографических данных будет служить отсканированный растровый поэтажный план, который мы и будем переводить в нужный нам формат.

Разберем понятия и форматы, которые мы будем использовать дальше.

Открытый формат GeoJSON, является модификацией другого текстового формата для хранения данных JSON, который в свою очередь основан на JavaScript. GeoJSON используется для хранения географических данных в виде различных примитивов, таких как точки (адреса и местоположения), линии (улицы, шоссе, границы), полигоны (страны, штаты,

участки земли). Формат довольно широко распространен в среде геоинформационных систем. Его поддерживают такие картографические программы, как OpenLayers, Leaflet, MapServer, Geoforge software, GeoServer, GeoDjango, GDAL, Safe Software FME, и CartoDB. Кроме этого, можно использовать GeoJSON с PostGIS и Mapnik, оба работают с форматом с помощью библиотеки GDAL OGR. Онлайн-сервисы Bing Maps, Yahoo! и Google также поддерживают GeoJSON в своих API. Интерфейс Javascript API v3 карт Google Maps напрямую поддерживают интеграцию слоёв данных GeoJSON по с 19 марта 2014 года. GitHub тоже поддерживает GeoJSON и GeoJSON-экспорт Potrace [20].

Также разберем подробнее программное обеспечение, которое мы используем для наших целей. Нам понадобится программа QGIS с установленными плагинами Georeferencer и OpenLayers. В нашей работе мы используем QGIS, потому что он бесплатен и удобен в использовании. Также возможно использование любой другой ГИС, к примеру, ArcMap. Главные требования к такому программному обеспечению — это поддержка геопривязки, создание полигонов и экспорт в формат GeoJSON [21].

Перейдем к инструкции.

Следует установить и запустить QGIS, затем убедится в том, что установлен плагин Georeferencer (меню Raster> Georeferencer). В версии под Windows он идет в комплекте. На OS X и Linux нам может потребоваться установить его через менеджер плагинов (Plugins> Manage and Install Plugins> Находим “Georeferenced”> Выберем “Georeferencer GDAL”> Нажимаем “Install Plugin”).

Мы также будем использовать плагин OpenLayers. Он позволит нам добавить картографические данные из различных источников на рабочую область программы. Это сделает определенные шаги на много легче, так как мы сможем сопоставлять наши данные с картой.

Перейдем к геопривязке нашего растрового плана.

К сожалению изображения, какими бы детальными и точными они не были, не содержат никаких географических данных, и они никак не привязаны к реальному миру. Тут нам и понадобится геопривязка.

Геопривязка в основном обозначает, что вы связываете точку P на изображении с координатой Q на карте. Это позволяет нам отобразить наше изображение на карте в корректном месте и с правильной ориентацией в пространстве.

Откроем QGIS и добавим слой карты через плагин OpenLayers. Находим наше здание на карте. Открываем Georeferencer (Raster> Georeferencer). Нажимаем кнопку Add Raster. Находим наш план и установим систему координат (CRS) WGS84 / Pseudo Mercator (EPSG: 3857).

Теперь нужно выбирать точки на вашем плане и ассоциировать их с географическими точками. Для этого в окне Georeferencer на панели инструментов мы нажимаем Add Point и затем выбираем первую точку на нашем плане. Лучше всего брать углы здания.

В открывшимся окне следует нажать на From map canvas. Это позволит нам выбрать точку на открытой ранее карте с нашим зданием. На карте выберите совпадающее с выбранной точкой место нашего здания. Следует обозначить таким образом хотя бы три угла нашего здания.

Затем открываем меню Transformation Settings и выберем там имя для файла, в который будет загружена информация о геопривязке, а также обозначим систему координат EPSG:3857. Жмем ОК.

Затем следует нажать на кнопку Start Georeferencing на той же панели, что и остальные. Изображение должно появится на карте.

Следует изменить опции прозрачности карты в свойствах слоя для того, чтобы видеть карту под изображением.

Следующим шагом будет создание нашей карты этажа.

Создадим новый слой через Layer > Create Layer > New Shapefile Layer. В окне выбираем тип Polygon и вновь выставляем нашу систему координат. Здесь же можно добавить поля свойств к слою, такие как id, имя и другие. После настройки всех необходимых параметров дадим слою подходящее имя, к примеру, по названию корпуса и этажа.

Далее нужно убедится, что созданный слой выбран и расположен сверху на панели слоев, а также следует настроить прозрачность слоя. Это позволит нам видеть через слой изображение с планом и карту, а также дает возможность отслеживать наложение объектов внутри слоя. Оно будет отображаться темнее.

Первый объект, который мы собираемся создать это основа нашего этажа. Для этого нужно войти в режим редактирования слоя. Нажимаем кнопку Toggle Editing, а затем Add Features. Обрисуем границы этажа. Когда начальная и конечная точка сойдутся, следует нажать правую кнопку и отрисованная геометрия сохранится. В диалоге сохранения объекта можно заполнить поля, созданные в слое (тип, имя и т.д.).

Теперь можно приступить к отрисовке помещений. Для более комфортной работы следует включить прилипание курсора к объектам на всех слоях. Для этого зайдем в Settings > Snapping Options и выставим параметр snapping mode на All layers. Также следует выбрать к каким объектам будет прилипать курсор. Нам нужно «To vertex and segment». Помещения отрисовываются по той же схеме, что и основа этажа, только теперь мы придерживаемся схемы помещений на заднем фоне. Также следует соблюдать прямые углы. Для этого нужно включить привязку к прямым углам. У каждого созданного помещения можно задать свойства для их дальнейшей обработки и сортировки.

Когда все помещения на этаже будут готовы, приступим к выводу нашего слоя в формат GeoJSON, который мы упоминали ранее в тексте. Для

этого следует нажать правой кнопкой по слою на панели слоев и выбрать пункт «Save as…». В появившемся меню выбрать нужный нам формат, имя и расположение файла [2].

Данный метод подготовки данных нужен нам для дальнейшего использования этих геоданных при создании интерактивной карты НИУ «БелГУ».

Перечислим различные преимущества, которые нам дает работа с представленными таким образом данными в нашем проекте.

Одним из преимуществ данного метода является то, что данные нашего плана становятся геопозиционированными, т.е. привязанными к определенному положению на карте.

Другое преимущество заключается в возможности задания различных параметров разным объектам на карте.

3 ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ КАРТЫ НИУ «БЕЛГУ»

3.1 Работа карты

Как было указано ранее, карта представляет собой веб-приложение и для ее использования достаточно зайти на сайт. Никаких установок не требуется.

Перейдя на сайт, вы сразу увидите карту. Ее центр будет находиться в определенной точке университета. Эта точка задается координатами. В дальнейшем планируется определять местоположение пользователя и центрировать карту относительно него.

При входе можно наблюдать следующее (см. Рисунок 2).

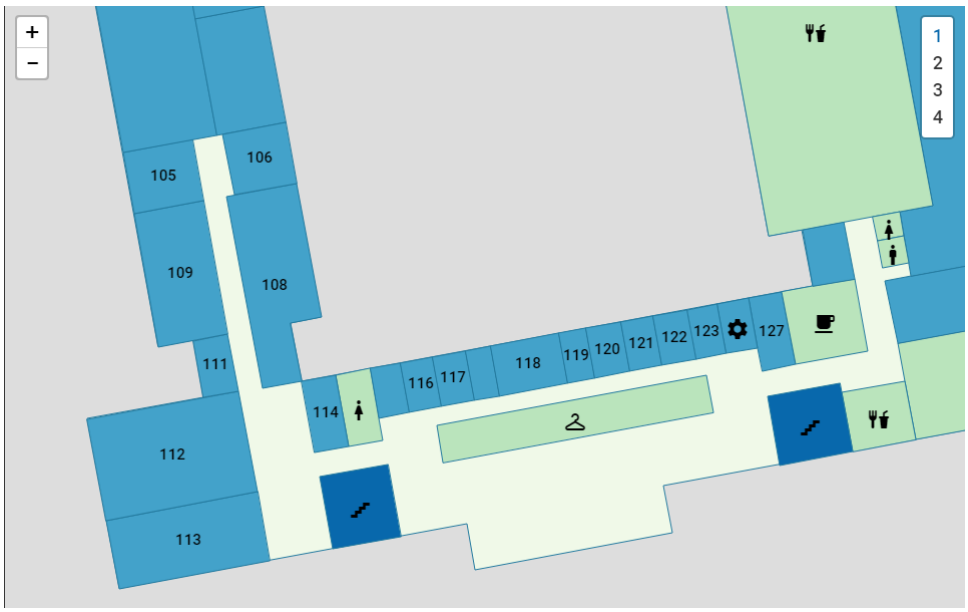

Рисунок 2 - Рабочая область карты

По углам вы можете наблюдать интерфейс взаимодействия с картой.

В левом верхнем углу находится переключатель уровней увеличения. Такой же функционал выполняет колесико мыши.

В правом - переключатель этажей. Выполняет простую функцию смены этажей. Возможна опция его сворачивания при большом количестве этажей.

Все помещения закрашены различными цветами в зависимости от их назначения. Классов помещений несколько:

- Технические помещения. К ним относятся всяческие подсобные помещения, а также все не учебные помещения, включая кафедры и деканат. В дальнейшем планируется выделение служебных помещений, таких как упомянутые кафедры и деканаты, а также других, либо в отдельный класс помещений, либо присвоение им специальных пиктограмм.
- Места интереса. Это всяческие столовые, гардероб и другие подобные помещения.
- Лестницы и лифты.
- Учебные аудитории. Все лекционные и другие учебные помещения.

Такое разделение по цветовой гамме позволяет легче воспринимать карту, а также быстрее находить глазами нужны тип аудиторий.

Цвета подобраны не случайно. Мы воспользовались сервисом ColorBrewer. Он позволяет подобрать рекомендуемые для использования в картографии цвета. Они в свою очередь ссылаются на исследования картографа Cynthia A. Brewer. Она написала книгу под названием Designing Better Maps: A Guide for GIS Users. В ней описаны рекомендации по дизайну карт [1].

Также использованы описанные в предыдущей главе иконки Material disign icons. Не будем здесь подробно на них останавливаться, напомним лишь о том, что каждому классу мест интереса и другим объектам присвоена своя пиктограмма.

Реализована возможность сокрытия иконок на разных уровнях увеличения (см. Рисунок 3).

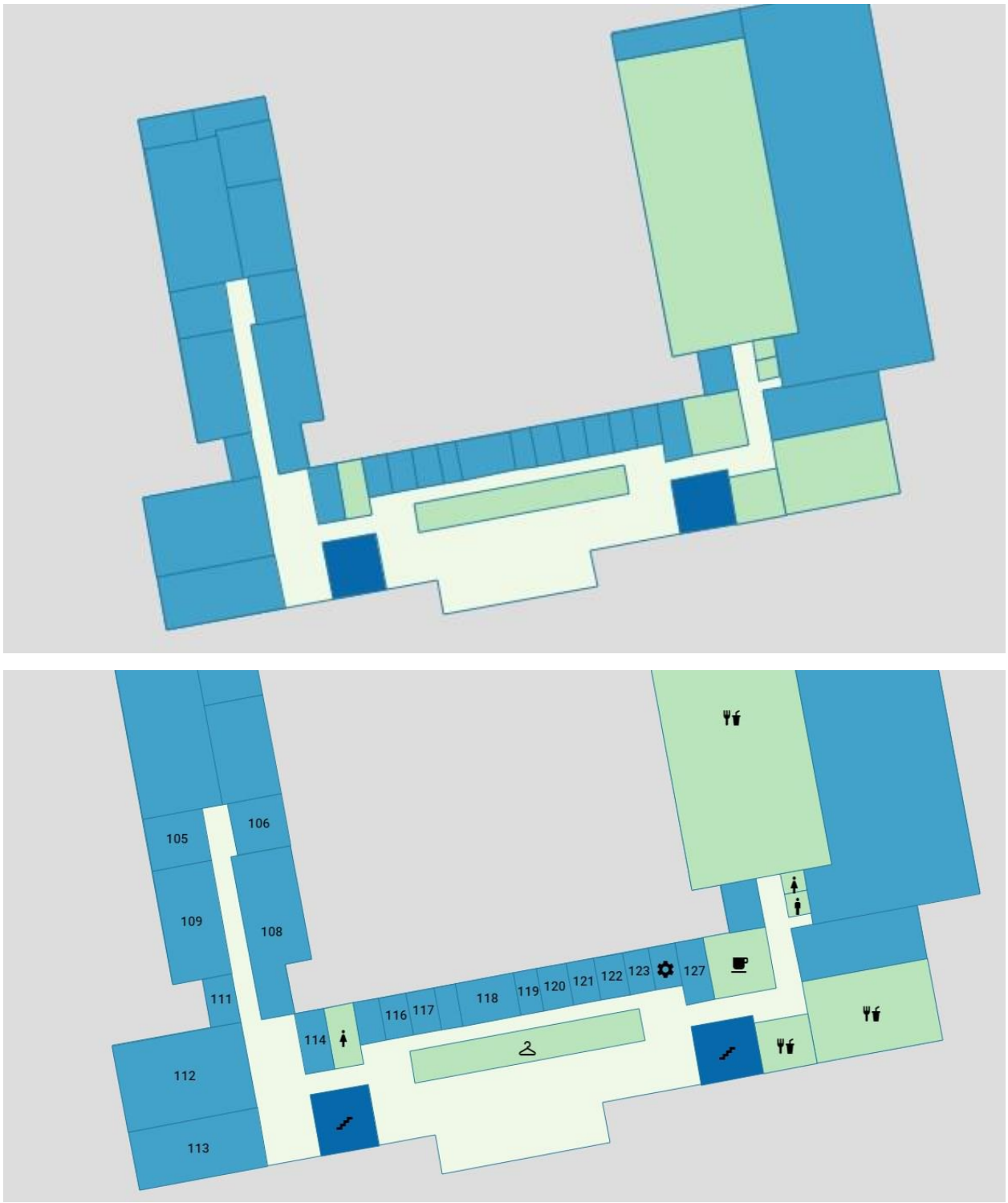

Рисунок 3 – Пример использования функции сокрытия маркеров в зависимости от уровня увеличения

При нажатии на помещение появляется всплывающее окно с информацией о помещении (Рисунок 4). В дальнейшем планируется расширение и классификация данной информации.

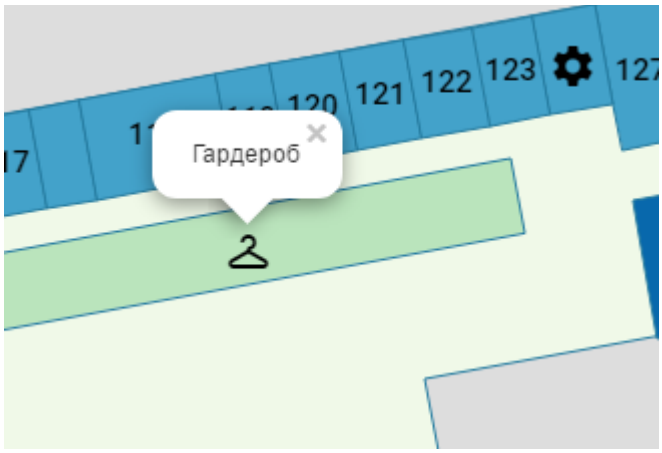

Рисунок 4 - Всплывающая информация

Также одним из планируемых улучшений является поиск помещения или объекта, а также расширение покрытия карты и, вследствие этого, добавление механизма выбора корпуса.

3.2 Использование Material design icons

В качестве обозначения мест интереса можно использовать различные средства, такие как текст, картинки, различные иконки. Было решено использовать такой инструмент как Material design icons.

Это набор иконок, Он имеет множество преимуществ перед другими решениями. К примеру легковесность, богатый выбор иконок, лаконичный дизайн и открытость использования. Но главным его преимуществом является способ представления на странице. По сути это такой специальный шрифт, используя который на выходе вы получаете векторную иконку вместо текста.

Они описываются с помощью стилей CSS. Для использования нужно подключить их описание и задать тегу нужный код класса, представляющий нужную вам иконку. В нашем проекте мы задаем этот код на этапе создания комнаты, прописывая его в свойствах объекта в поле name. Вот как это выглядит в данных:

```
"features": [
   {
     "type": "Feature",
     "properties": {
       "id": 3,
       "type": "inter",
       "name": "mdi-food-fork-drink",
       "level": 1,
       "popup": "Преподавательская столовая"
     },
     "geometry": {
       "type": "Polygon",
       "coordinates": [
         [
           [
             36.57939345903364,
             50.6215198821121
             …
         ]
    ]
}
```

В данном примере мы описываем преподавательскую столовую. Для отображения нужной нам иконки мы прописываем в поле name “mdi-food-fork-drink”. Это отобразит иконку столовой (см. Рисунок 5).

Рисунок 5 – Иконки на карте

В дальнейшем следует создать единую легенду для всех мест интереса и присвоить им соответствующие значения.

ЗАКЛЮЧЕНИЕ

Интерактивные карты являются живой и быстроразвивающейся областью навигационных технологий. На рынке стало появляться большое количество решений и различных продуктов. Наш проект не теряется среди этого потока благодаря использованию опыта уже существующих технологий, а также его узкой направленности на навигацию в помещениях. Среди навигационных систем эта область представлена не так широко, как например навигация по городу.

Задачи, которые мы себе ставили на начальных этапах выполнения проекта, были реализованы.

Мы изучили потребностей потенциальных пользователей путем их опроса. Выявили различные пожелания касательно наполняемости и функционала карты. В дальнейшей разработке учитывались эти пожелания и на их основе были сформированы основные требования к проекту и используемым в нем технологиям.

Мы изучили различные технологии и варианты решения поставленных нами требований к проекту и выбрали подходящие нам.

Кроме самой карты была разработана система данных для отображения на ней, к которой мы также разработали инструкцию для дальнейшего внедрения новых областей на нашу карту

Мы сумели добиться жизнеспособного и рабочего продукта, благодаря модульной и расширяемой структуре которого мы оставили возможности для его дальнейшего развития и расширения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Brewer C. Designing better maps: a guide for GIS users // ESRI Press, 2005 – 454 p.

2 Georeference the floor plan image. - 2017. - (Англ.). - URL: https://github.com/eegeo/eegeo-indoor-maps-api/blob/master/TUTORIAL.md#georeference-the-floor-plan-image

3 GIS-Lab неформальное сообщество специалистов в области ГИС и ДЗЗ. - 2017. - (Рус.). - URL: http://gis-lab. info/qa/google-web.html

4 HTML и SVG: создаём интерактивную карту - 2017. - (Рус.). - URL: https://habrahabr.ru/post/127994/

5 Material design icons - 2017. - (Англ.). - URL: https://github.com/Templarian/MaterialDesign

6 SVG в вебе. Практическое руководство - 2017. - (Англ.). - URL: https://svgontheweb.com/ru/

7 Геоинформационный портал ГИС-ассоциации. - 2017. - (Рус.). - URL: http://www.gisa.ru/86604.html.

8 Документация API Яндекс. Карт. - 2017. - (Рус.). - URL: https://tech.yandex.ru/maps/doc/jsapi/2.1/quick-start/tasks/quick-start-docpage/.

9 Документация Google Maps API. - 2017. - (Рус.). - URL: https://developers.google.com/maps/?hl=ru.

10 Касьянова Е. Л. Интерактивные карты современный метод представления информации // Интерэкспо Гео-Сибирь. 2008. №-2. С.199-202

11 Надыров И. О. Описание концепции интерактивной карты // Вестник СГУГиТ (Сибирского государственного университета геосистем и технологий). 2011. №1 (14). С.62-68

12 Сайт API Яндекс Карт – 2017 – URL: https://tech.yandex.ru/maps/.

13 Сайт JavaScript библиотеки Leaflet. - 2017. - (Англ.). - URL: http://leafletjs.com/.

14 Сайт JavaScript библиотеки OpenLayers. - 2017. - (Англ.). - URL: https://openlayers.org/.

15 Сайт картографического веб-сервиса OpenStreetMap – 2017 – URL: https://www.openstreetmap.org/about.

16 Сайт картографического веб-сервиса WikiMapia – 2017 – URL: http://wikimapia.org/about/.

17 Сайт картографического веб-сервиса компании Microsoft. – 2017. – (Англ.). – URL: http://www.microsoft.com/maps/.

18 Сайт компании CloudMade. - 2017. - (Англ.). - URL: http://cloudmade.com/products.

19 Создание пользовательского слоя карты. - 2017. - (Рус.). - URL: https://tech.yandex.ru/maps/doc/jsapi/1.x/dg/tasks/how-to-create-user-layer-docpage/

20 Спецификация формата GeoJSON - 2017. - (Англ.). - URL: http://geojson.org/geojson-spec.html

21 Страница проекта QGIS - 2017. - (Англ.). - URL: http://qgis.org/ru/site/about/index.html