

**«Санкт-Петербургский государственный электротехнический университет
«ЛЭТИ» им. В.И.Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)**

Направление	09.03.01 – Информатика и вычислительная техника
Профиль	Вычислительные машины, комплексы системы и сети
Факультет	КТИ
Кафедра	ВТ

К защите допустить

Зав. кафедрой

Куприянов М. С.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
БАКАЛАВРА**

**Тема: ИНТЕРАКТИВНАЯ КАРТА УНИВЕРСИТЕТА С
ОТОБРАЖЕНИЕМ РАСПИСАНИЯ
ПО АУДИТОРИЯМ**

Студент	_____	Шохин Е. П.
Руководитель	_____	Калишенко Е. Л.
Консультант	к.т.н., доцент _____	Зуев И. С.
Консультант	д.э.н., профессор _____	Семенов В. П.

Санкт-Петербург

2016

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю

Зав. Кафедры ВТ

_____ Куприянов М. С.

« ____ » _____ 20__ г.

Студент: Шохин Е. П., группа 2306

Тема работы: интерактивная карта университета с отображением расписания
по аудиториям

Место выполнения ВКР: СПбГЭТУ «ЛЭТИ», кафедра ВТ

Исходные данные (технические требования): перечень требований к разработке

Содержание ВКР: Интерактивные гео-карты, выбор технологий разработки, разработка
приложения, тестирование, выводы

Перечень отчетных материалов: пояснительная записка, иллюстративный материал,
реферат, презентация, разработанная программа с исходными кодами.

Дополнительные разделы: Обеспечение качества разработки программного продукта

Дата выдачи задания

Дата представления ВКР к защите

«15» Марта 2016г.

« ____ » _____ 20__ г.

Студент

Шохин Е. П.

Руководитель

Калишенко Е. Л.

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю

Зав. Кафедры ВТ

_____ Куприянов М. С.

«__» _____ 20__ г.

Студент: Шохин Е. П., группа 2306

Тема работы: Интерактивная карта университета с отображением расписания по аудиториям

№ п/п	Наименование работ	Срок выполнения
1	Анализ технологий	15.03.16 – 01.04.16
2	Разработка архитектурной схемы	01.04.16 – 03.04.16
3	Разработка схемы таблиц базы данных	03.04.16 – 05.04.16
4	Создание веб-страницы с интерактивностью	05.04.16 – 09.04.16
5	Создание сервера с логикой управления базой данных	09.04.16 – 16.04.16
6	Создание связи страницы с сервером, отображение данных	16.04.16 – 20.04.16
7	Отладка и тестирование	20.04.16 – 26.04.16
8	Оформление пояснительной записки и слайдов для защиты	26.04.16 – 07.06.16
9	Предварительная защита	08.06.16
10	Представление к защите	17.06.16

Студент _____ Шохин Е. П.

Руководитель _____ Калишенко Е. Л.

РЕФЕРАТ

Пояснительная записка 47 стр., 9 рис., 3 табл.

ИНТЕРАКТИВНАЯ КАРТА, SVG, NODEJS, САЙТ, ВЕКТОРНАЯ ГРАФИКА, БАЗА ДАННЫХ, ИНТЕРАКТИВНОСТЬ.

Цель: разработка интерактивной карты университета с отображением расписания по аудиториям.

Объект исследования:

Формирование карты университета для оптимального отображения информации по отдельным аудиториям в интерактивном режиме.

Предмет исследования:

Расположение актуальных данных, касаемо текущего расписания в университете, на карте и ее отображение.

В выпускной квалификационной работе приводится обзор основных средств корректного использования интерактивных карт для отображения данных об объектах. Описаны различные технологии и библиотека, которые будут использоваться для создания данной системы, а также приведено обоснование использования именно этих технологий.

Были проанализированы различные архитектурные решения, а также произведено формальное тестирование программы.

Программный продукт был разработан в среде WebStorm с использованием различных технологий с применением языка JavaScript.

ABSTRACT

Final qualifying work "Interactive map of the University with the display of the schedule is devoted to the study and implementation of ways of interacting with image data in the form of a map.

The purpose of the work is to create an interactive map, which will efficiently and conveniently display information about each audience, and will allow the most convenient to navigate in the University.

The results can be useful for improving the existing system and use this program to display data of other buildings.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ.....	7
ВВЕДЕНИЕ	8
1 Интерактивные гео-карты.....	9
1.1 Гео-карты: виды и сферы применения	9
1.2 Интерактивные Flash – карты.....	11
1.3 ГИС - карты.....	12
1.4 Google Maps.....	14
1.5 2ГИС	15
1.6 Сравнительный анализ интерактивных карт	16
2 Выбор технологий разработки	18
2.1 Сравнение растровой и векторной графики.....	18
2.1 SVG – формат.....	19
2.2 Графический редактор	21
2.3 Архитектура решения.....	22
2.3.1 Использование базы данных.....	22
3 Разработка интерактивной карты.....	26
3.1 HTML – страница	26
3.2 Веб – сервер.....	27
3.3 Интерактивность.....	28
3.4 Взаимодействие с данными	31
3.5 Отображение данных	33
4 Результаты разработки	35
4.1 Загрузка на сервер	35
4.2 Организация тестирования программы	35
4.3 Испытание программы	36
5 Обеспечение качества разработки	39
5.1 Понятие качества и стандарты качества.....	39
5.2 Оценка качества разработки интерактивной карты	44
5.3 Вывод	45
ЗАКЛЮЧЕНИЕ.....	46
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	47

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяют следующие термины с соответствующими определениями:

Кроссплатформенность – свойство программного обеспечения, работать более чем на одной аппаратной платформе и/или операционной системе.

БД – база данных — представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны

СУБД – система управления базой данных.

ID – идентификатор – уникальный признак объекта, позволяющий отличать его от других объектов.

PopUp – модальное окно – в графическом интерфейсе пользователя называется окно, блокирующее работу пользователя с родительским приложением до тех пор, пока пользователь это окно не закроет. Модальными преимущественно реализованы диалоговые окна.

Прикладное ПО, приложение – программа, предназначенная для выполнения определенных пользовательских задач и рассчитанная на непосредственное взаимодействие с пользователем.

ME – браузер Microsoft Edge.

ВВЕДЕНИЕ

Программа разрабатывается в рамках создания обобщенной системы ориентирования в стенах СПбГЭТУ «ЛЭТИ».

Процедура оценки и формирования расписания сопровождается большим количеством документов и не содержит в себе интеграцию с картой вуза.

В связи с увеличением расписания и использования некоторых аудиторий для проведения занятий, до которых достаточно сложно добраться, было принято решение найти способ объединить планы эвакуации каждого корпуса с расписанием.

Цель: разработка интерактивной карты университета с отображением расписания по аудиториям.

Задачи:

- 1) Оценка направлений разработки;
- 2) Выбор вида изображений для последующей работы;
- 3) Оценка форматов выбранного типа изображений для конкретного направления разработки;
- 4) Выбор фреймворка для преобразований, рисования и обработки графической составляющей;
- 5) Преобразование планов эвакуаций и корректировка полученных изображений;
- 6) Добавление первоначальной интерактивности;
- 7) Оценка различных видов систем управления базой данных и базы данных и выбор наиболее подходящих под конкретное задание;
- 8) Составление базы данных и заполнение данными;
- 9) Совмещение карты с базой данных;
- 10) Корректировка представления информации, графики и интерактивности под различные виды и способы представления;

1 Интерактивные гео-карты

Интерактивная карта – это карта, которой можно управлять, т.е. перемещаться по карте, фильтровать информацию, которая отображается в нужный вам момент времени, искать информацию по карте при помощи формы поиска и фильтрации данных, получать более подробную информацию по интересующему вас объекту карты. Видов карт бесчисленное множество. Различаются они как по визуальной составляющей, так и по функциональной. Все это зависит от того, для какой задачи та или иная модель предусмотрена.

1.1 Гео-карты: виды и сферы применения

Не так давно люди, желающие попасть в какое-либо место, определиться с маршрутом или узнать информацию о какой-либо местности, разворачивали на столе огромные атласы и тратили уйму времени на поиск данных о нужном их объекте. В большинстве случаев, даже используя различные вспомогательные средства: содержание, алфавитный указатель, примерную область местоположения, быстро найти нужную точку удавалось не каждый раз. Что еще говорить об удобстве и компактности. Часто подобные атласы и карты занимали огромные по толщине книги или не уместившиеся на полу квартиры полотна бумаги. Это было крайне неудобно.

Совсем недавно на смену громоздким картам и атласам, напечатанным на бумаге, пришли электронные, или интерактивные, доступ к которым осуществлялся либо переходом на соответствующую ссылку, или обращением к конкретному приложению. Возможности их поражали: в считанные секунды, введя название нужного вам объекта и нажав на соответствующую кнопку, нужный фрагмент с местностью оказывался перед глазами с указанием подробной информацией или ссылкой на ресурс об объекте. Но главным преимуществом перед бумажными картами - реализация двустороннего диалогового взаимодействия человека и компьютера и представление в виде визуальной информационной системы. Степенью интерактивности они могут

различаться. В одном случае можно просто получить информацию о конкретном объекте, а в другом построить маршрут или разложить здание по этажам и офисам.

Используя технологию геоинформационных систем, пользователь способен видеть лишь ту часть карты, которая его интересует в конкретный момент времени. Улучшать визуальную составляющую можно сколько угодно. Например, совместить карту со снимками из космоса. Идеально объединив их можно получить не просто плоскость с условными обозначениями, а реальные пейзажи, снятые со спутника.

Интерактивные карты могут быть представлены в нескольких вариантах: распространяться на каком-либо внешнем носителе информации, в качестве приложения на мобильные платформы или ПК, а также в Интернете. В некоторых случаях для качественного отображения данных и самой визуальной составляющей требуется установить некоторое программное обеспечение, иногда – самой последней версии.

Интерактивные карты незаменимы в различных аспектах, особенно когда необходимо что-либо найти, и тем более желательно показать расположение рассматриваемого объекта, неважно, что это будет – план здания, магазин, университет, целый комплекс зданий или даже целая страна на карте.

Когда появились первые электронные карты, они отображали только информацию, касаемо некоторой области земной поверхности и обладали функцией по большей мере справочника. Поэтому их больше всего использовали в образовательной, познавательной и исследовательской деятельности.

В настоящее время карты усложнились. Теперь их можно рассматривать не как обычный справочник, а как подробную библиотеку разнообразных сведений. Примерами могут служить карты магазинов и торговых центров, с отображением информации по каждому отделу, карты наземного транспорта и метро.

1.2 Интерактивные Flash – карты

Интерактивные flash-карты получили очень сильное распространение в корпоративных и образовательных средах. Сайты, на которых они распространены, гораздо полезнее и приятнее визуально для посетителей. На картах можно показать не только информацию о компании, о предполагаемых объектах для сдачи, но и имеет возможность увеличения определенного фрагмента карты, что позволит сориентироваться в её расположении. Так же интерактивная карта имеет свойства наглядно предоставить аналитические, статистические и иные данные об объекте, месте, которые позволят пользователю сориентироваться, например, карта многоэтажного дома, квартиры которого в скором времени будут выставлены на продажу, позволят покупателям выбрать понравившийся им этаж, и квартиру с нужной им планировкой.

1.3 ГИС - карты

Для получения картографических данных, связанных с какой-либо географической информацией лучше всего подходят ГИС-карты [1]. Данный тип интерактивных карт позволит лучше понять местность, подробнее узнать о построенном маршруте следования. В дополнении ко всему часто указывается и расписание движения транспорта, а в некоторых ситуациях и примерное их расположение.

Большое количество ГИС-карт применяется в навигации. Если нужно отправиться в какое-либо место, то в первую очередь пользователь задается вопросами: где это находится и как быстро туда добраться, то есть построить оптимальный маршрут. Найти нужное место на карте – самая простая задача, а вот построить путь до этой точки куда сложнее. Существует два способа решения данной навигационной задачи: построение вручную, анализируя все данные самому или позволить системе автоматически построить маршрут. Большинство интерактивных карт позволяют с высокой точностью и правильностью подобрать тот маршрут, который будет считаться одним из оптимальных. Все это благодаря специальному модулю – Driving Direction, который большинство сервисов, таких как Google Maps, Yandex Карты встраивают себе на сайты.

При поиске нужного для человека объекта зачастую он обладает каким-либо минимумом данных: номер телефона, название ВУЗа, адрес – улица и номер дома, названия населенного пункта и так далее. Для отображения системы зданий конкретного учебного заведения информации у наблюдателя может быть еще меньше.

Степень детализации для интерактивных карт зависит больше от того, какую задачу они будут решать. Если нужно найти какой-либо населенный пункт, то ограничения будут не значительные. При поиске конкретного здания в том или ином городе степень детализации ценится так же высоко, как и актуальность и новизна информации. При отображении системы зданий тре-

буется знать их точное местоположение, а также то, что находится на каждом этаже с возможностью поиска по помещениям. Бывают такие ситуации, когда нужная аудитория, которая должна быть на конкретном этаже, присутствует совсем в другой стороне и это может запутать незнающего человека. Поэтому, присутствие поиска, как по названию, так и по некоторым другим данным: группа, предмет или преподаватель сыграет большую роль в представлении ее местоположения, а также получения информации о том, что там находится.

1.4 Google Maps

Лидером среди современных сервисов для представления интерактивных карт является Google Maps [10]. Обладание мощным API, доступом к спутниковым снимкам и информации о мероприятиях (погодных и транспортных) для конкретной области подчеркивает его лидерство. Особенно стоит подчеркнуть режим просмотра улиц, где пользователь способен, в прямом смысле, пройтись по улицам города и узнать, как он выглядит.

Из плюсов стоит выделить обширное количество информации по объектам, указание не только номера телефона и режима работы, но и возможность взаимодействия с пользователем: отзывы, оценки, доработки, а также наличие фотографий для удобной ориентации в поиске.

Google Maps предназначен в большинстве случаев для того, чтобы найти самые детализированные спутниковые фотографии для различных регионов: стран, городов и отдельные улицы, но он не имеет функциональности для того, чтобы заглянуть внутрь конкретного здания. Пользователь не может узнать, где вход, а также, если нужно для человека место находится внутри, то мы не можем определить ни этаж, ни примерный маршрут до него, что затруднит поиск.

1.5 2ГИС

2ГИС [9] выгодно отличается от других сервисов тем, что помимо общей информации предоставляет еще и дополнительные данные. Кроме местоположения нужного для пользователя объекта, можно узнать такую информацию как: список расположенных в здании организаций, номера их телефонов, график работы, а также ссылки на иную дополнительную полезную информацию.

Для каждой организации в справочнике приведены адрес, телефон, время работы, интернет-адрес и расположение входа в здание. Помимо этого, в так называемой карточке компании может содержаться информация, специфическая для рода деятельности организации. Например, способы оплаты, виды кухни (для заведений общепита), перечень услуг и т.д.

Информация в справочнике по имеющимся организациям актуализируется четыре раза в год специалистами. Удобный API, понятная документация и взаимодействие с разработчиками, позволяют создавать виджеты и карты для своих целей используя все функции, которые нам предлагают, а также создавать свои, но каркасом должна быть исходная карта 2ГИС, что немного ограничивает разработчика.

Карты 2ГИС отрисовываются на основе спутниковых снимков территории, а затем выверяются специалистами - «пешеходами». Трехмерные модели зданий изготавливаются на основе снимков строения с нескольких ракурсов.

В 2014 году в 2ГИС появились «Этажи»: подробные схемы внутреннего устройства торговых центров. Впервые Этажи были запущены с планами шести торговых центров Москвы.

1.6 Сравнительный анализ интерактивных карт

Таблица 1 – Функциональный анализ интерактивных карт

	Google Maps	Яндекс.Карты	2ГИС
Покрытие	Весь мир	Лучшее покрытие России, чем у других, но хуже мир	Среднее
Детализация	Хорошая детализация по всему миру. На карте России могут отсутствовать крупные города. В плане отображения невидная детализация. Объекты хорошо видны только при достаточно сильном приближении.	Хорошая детализация России, достаточная в мире.	Самая лучшая детализация.
Детализация на уровне зданий	Крупные торговые центры в виде фотографий.	Нет	Торговые центры и некоторые здания.
Использование офлайн	Да. Большой объем загрузки данных.	Да. Большой объем загрузки данных.	Да
Редактирование	Сообщение об ошибке	Только «Народная карта»	Сообщение об ошибке
Выбор отображения	Карта, спутник, Велокарта, общественный транспорт	Карта, спутник, народная карта	Карта
Местоположение соответствует реалиям	Средне	Средне	Отлично
Охват территорий	Отлично	Отлично	Ограничен
Универсальный поиск	Интеллектуальный поиск	Интеллектуальный поиск	Да

Продолжение таблицы 1

Справочная информация	Хуже других знает российские организации	Подробная информация об организациях	Подробная информация об организациях. Ежемесячные обновления
Актуальность гео-данных	Нет информации	Нет информации	Обновления каждый месяц
Интерфейс	Современный интерфейс. Некоторые функции не до конца понятны на интуитивном уровне	Современный интерфейс. Осуществление большинства функций возможно в два шага	Для некоторых устройств не адаптирован для некоторых устройств.
API	Доступный API	Доступный API. Платное использование.	Доступный API
Итог:	Удобное и функциональное построение маршрутов. География на уровне мира, но хуже отображение на уровне городов и улиц.	Хорошо знает географию и организации по всей России.	Детальная информация по организациям и высокая детализация карт в городах присутствия.

2 Выбор технологий разработки

Для того чтобы разработать качественное программное обеспечение, нужно определиться с тем, над чем будет проходить работа. Изучить исходные файлы, подобрать наиболее подходящие программы, которые позволят качественно, и не прикладывая больших усилий, реализовать поставленную задачу.

2.1 Сравнение растровой и векторной графики

Таблица 2 – Анализ видов графики

	Растровая графика	Векторная графика
Элементарный объект	Пиксель	Контуры и линии
Изображение	Совокупность пикселей	Совокупность объектов
Фотографическое качество	Да	Нет
Объем памяти	Очень большой	Зависит от количества объектов
Масштабирование	Потеря качества	Да
Использование	Хранение фотографии, элементов интерфейса	Чертежи, макеты, планы, рисунки с четкими контурами

Вывод:

Произведя сравнение растрового и векторного формата изображений и того, что разработка интерактивной карты будет выглядеть как интернет-страница, то можно сделать вывод, что векторная графика подходит больше всего. Самым популярным и удобным форматом, который используется в интернете и позволяет без потери качества изменять изображение и обращаться к каждому элементу является SVG.

2.1 SVG – формат

SVG (Scalable Vector Graphics standard, стандарт масштабируемой векторной графики) [11] – векторный формат изображения, который представляет собой XML разметку, оптимально отображаемую в браузерах. Данная технология позволяет отобразить на странице как текст, так и графику с интерактивностью и анимацией.

Для работы с большинством векторных форматов требуются специальное программное обеспечение, в котором оно было создано. Часто бывает, что полученное изображение не всегда совместимо с другими и достаточно трудно найти ПО для конвертации в тот или иной формат без потери качества и интерактивности. SVG представляет собой совмещение XML разметки и каскадной таблицы стилей CSS, с помощью которой можно манипулировать и изменять свойства объектов. Данный формат имеет очень полезное свойство – чтобы его изменить, в большинстве случаев не нужно обладать специальной утилитой. Редактировать SVG можно в любом текстовом редакторе. Другие программы нужны лишь для того, чтобы визуально представить XML код.

Возможности:

- Задание любой фигуры с помощью описания путей, обозначения отдельных точек, использование специальных конструкций позволяет отрисовать практически любой объект.
- Использование таблицы стилей позволяет воспользоваться огромным спектром свойств, которые присущи объекту: заливка, прозрачность и т. д.
- Интерактивность.
- Воспользовавшись языком программирования JavaScript можно добавлять к отдельным элементам или группе сценарии, которые в свою очередь будут связаны с изменением свойств объекта, его положения, сложных математических вычислений или передачи данных.

Достоинства:

- Текстовый формат — файлы SVG можно читать и редактировать (при наличии некоторых навыков) при помощи обычных текстовых редакторов;
- Масштабируемость;
- SVG — открытый стандарт. В отличие от некоторых других форматов, SVG не является чьей-либо собственностью;
- SVG документы легко интегрируются с HTML и XHTML документами;
- Совместимость с CSS (англ. *Cascading Style Sheets*). Отображением (форматированием и декорированием) SVG элементов можно управлять с помощью таблицы стилей CSS 2.0 и её расширений, либо напрямую с помощью атрибутов SVG элементов.

Недостатки:

- SVG наследует все недостатки XML, такие как большой размер файла;
- Сложность использования в крупных картографических приложениях из-за того, что для правильного отображения маленькой части изображения документ необходимо прочитать целиком;
- Чем больше в изображении мелких деталей, тем быстрее растёт размер SVG-данных.

2.2 Графический редактор

Существует большое количество программ для работы с SVG – форматом представления векторных изображений, каждая из которых имеет свой ряд плюсов и минусов. К тому же, плюсом данного формата является то, что редактировать его можно из любого текстового редактора, так как он представляет собой язык XML. Проведем сравнительную характеристику двух программных продуктов для работы с векторной графикой: Inkscape и CorelDraw.

Таблица 3 – Сравнение программ для работы с векторной графикой

	CorelDraw	Inkscape
Цена	Более 300\$	Бесплатно
Операционная система	Windows	Кроссплатформенно
Легкость изучения	Требуется подготовка	Несложен в практическом изучении
Особые возможности	Обширный набор инструментов, работа с базами данных	Возможность редактировать объекты с помощью встроенного XML редактора
Поддержка SVG	Да	Да
Векторизация	Да	Да
Исправление существующего кода	Незначительные исправления	Включение дополнительных библиотек и приведение кода к стандарту Inkscape

2.3 Архитектура решения

Перед тем как приступить к разработке следует разобраться с взаимодействием отдельных компонентов и модулей программы друг с другом. Где будут храниться данные, как будут передаваться, каким образом происходит взаимодействие модулей.

2.3.1 Использование базы данных

После того, как с созданием модели карты закончено и при добавлении ее на html – страницу она отображается корректно, работает анимация по нажатию клавиш мыши стоит задуматься о том, где и каким образом хранить информацию о расписании по каждой аудитории. В данном проекте используется база данных MySQL.

Чтобы верно распределить данные и структурно обозначить их по категориям составим несколько таблиц, опишем каждое поле и составим UML-диаграмму связей между ними. На рисунке 1 представлена ER – диаграмма связей между отдельными таблицами в базе данных.

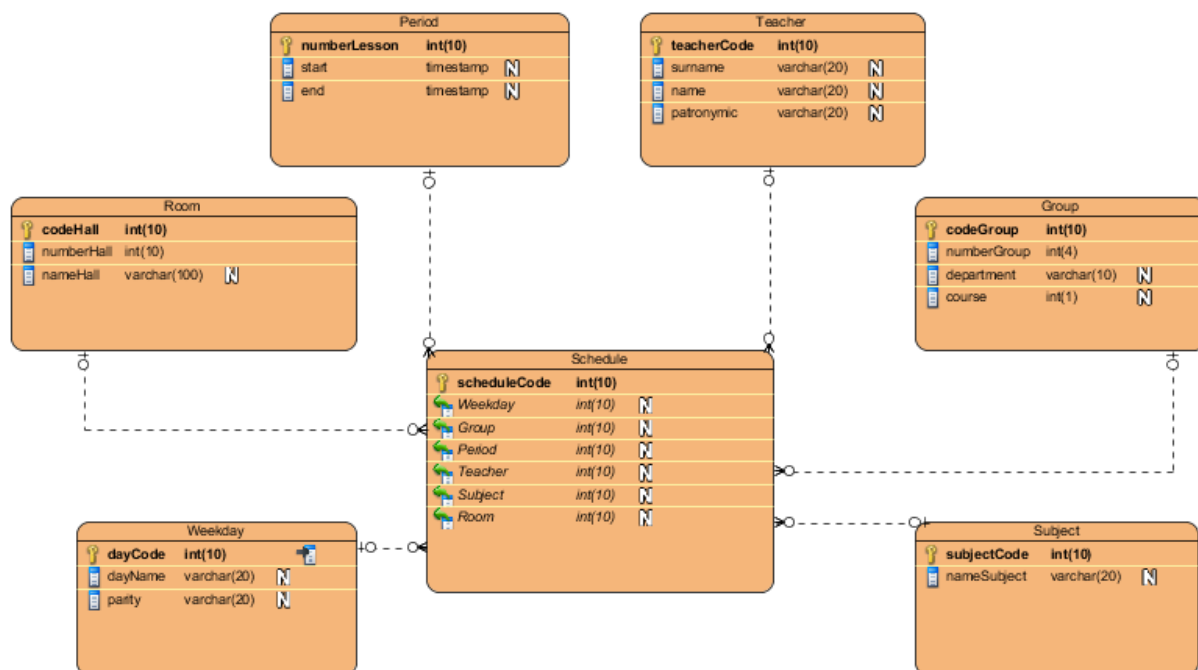


Рисунок 1 – ER – диаграмма базы данных

Опишем каждую структуру в отдельности.

- 1) Таблица Group показывает множество групп, на которые разбиты студенты.
 - a. codeGroup – уникальный идентификатор для каждой группы;
 - b. numberGroup – номер группы;
 - c. department – факультет, к которому принадлежит данная группа;
 - d. course – курс, к которому принадлежит данная группа;
- 2) Таблица Teacher является минимальным описанием каждого преподавателя вуза.
 - a. teacherCode – уникальный код для каждого преподавателя;
 - b. surname – фамилия преподавателя;
 - c. name – имя преподавателя;
 - d. patronymic – отчество преподавателя.
- 3) Таблица Period указывает на множество временных отрезков, во время которых проходит занятие.
 - a. numberLesson – номер занятия по порядку относительно начала дня;
 - b. start – начало пары;
 - c. end – конец пары.
- 4) Таблица Room содержит описание аудиторий учебного заведения.
 - a. codeHall – уникальный идентификатор аудитории;
 - b. numberHall – номер аудитории;
 - c. nameHall – название аудитории (если есть).
- 5) Таблица Weekday содержит описание дня недели.
 - a. dayCode – уникальный код дня недели;
 - b. dayName – название дня недели (Понедельник, Вторник и т. д.);
 - c. parity – четность недели.

6) Таблица Subject краткое описание дисциплин вуза.

- a. subjectCode – уникальный код для каждой дисциплины;
- b. nameSubject – название дисциплины.

7) Таблица Schedule – объединение составляющих параметров расписания в единое целое. Каждое поле данной таблицы является ссылкой на уникальный код той таблицы, информация которой должна содержаться в данном поле.

После создания и заполнения информацией всех таблиц стоит задуматься о том, каким образом и по каким критериям стоит выбирать данные. Расписание для каждой аудитории разделяется по трем параметрам: номер аудитории, день недели и четность этой недели в году. Имеем, что по данным параметрам стоит делать запрос к базе и совершать выборку данных.

2.3.2 Клиент-серверное взаимодействие

Для того чтобы качественно отобразить html-страницу и тестировать ее нужно создать такую систему, которая будет обрабатывать все данные, полученные от страницы и осуществлять асинхронное взаимодействие с клиентом.

В интернете можно найти множество примеров, которые позволяют работать с JavaScript и в основном все они связаны с библиотекой NodeJS. Node не является веб - сервером, сам по себе он ничего не делает, не содержит файлов конфигурации, в которых в большинстве серверов указывается путь до исходной страницы.

NodeJS обладает инструментарием, который позволит создать HTTP-сервер, подключить к нему базу данных и осуществить взаимодействие по Request и Response запросам с клиентом, в нашем случае – браузером.

Отобразим на рисунке 2 клиент-серверное взаимодействие в системе.

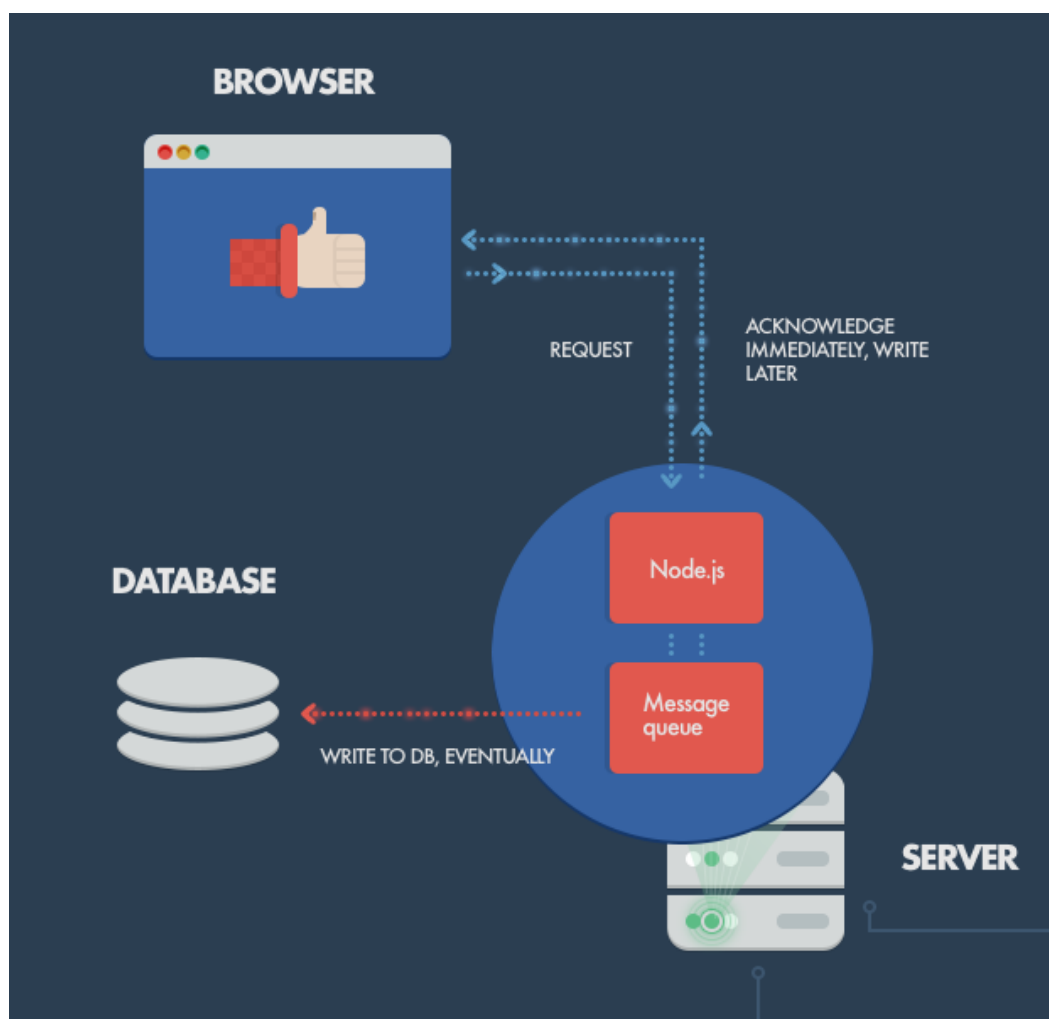


Рисунок 2 – Модель поведения клиент-серверного приложения

Поскольку обращение к базе данных является операцией блокирующей, то возникают проблемы, связанные с конкурентными соединениями. Встает вопрос о том, как обозначить действия клиента до того, как данные будут считаны из базы.

При использовании данного подхода повышается отзывчивость системы. Это полезно в том случае, что клиенту не требуется подтверждение о чтении или записи данных. При помощи специальной инфраструктуры для кэширования из данных формируется очередь, и все они обрабатываются отдельным процессом, предназначенным для данного типа задачи. Такое поведение можно реализовать с помощью других фреймворков, но все это будет происходить не с такой высокой скоростью и точностью, как с помощью NodeJS.

3 Разработка интерактивной карты

Определившись с архитектурой, способом хранения данных и взаимодействием модулей друг с другом стоит приступить к разработке как визуальной составляющей, так всей логике работы приложения в целом.

3.1 HTML – страница

Первоначально следует обратить внимание на то, как будут располагаться элементы на предполагаемой веб – странице, в каких тегах будет находиться тот или иной элемент.

1. SVG – элемент

Существует 4 способа того, как можно вставить SVG картинку в html документ, каждый из которых различается не только кодом, но и функциональностью. Рассмотрим их поподробнее и выберем наиболее подходящий с учетом того, что SVG каждого этажа у нас хранится в отдельном файле.

1.1. Тег <iframe>

```
<iframe src="SvgImg.svg">
  
</iframe>
```

Данный способ очень подходит для того, чтобы распределить отдельно по файлам изображение и JavaScript код, но манипуляции с главной страницы могут составить некоторые трудности.

1.2. Тег

```

```

При использовании данного тега с svg изображениями отключается любая интерактивность, которую можно использовать.

1.3. Тег <svg>

```
<svg width="300px" height="300px" xmlns="http://www.w3.org/2000/svg">
  <text x="10" y="50" font-size="30">My SVG</text>
</svg>
```

Этот метод работает почти во всех браузерах, очень качественно отображает и анимацию, и интерактивность, но имеет существенный минус.

Такой вариант возможен, когда имеем дело с небольшими SVG изображениями, все элементы нужно явным образом переписывать в код страницы.

1.4. Тег <object>

```
<object type="image/svg+xml" data="image.svg" width="200" height="200" >  
  Your browser does not support SVG  
</object>
```

Данный способ очень подходит в том случае, когда планируется использовать таблицы стилей, SVG изображения [5] и скрипты, разбив их на несколько файлов. Данный вариант имеет большую поддержку среди многих браузеров.

Вывод: изучив основные способы добавления SVG изображения на страницу, их плюсы и минусы можно прийти к выводу, что для того, чтобы использовать множество стилей для изменения свойств объектов и удобнее реализовать интерактивность стоит использовать тег <object>.

3.2 Веб – сервер

Для того чтобы начать разработку сервера, на котором будет происходить обработка всей поступившей информации, нужно разобраться с тем, как его создать. Воспользуемся инструментарием, который предлагает библиотека NodeJS.

Общепринято среди веб – разработчиков создавать единую папку, в которой будет находиться html страница со всеми приложенными к ней исходными файлами: скрипты, таблицы стилей, изображения, базы данных и т. д. В нашем случае эта папка будет называться – static. Поместим в нее все исходные файлы, распределив их по соответствующим папкам, зависящим от их назначения.

Далее, вне папки создадим файл server.js, в котором будет находиться код нашего сервера.

```
var http = require('http');  
var static = require('node-static');  
var file = new static.Server('./static');  
http.createServer(function(req, res) {  
  file.serve(req, res);  
}).listen(8080);
```

Как можно заметить, код для создания сервера выглядит очень компактно и занимает всего пару строк. В результате, запустив данный скрипт с помощью команды node server.js и открыв в браузере страницу localhost с

портом, который указан в `listen` (в данном случае это порт 8080), отобразится та страница, которая находилась в корне папки `static`.

После того как сервер создан, нужно подумать о том, чтобы он мог принимать запросы от клиента. Для этого нужно обозначить ту часть адреса, с которого мы неявно будем отправлять данные, и разделить их функционально. Обозначим функцию обработки запросов как `processGetData`, которая будет активизироваться, если запрос приходит с адреса `/getData`.

3.3 Интерактивность

Чтобы иметь возможность взаимодействовать с каждым объектом, из которого состоит та или иная SVG картинка, нужно во время ее создания каждому элементу присвоить уникальный идентификатор, а также желательно, чтобы каждая отдельная карта имела свой ID.

Основным инструментом работы с объектами на странице является DOM (Document Object Model) согласно которой страница представляет собой иерархию из множества элементов, тесно взаимодействующих друг с другом, иначе сказать – DOM это дерево тегов.

Рассмотрим на примере кода, как выглядит DOM дерево.

```
<html>
  <head>
    <title>Заголовок</title>
  </head>
  <body>
    Прекрасный документа
  </body>
</html>
```

Самым верхним тегом в дереве является `<html>` и оно начинает расти из него.

Внутри `<html>` находится два обязательных узла: `<head>` – отвечающий за заголовок и `<body>` – тело страницы.

Все теги образуют узлы-элементы (`element-node`), а текста представлен в виде текст-элементов (`text node`). Отобразим все это на рисунке 3.

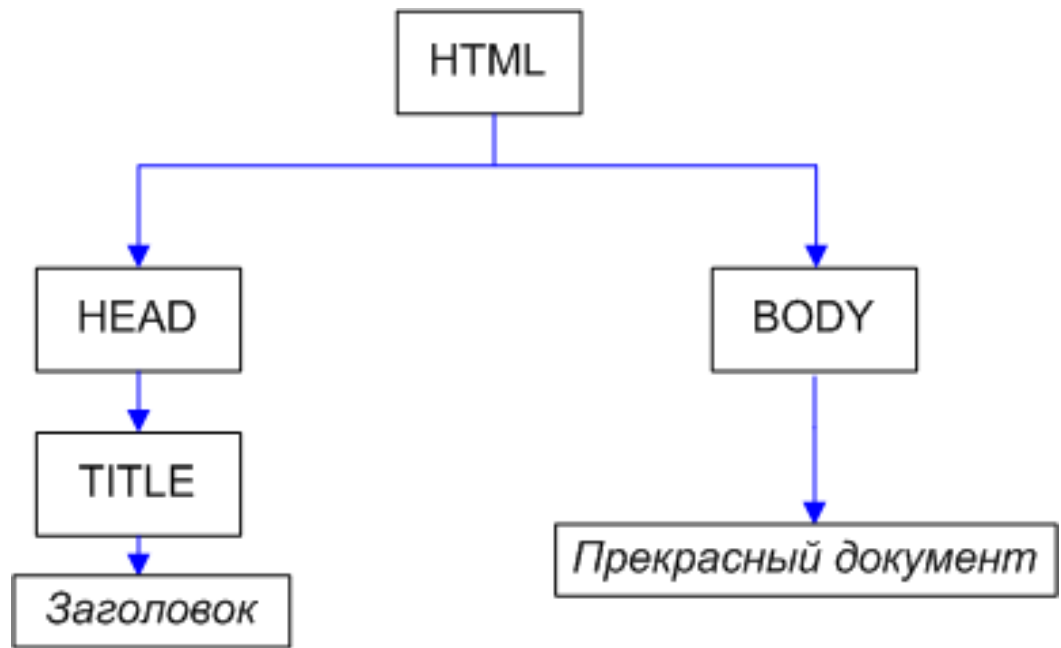


Рисунок 3 – Простейшее DOM -дерево

Разобравшись со структурой html – страницы, можно приступить к отображению на ней элементов и добавления к ним соответствующей интерактивности.

Первоначально мы имеем два типа карт: общая (lmap) и карта этажей (floor), каждая из которых находится в соответствующем теге <object> на странице.

Во время загрузки html нужно определить для каждого элемента тот сценарий, который будет выполняться вследствие того или иного действия[4]. В нашем случае имеем действие click и два элемента, обработав которые нужно после выполнения нажатия.

```

$(window).load(function () {
    var svgobject = document.getElementById('imap');
    if ('contentDocument' in svgobject)
        var svgdom = svgobject.contentDocument;

    svgdom.getElementById('lmap').addEventListener('click', function
corp(event) {
        var target = event.target;
        if(target.correspondingUseElement)
            target = target.correspondingUseElement;
        toCorp(target);
    }, false);
    var svgObjectFloor = document.getElementById('floorMap');
    if ('contentDocument' in svgobject)
        var svgDomFloor = svgObjectFloor.contentDocument;
    svgDomFloor.getElementById('floor').addEventListener('click', function
corp(event) {

```

```

        var target = event.target;
        if(target.correspondingUseElement)
            target = target.correspondingUseElement;
        toFloor(target);
    }, false);
});

```

Как можно заметить, оба сценария в своей обработке посылают в функции `toCorp` и `toFloor`, аналогично, переменную `target`, которая в свою очередь является тем элементом, на который было произведено нажатие.

Но, это не интерактивность. Для добавления переходов, затухания и более качественного отображения данных на странице, нужно обозначить определенные стили для элементов.

Основная карта, которая содержит в себе информацию о расположении корпусов, которые в свою очередь имеют неправильную форму, описаны с помощью `svg` элемента – `polygon`. ID каждого такого элемента является наименованием каждого из зданий, а чтобы отобразить их, воспользуемся элементом `<text>` и присвоим каждому имя.

Для того чтобы отобразить затухание на карте при наведении, у каждого элемента есть свойство `opacity[3]`. Задав определенную заливку для полигонов и в методе `hover` присвоив значение `opacity 0.5`, при наведении получим тот результат, которого и планировали.

Следует отметить, что при наведении на текстовое поле, затухание работать перестает и, к тому же мы можем обратиться к содержимому данного поля, что ни есть хорошо. Чтобы этого избежать, для текстового поля зададим определенные свойства:

```

#textId
{
    pointer-events: none;
    font-weight:normal;
    font-size:500px;
    font-family:'Arial';
    filter: alpha(Opacity=50);
    -moz-opacity: 0.5;
    -khtml-opacity: 0.5;
    cursor: default;

    -webkit-user-select: none;
    -moz-user-select: none;
}

```

Аналогичные действия нужно совершить и со стилями карты «Floor».

3.4 Взаимодействие с данными

Определившись с тем, что все данные расписания хранятся в базе данных, на уровне сервера нужно обеспечить connect с нужной нам системой таблиц и составить методы, благодаря которым будет возможно получить отобразить данные на html – странице.

Библиотека NodeJS [8] в своем инструментарии содержит модуль, позволяющий удобно подключиться к БД.

Перед тем, как реализовывать подключение, нужно подготовить данные, по которым будем производить запрос.

Напишем функцию, которая в своем содержании будет определять название текущей недели и высчитывать ее номер, а также записывать эти данные в заранее подготовленный SQL – запрос.

```
function createSelect() {  
    //Получение названия дня недели  
    var date = new Date(); //текущая дата  
    var nameDay = date.toLocaleString('en', {weekday: 'long'});  
    var parity;  
    var year = new Date().getFullYear();  
    var month = new Date().getMonth();  
    var today = new Date(year, month, 0).getTime();  
    var now = new Date().getTime();  
    var week = Math.round((now - today) / (1000 * 60 * 60 * 24 * 7));  
    if (week % 2) {  
        parity = 0;  
    } else {  
        parity = 1;  
    }  
    var select = 'SELECT * FROM schedule WHERE ' +  
        '( Room = (SELECT codeHall FROM map.room WHERE room.numberHall=?)' +  
        ' AND Weekday = (SELECT dayCode FROM map.weekday WHERE week-  
day.dayName ' + '=' + nameDay + ' AND weekday.parity = ' + parity + '))';  
    return select;  
}
```

Так как таблица schedule в каждом из своих полей содержит ID на определенный элемент в других таблицах, то нам нужно создать запрос, в котором идентификаторы нужных нам элементов получим, обращаясь к другим таблицам.

После того, как данные подготовлены, можно подключаться к БД. Для этого, нужно создать элемент mysql и создать коннект к нужной нам базе данных через логин и пароль пользователя, у которого есть к ней доступ.

```

var connection = mysql.createConnection({
    host      : 'localhost',
    user      : 'user',
    password  : 'password',
    database  : 'database name'
});
connection.connect();

```

Создав коннект к базе данных, и написав соответствующий запрос, NodeJS позволяет распределить данные относительно их названия по переменным для передачи с помощью JSON, которые с аналогичными названиями будут доступны на клиенте.

```

connection.query('SELECT * FROM map.teacher WHERE teacherCode=?', teacher,
function (err, row) {
    name = row[0].name;
    surname = row[0].surname;
    patronymic = row[0].patronymic;

    res.end(JSON.stringify({
        name: name,
        surname: surname,
        patronymic: patronymic
    }));
});

```

Составив систему запросов, результатом которых будут данные, которые следует отобразить на странице, обернем в различные функции методы получения данных для различных карт:

- getDataFromCorp – для корпусов;
- getDataFromFloor – для этажей.

Аналогично, существуют такие аудитории, информация о которых должна предоставляться независимо от дня недели и она имеет более содержательный характер, нежели остальная. В нашем случае – это деканат, ректорат, канцелярия и т. д. Информация о таких местах меняется крайне редко. Поэтому, рациональнее всего расположить ее в отдельные файлы и создать небольшую иерархию каталогов, корневой у которых будет являться номером данной аудитории.

Возвращаясь к описанию сервера, добавим обработку информации от клиента по пришедшему ID элемента и обращаемся к тому источнику, из которого нужно получить информацию.

```

function processGetData(req, res) {
    res.writeHead(200, headers);
    var id = req.url.split('?')[1].split('=')[1];

```



```

if((id == '2224') || (id == '2201'))
{
    var dean = fs.readFileSync('static/Декан.txt', 'utf8');
    var depDean = fs.readFileSync('static/Заместители.txt', 'utf8');
    var contacts = fs.readFileSync('static/Контакты.txt', 'utf8');
    res.end(JSON.stringify({
        dean: dean,
        depDean: depDean,
        contacts: contacts
    }));
}
else
{
    var nameMap = req.url.split('map')[1].split('=')[1];
    if (nameMap == 'corp')
        sql.getDataFromCorp(id, res);
    else
        sql.getDataFromRoom(id, res);
}
}

```

Чтобы на клиенте, который в нашем случае является интернет-браузером, отправить данные на сервер и получить их в виде JSON - структуры response, нужно воспользоваться объектом XMLHttpRequest.

```

function postHttp(id, map) {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', '/getData?id='+id + '?map='+map, false);

    // Отсылаем запрос
    xhr.send();
    // Если код ответа сервера не 200, то это ошибка
    if (xhr.status != 200) {
        // обработать ошибку
        alert( xhr.status + ': ' + xhr.statusText ); // пример вывода: 404:
Not Found
    } else {
        // вывести результат
        var response = JSON.parse(xhr.responseText); // responseText -- текст
ответа.
        return response;
    }
}

```

3.5 Отображение данных

Отображение данных, полученных от веб - сервера после взаимодействия с базой данных является самой последней частью. Чтобы наиболее компактно и с наибольшей информативностью предоставить информацию, касаемо расписания для конкретной аудитории, воспользуемся созданием модального окна. Внешне оно представляет собой всплывающее окно с данными, во время появления которого фон затемняется, и взаимодействие с другими элементами прекращено до момента закрытия PopUp.

Элементная база модального окна состоит из блокирующего фона и области, в которой будут отображаться данные, переданные в нее.

```
var modalWindow = {
  _block: null,
  _win: null,
  initBlock: function() {
    _block = document.getElementById('overlay');
    if (!_block)
    {
      var parent = document.getElementsByTagName('body')[0];
      var obj = parent.firstChild;
      _block = document.createElement('div');
      parent.insertBefore(_block, obj);
      _block.onclick = function()
      {
        modalWindow.close();
      }
    }
    _block.style.display = 'inline';
  },
  initWin: function(html)
  {
    _win = document.getElementById('popup');
    if (!_win)
    {
      var parent = document.getElementsByTagName('body')[0];
      var obj = parent.firstChild;
      _win = document.createElement('div');
      _win.id = 'popup';
      parent.insertBefore(_win, obj);
    }
    _win.style.display = 'inline';

    //Свойства для появления
    _win.style.webkitTransition = 'all ease .5s';
    _win.style.mozTransition = 'all ease .5s';
    _win.style.oTransition = 'all ease .5s';
    _win.style.transition = 'all ease .5s';
    _win.innerHTML = html;
    _win.style.left = '50%'; //Позиция по горизонтали
    _win.style.top = '50%'; //Позиция по вертикали

    _win.style.marginTop = -(_win.offsetHeight / 2) + 'px';
    _win.style.marginLeft = -100 + 'px';
  },
  close: function()
  {
    document.getElementById('overlay').style.display = 'none';
    document.getElementById('popup').style.display = 'none';
  },
  show: function(html){
    modalWindow.initBlock();
    modalWindow.initWin(html);
  }
};
```

4 Результаты разработки

Для того чтобы убедиться в эффективности и правильности разработанной программы нужно загрузить ее на сервер, запустить и сравнить полученные результаты с ожидаемыми.

4.1 Загрузка на сервер

В результате того, что исполняемый код проекта хранится на внешнем веб-сервисе для хостинга проектов, получить к нему доступ из под любой системы не составит труда.

Выкачав проект нужно разобраться с тем, как его запустить таким образом, чтобы он находился в отдельном процессе долгий промежуток времени.

В комплекте с библиотекой NodeJS поставляется модуль `forever`. Он нужен для того, чтобы запускать любое приложение в виде демона.

4.2 Организация тестирования программы

Для обеспечения правильности работы программы нужно проверить выполнение некоторого набора условий:

- корректная загрузка всех изображений на страницу;
- правильность вывода сцены в окно браузера;
- анимация при наведении на объект;
- отсутствие выделения, а также наведения на текстовые фрагменты;
- отсутствие реакции на фон и на объекты без названия;
- реакция «click» на объект с выводом диалогового окна;
- коннект с базой данных;
- отображение данных в PopUp окне на основной карте;
- корректное вычисление параметров для запроса и отображение данных на карте этажа.

4.3 Испытание программы

Чтобы проверить правильность работы программы, запустим ее на трех разных браузерах и сравним результаты работы. Для примера возьмем Microsoft Edge, Google Chrome, Mozilla Firefox.

1) Microsoft Edge (Рисунок 4)



Рисунок 4 – Отображение карты в Microsoft Edge

Загрузка карты произошла достаточно быстро, все элементы качественно прорисованы за исключением сразу проявившейся ошибки. Все то, что находится выше красной линии, отмеченной на рисунке, не реагирует на наведение мыши. Как выяснилось, данный браузер еще не умеет корректно работать со свойством `display`, которое прописано для каждого `object` на главной странице.

Проверим подключение к базе данных. На рисунке 5 отображена начальная страница, которая содержит простой запрос к одной таблице и вывод данных лишь по ID.

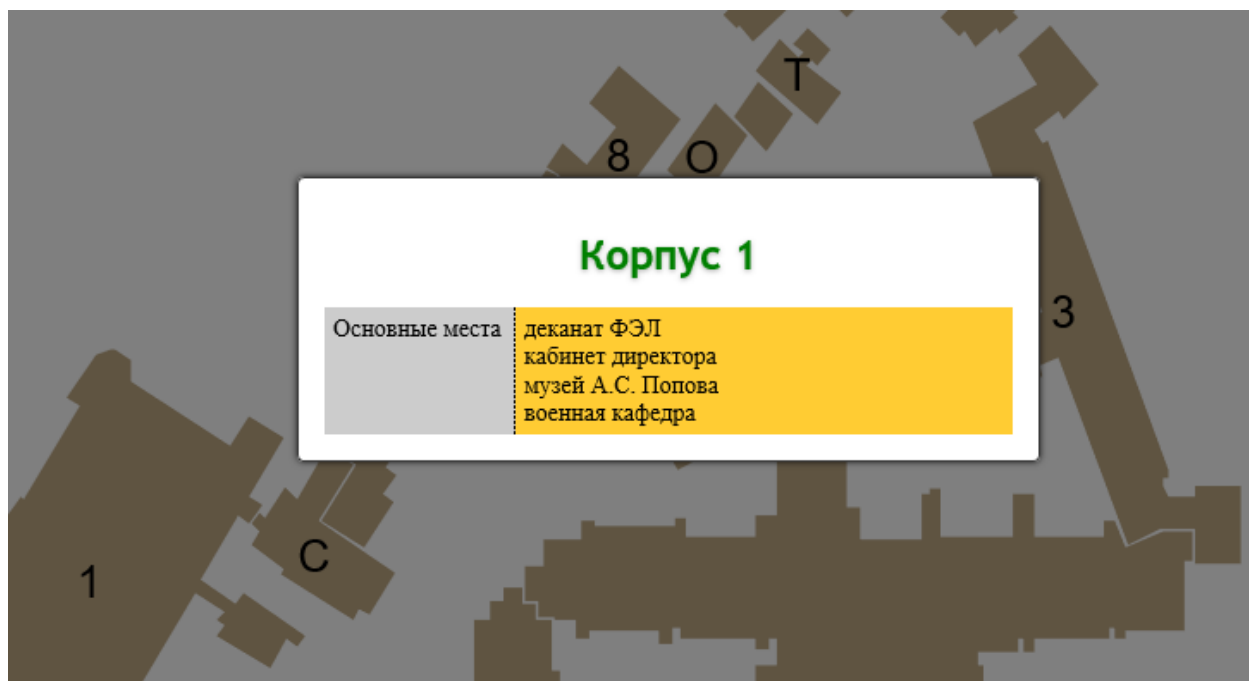


Рисунок 5 – Отображение данных в PopUp окно в ME

Простые запросы и закрытие PopUp окна работают корректно.

На рисунке 6 проверим правильность вычисления данных для сложного запроса, а также отображение данных. Если свериться с базой данных, то для среды, находящейся на черной неделе, работать должно отображение данных на аудитории №2232.

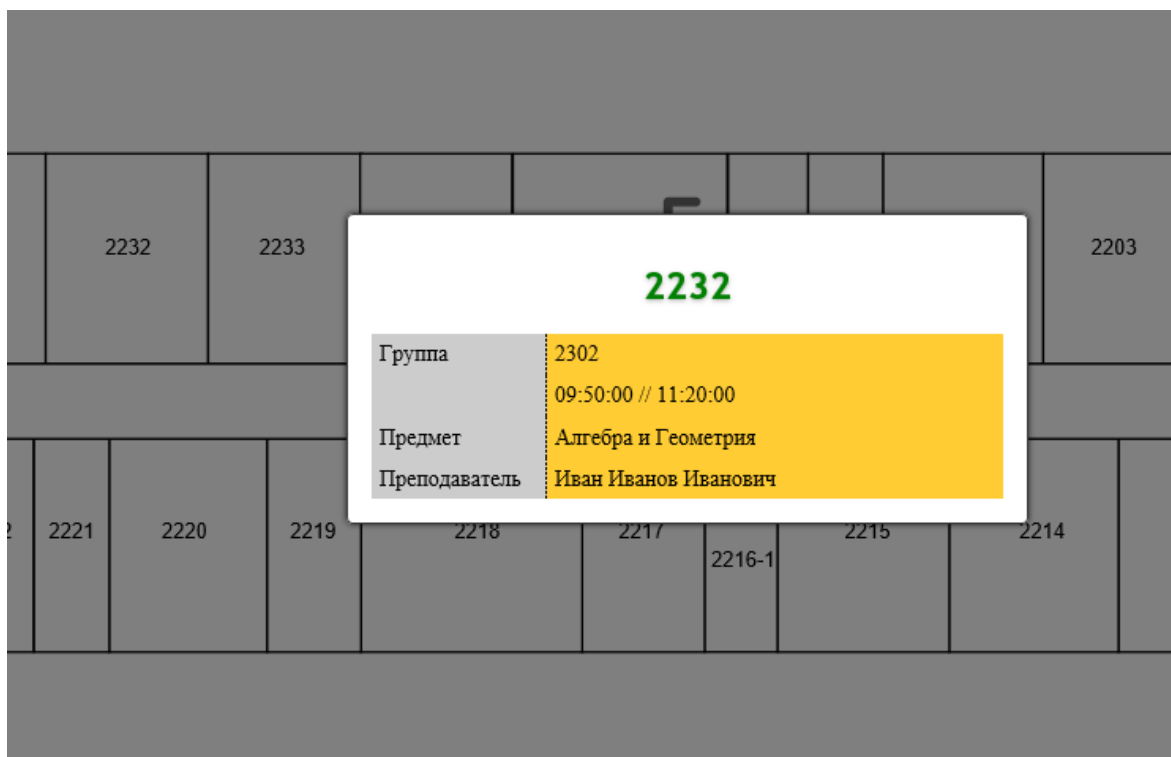
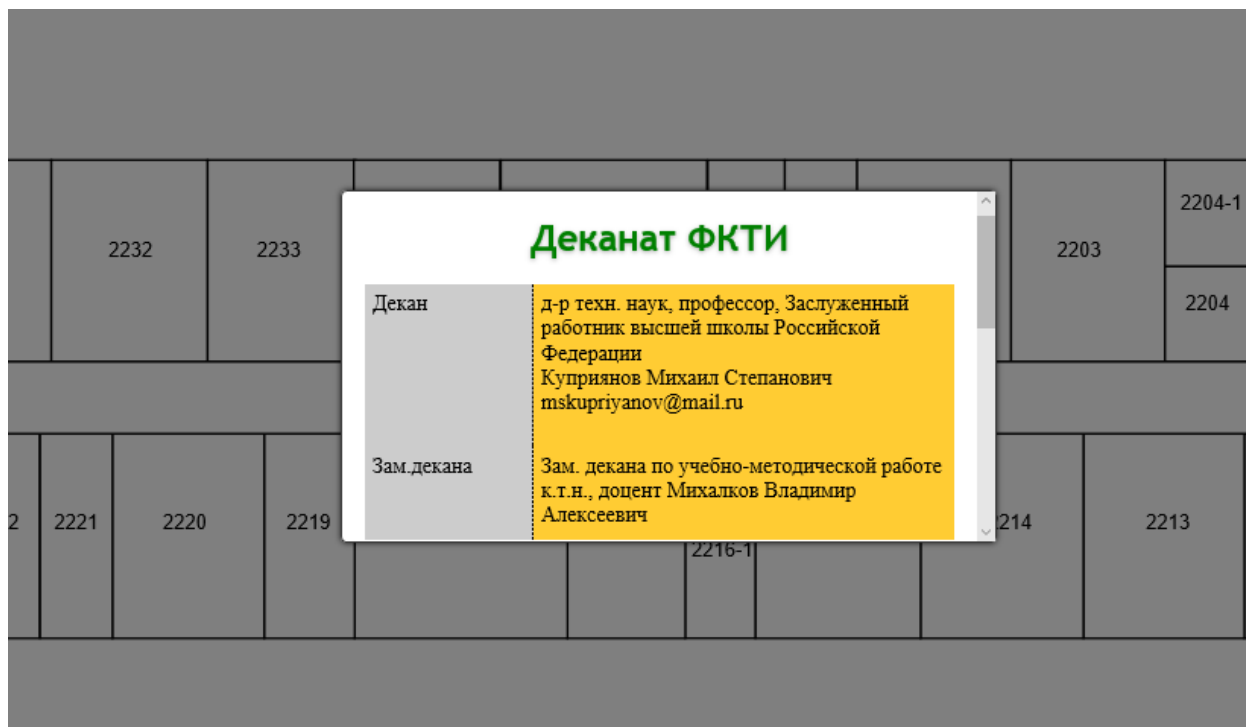


Рисунок 6 – Отображение данных по сложному запросу в ME

Как можно заметить, данные в соответствии с базой данных, отобразились корректно.

На рисунке 7 проверим отображение данных касемо уникальных аудиторий. В нашем случае – деканат. Для примера, возьмем аудиторию №2224, в которой располагается деканат ФКТИ.



	2232	2233					2203	2204-1	
								2204	
2	2221	2220	2219					2214	2213
						2216-1			

Деканат ФКТИ

Декан	д-р техн. наук, профессор, Заслуженный работник высшей школы Российской Федерации Куприянов Михаил Степанович mskupriyanov@mail.ru
Зам.декана	Зам. декана по учебно-методической работе к.т.н., доцент Михалков Владимир Алексеевич

Рисунок 7 – отображение данных по уникальным аудиториям в МЕ

2) Google Chrome и Mozilla Firefox.

Сверяя отображение данных относительно браузеров Google и Mozilla можно сделать вывод, что по сравнению с Edge главная страница работает лучше, анимация на наведения отображается корректно, а также, стоит отметить, что открытие страниц и плавное появление PopUp окна отчетливее выглядит, нежели в других браузерах.

5 Обеспечение качества разработки

С тех пор как программное обеспечение стало неотъемлемой частью нашей повседневной жизни, спрос на него значительно увеличился. Сегодня высокое качество воспринимается как обязательный компонент программного обеспечения.

5.1 Понятие качества и стандарты качества

В современном мире понятию «Качество» как разработчики программного обеспечения, так и пользователи уделяют много внимания, и каждый может истолковать его по-своему. Обратимся к определению «качества ПО» в толковании международных стандартов:

[1061-1998 IEEE Standard for Software Quality Metrics Methodology]

Качество программного обеспечения - это степень, в которой ПО обладает требуемой комбинацией свойств.

[ISO 8402:1994 Quality management and quality assurance]

Качество программного обеспечения - это совокупность характеристик ПО, относящихся к его способности удовлетворять установленные и предполагаемые потребности.

Можно заметить, что оба понятия указывают на то, что программа должна обладать неким набором свойств, которые способны удовлетворять потребности конкретных потребителей.

Учитывая все совокупности характеристик, которые важны для пользователей, можно обеспечить удовлетворение всех сторон (от разработчиков и администраторов систем до конечных пользователей), а также можно гарантировать устойчивое положение на рынке и повышение его развития.

Различные контексты использования, направленность на конкретные устройства, ограничение потребителей программного продукта, позволит разработчику более правильно и точно поставить себе задачи на разработку программного продукта и результатом будет именно то, что соответствовало ожиданиям.

По рисунку 8 можно различить два вида качества: внутреннее качество, то есть качество для разработчиков и тестировщиков программного продукта (удобная архитектура, соответствие требованиям, простота изменений, отслеживание ошибок и доработок), а также внешнее качество, которое подразумевает удобство использования, хорошую производительность и отсутствие ошибок. Но даже это не полностью раскрывает суть «качества ПО».

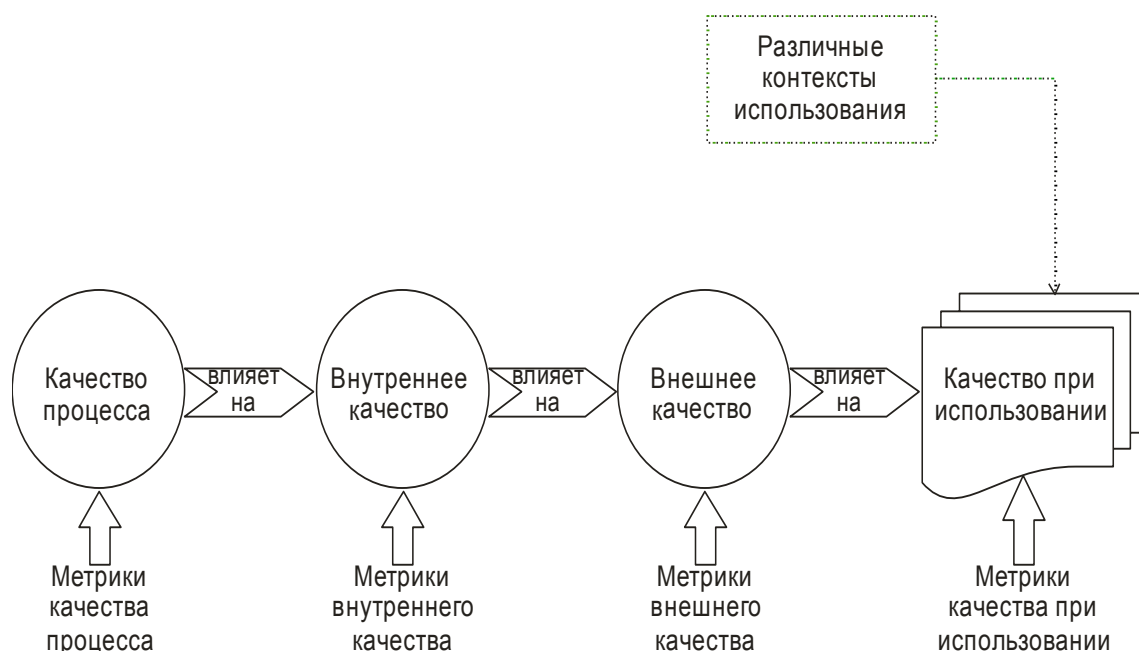


Рисунок 8 – Представление качества

Стоит отметить некоторый набор основных характеристик, которые относятся к определению качества программного продукта:

- **Функциональность** – характеристика, которая соответствует потребностям конкретных пользователей о том, что программа способна решать задачи при заданных наборах начальных условий. Она отвечает за исправную и точную работу, функциональную совместимость и защиту от внешнего доступа тех, кто не является ни разработчиком, ни администратором.
- **Надежность** – характеристика, которая определяет отказоустойчивость программного продукта, а также целостность всей системы.

- Удобство в использовании – определяет легкий и интуитивно понятный для пользователя интерфейс, который без должного обучения позволит разобраться в работе данного продукта.

- Эффективность – обеспечение требуемого уровня производительности в соответствии с обозначенными условиями и выделенными ресурсами.

Чтобы поддерживать характеристики, указанные выше, нужно обозначить методы контроля качества:

- Верификация – соответствие программного продукта установленным требованиям на каждом этапе разработки.

- Валидация – соответствие потребностям и ожиданиям заказчиков, пользователей и других, заинтересованных в проекте сторон.

Эффективность валидации и верификации зависит от корректности и полноты формулировки требований к проекту.

Специалист в области определения качества Джозеф М.Джуран в своих работах часто использует еще одно понятие – «стоимость качества». К нему не относится ни цена продукта на рынке, ни стоимость создания продукта. Данное понятие состоит из:

- Стоимости предотвращения низкого качества разработки, которое включает: review продукта, планирование, оценка процессов, обучение.

- Стоимость проверки, измерения и оценки соответствия стандартам: стоимость тестирования, стоимость review, затраты на выявление и исправление дефектов.

- Стоимость неудач, ошибок, которые обнаружены до и после передачи программного продукта заказчику: стоимость переработок, повторное тестирование, обработка жалоб заказчика.

На рисунке 9 графически представлено то, от чего и как зависит стоимость качества.



Рисунок 9 – Стоимость качества

Разобравшись с требованиями, ресурсами и запросами заказчика, а также потребителей данного ПО, стоит уделить внимание повышению качества разработки с приведением рекомендаций и предложений.

Чтобы разработать приложения высокого качества, нужно использовать такие среды разработки, которые обладают инструментами, с помощью которых разработчик может создавать качественный и понятный код, проводить тесты, чтобы определить уровень качества.

Не исключено, что данные среды обладают инструментами, библиотеками, которые поддерживают сбор и обработку данных, касаемых программы и её работы, управление изменениями, контроль версий, управлением конфигурации, дефектами, рисками и различными видами взаимодействиями. В современном мире существует достаточное количество пакетов, которые содержат в своем ассортименте технологии обеспечения качества разработки как в практическом, так и в теоретических подходах.

Первоначально, разработчику стоит уделить внимание тестированию различных аспектов своего программного продукта и выявление ошибок, связанных с критическими ситуациями, неправильными действиями. Существует несколько уровней тестирования:

Модульное тестирование (Unit-testing) – уровень, при котором минимально возможный компонент, на который разбита программа (класс или функция) тестируется на правильность выполнения при различных входных данных. В этом случае на вход подается информация, удовлетворяющая критериям корректности, отслеживается реакция и сравнивается полученный результат с ожидаемым. Недостаток такого тестирования заключается в том, что применять его можно лишь в том случае, когда проверяемый элемент программы уже разработан. Часто используют обратный подход – подготовка тестов заранее, на основе требований, когда исходного кода еще нет. Данный уровень является важной частью для отладки написанного кода.

Интеграционное тестирование (Integration testing) – уровень, при котором в качестве входных данных используются минимальные компоненты программы, группируются в более крупные множества и выполняется проверка их совместной работы при обмене данными. Данный уровень выполняется разработчиками на более позднем этапе разработки.

Системное тестирование (System testing) – уровень, при котором разработчиком выполняется имитация того, как пользователь будет использовать программу. Проверка интерфейса выполняется с целью оценки системы тем требованиям, которые были описаны в самом начале разработки программы. В основном используется два направления тестирования: графический пользовательский интерфейс (GUI) и интерфейс Web-приложения (WebUI).

Приемочное тестирование (Acceptance testing) – уровень, при котором выпускается первоначальная версия продукта и её испытывают заинтересованные пользователи в реальном окружении, выявляют ошибки, которые не появились на предыдущих этапах и сообщают разработчикам для последующей доработки. Последующие поиски ошибок, багов или неисправностей не повысит качества программы. К тому же, не все найденные ошибки будут исправлены к срокам сдачи и с некоторыми, не особо значи-

мыми ошибками, существенно не влияющие на работу, программа уйдет к пользователям или заказчику.

Чтобы ответить на вопрос о том, готова ли программа для сдачи, используя результаты тестирования, стоит применить закон убывающей предельной полезности. При разработке программного продукта может возникнуть момент, когда достигается максимум прибыли от реализации программы. После данного момента, какие бы не были улучшения или доработки, они не повлияют на уровень качества.

Кроме тестирования стоит упомянуть и дополнение платформ для использования. С увеличением количества систем, на которых пользователи могли бы использовать данный программный продукт, возросла потребность разработчиков адаптировать ПО под конкретную ОС. В большинстве случаев, доработка программного продукта заключается в использовании аналогичных функций и методов, но в библиотеках, которые доступны изначально в конкретной системе. Бывают ситуации, когда, разработав ПО, доработка не нужна. В особенности это относится к веб-приложениям.

5.2 Оценка качества разработки интерактивной карты

Интерактивная карта университета разрабатывается в качестве html страницы для студентов вуза с целью упрощения и более удобного отображения данных, касаемо учебного заведения.

Разработка карты в виде интернет-страницы позволяет осуществить кроссплатформенность гарантируя то, что пользователи смогут её использовать на большинстве современных устройств. Удобство в использовании и гарантия того, что данные отображаются корректно и без задержек повышает качество проекта.

Тестирование интерактивной карты можно осуществить в 2 подхода:

- 1) Написание модульных тестов для проверки клиент-серверного взаимодействия, доступа к базе данных и правильности отображения информации в PopUp окно.

- 2) Системное тестирование для определения корректности анимаций, переходов и откликов на нажатие мыши или тап по экрану.

При успешных прохождении тестов стоит задуматься об улучшении элементов интерфейса, которые позволят улучшить визуальную составляющую и способ доступа к элементам карты, а также улучшить методы обращения «клиент-сервер» направленных на уменьшение скорости отображения данных, что очень повлияет на качество.

Стоит отметить, что к мероприятиям по усовершенствованию ПО можно включить разработку под конкретную операционную систему. Все это обосновывается удобством в использовании карты на мобильных устройствах, а также увеличение функции, которые может предоставить программа для пользователя по сравнению с интернет-страницей, у которой возможности ограничены.

5.3 Вывод

Проанализировав некоторые способы контроля и повышения качества продукта, можно прийти к выводу, что для улучшения ПО следует как можно чаще использовать тесты, чтобы удостовериться в правильности работы, а также при нахождении недоработок или ошибок, своевременно их поправлять. А также, желательно, не останавливаться на достигнутом и обновлять проект до совместимости с конкретной операционной системой.

ЗАКЛЮЧЕНИЕ

В ходе исследования различных технологий для разработки web приложений был выбран наиболее подходящий фреймворк, а также был использован инструмент для создания удобной связи клиент-сервер.

В ходе дипломной работы была создана программа – сайт, с подключенной к ней базой данных.

В качестве входных данных для формирования программы являются svg - изображения, а также база данных MySQL, информация из которой в следствии будет отображаться в сплывающих окнах.

Для эффективного взаимодействия пользователя с программой разработан интуитивно понятный интерфейс, позволяющий выполнять обращение к отдельным элементам корпусов и оптимально выводить информацию относительно каждого браузера. Для простоты освоения и удобства работы с программой были использованы стандартные средства организации пользовательского интерфейса, диалоговые окна.

Автором настоящего проекта был проведён краткий анализ использованных средств веб - программирования: среды разработки JavaScript и HTML5, а также технологии NodeJS, JSON, CSS, MySQL. Была построена архитектура программной системы, а затем спроектирована и реализована ER - диаграмма с детальным описанием ключевых объектов.

Исходный код программы является открытым, и в дальнейшем возможно расширение её функциональных возможностей. Основные направления совершенствования программы это:

- дополнение информации в базу данных, для более подробного формирования пользователя;
- переход программы дополнительно на определенные специальные операционные системы;
- добавление поиска и прокладывания маршрута между объектами.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Коллективный блог Хабрахабр. Описание способов создания интерактивных карт. – <https://habrahabr.ru/company/airbnb/blog/266575/>
- 2) Видеоуроки по созданию сайтов. Рассмотрение подключения JavaScript. – <http://ruseller.com/lessons.php?id=952>
- 3) Коллективный блог Хабрахабр. Статья взаимодействия SVG и HTML. – <https://habrahabr.ru/post/127994/>
- 4) Блог дизайна и разработки. Реализация взаимодействия с SVG. – <http://followdesign.com/development/2011/11/27/klikabelnye-javascript-karty-evropy.html>
- 5) Фирма Mozilla. Mozilla Developer Network = Сеть разработчика компании Mozilla. Документация по SVG . – <https://developer.mozilla.org/ru/docs/Web/SVG>
- 6) Платформа NodaJS. Описание и документация. – <http://nodejs.ru/>
- 7) Современный учебник по JavaScript. Стринкаст по NodeJS. – <https://learn.javascript.ru/screencast/nodejs>
- 8) Подключение к базе данных с помощью NodeJS. – <https://github.com/nodejs/node>
- 9) Фирма 2GIS. Интерактивная карта 2ГИС. – <https://2gis.ru>
- 10) Фирма Google. Google Inc Corporation. Интерактивная карта Google Maps. – <https://maps.google.ru>
- 11) Сетевая энциклопедия Wikipedia = Википедия. Описания графического формата SVG. – <https://ru.wikipedia.org/wiki/SVG>