Содержание

[Введение 4](#_Toc121737701)

[Основная часть 5](#_Toc121737702)

[Локальное приложение 5](#_Toc121737703)

[Клиентское приложение 8](#_Toc121737704)

[Проектирование процессов работы приложения 9](#_Toc121737705)

[Выбор архитектурных подходов и технологий 10](#_Toc121737706)

[Процесс разработки клиентского приложения 13](#_Toc121737707)

[Заключение 18](#_Toc121737708)

[Литература 19](#_Toc121737709)

# Введение

Развитие компьютерной техники происходит фантастическими темпами. Изначально создаваемая исключительно для математических расчетов, ЭВМ постепенно нашла применение во всех сферах человеческой деятельности, включая и экономику. На современном этапе развития вычислительной техники с помощью персонального компьютера решаются практически любые производственные задачи с применением автоматизированных информационных систем.

Производственная практика — практическая часть учебного процесса подготовки квалифицированных рабочих и специалистов, проходящая, как правило, на различных предприятиях в условиях реального производства. Является заключительной частью учебной практики, проходящей в учебном заведении. Во время производственной практики происходит закрепление и конкретизация результатов теоретического учебно-практического обучения, приобретение студентами умения и навыков практической работы по присваиваемой квалификации и избранной специальности или профессии.

Целью данной производственной практики является закрепление полученных навыков разработки АИС, составление планов, проектов разработки АИС, ведение отчетной и технической документации.

Тема производственной практики: «Участие в разработке информационных систем».

Производственная практика пройдена в ПАО «Газпром газораспределение Уфа» в должности front-end разработчика.

# Основная часть

## Локальное приложение

Локальное приложение - это программное обеспечение, специально разработанное под конкретную платформу и использующее только ресурсы вычислительной техники, на которой она установлена.

В ходе производственной практики было разработано локальное приложение для интерактивного информационного стенда на базе операционной системы Microsoft Windows 7 данной организации, которое запускается в среде браузера Chrome.

Информационным стендом, или киоском, называют мобильный терминал с сенсорным экраном. Он транслирует динамически меняющиеся сведения и может иметь внешнее управление. Прикосновением к экрану или только голосом пользователь может запросить на инфостенде нужную информацию.

Языки и технологии использованные в ходе разработки:

* HTML;
* CSS;
* JavaScript;
* JQuery библиотека, написанная на языке JavaScript;
* Fancybox библиотека, написанная на языке JavaScript.

Локальное приложение включает в себя:

* Схему проезда до единого центра обслуживания клиентов данной организации;
* Порядок предоставления услуг;
* Образцы заявлений для заполнения;
* Адреса всех участков предприятия;
* Внутренний телефонный справочник здания.

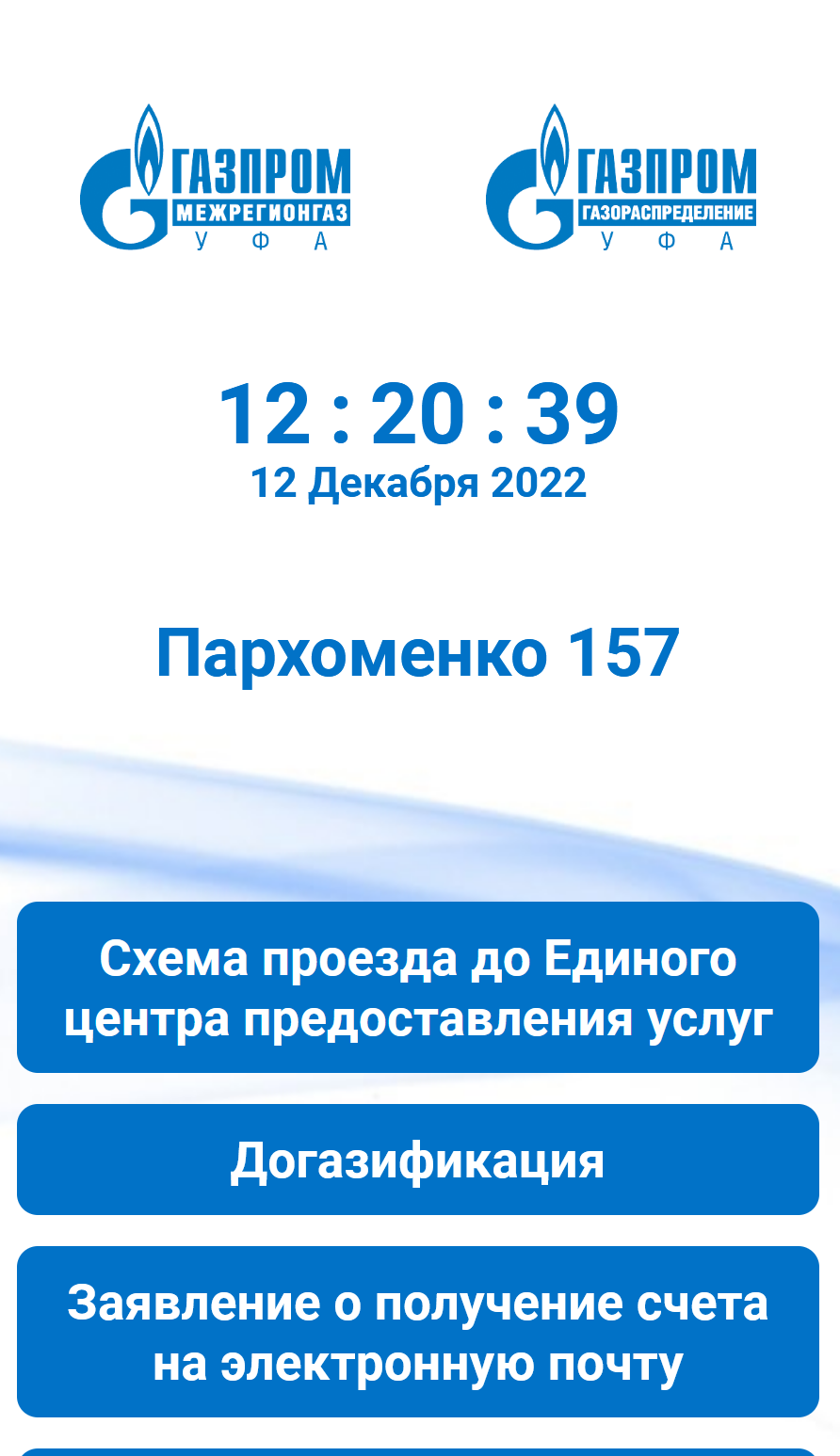


Рисунок 1 – Внешний вид локального приложения

Поскольку интерактивный информационный стенд имеет вытянутый экран со специфичным разрешением экрана, внешний вид приложение адаптирован под данные характеристики дисплея.

Процесс создания локального приложения, исполняемого в среде браузера, выглядит следующим образом:

* Построение каркаса графического интерфейса, верстка с помощью языка HTML;
* В целях улучшить качество графического интерфейса и его восприятия применяются каскадные таблицы стилей, язык CSS;
* Динамика и функционал приложения обеспечивается языком JavaScript и вспомогательными библиотеками.

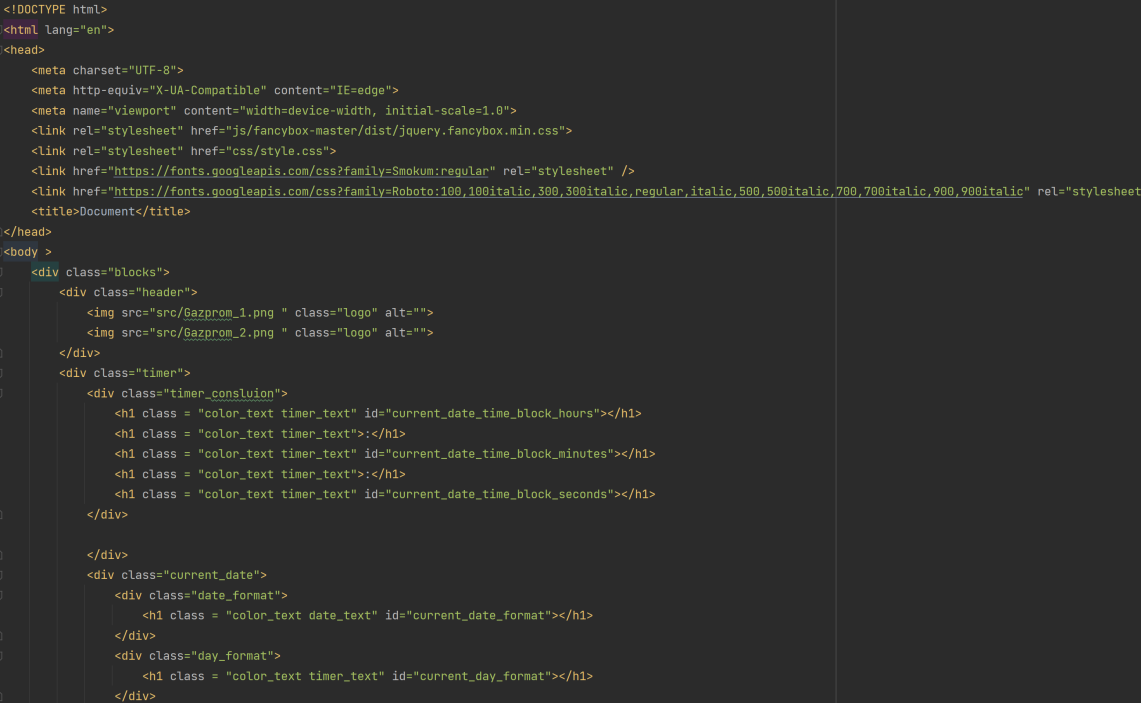


Рисунок 2 – HTML вёрстка

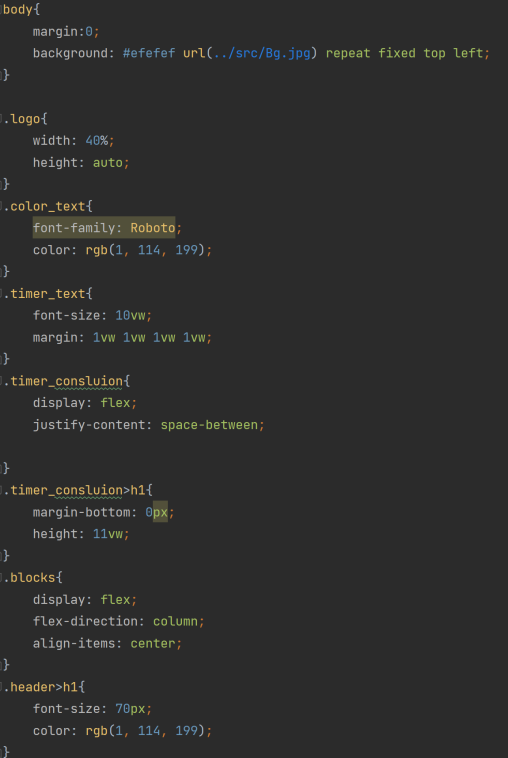


Рисунок 3 – стилизация графического интерфейса

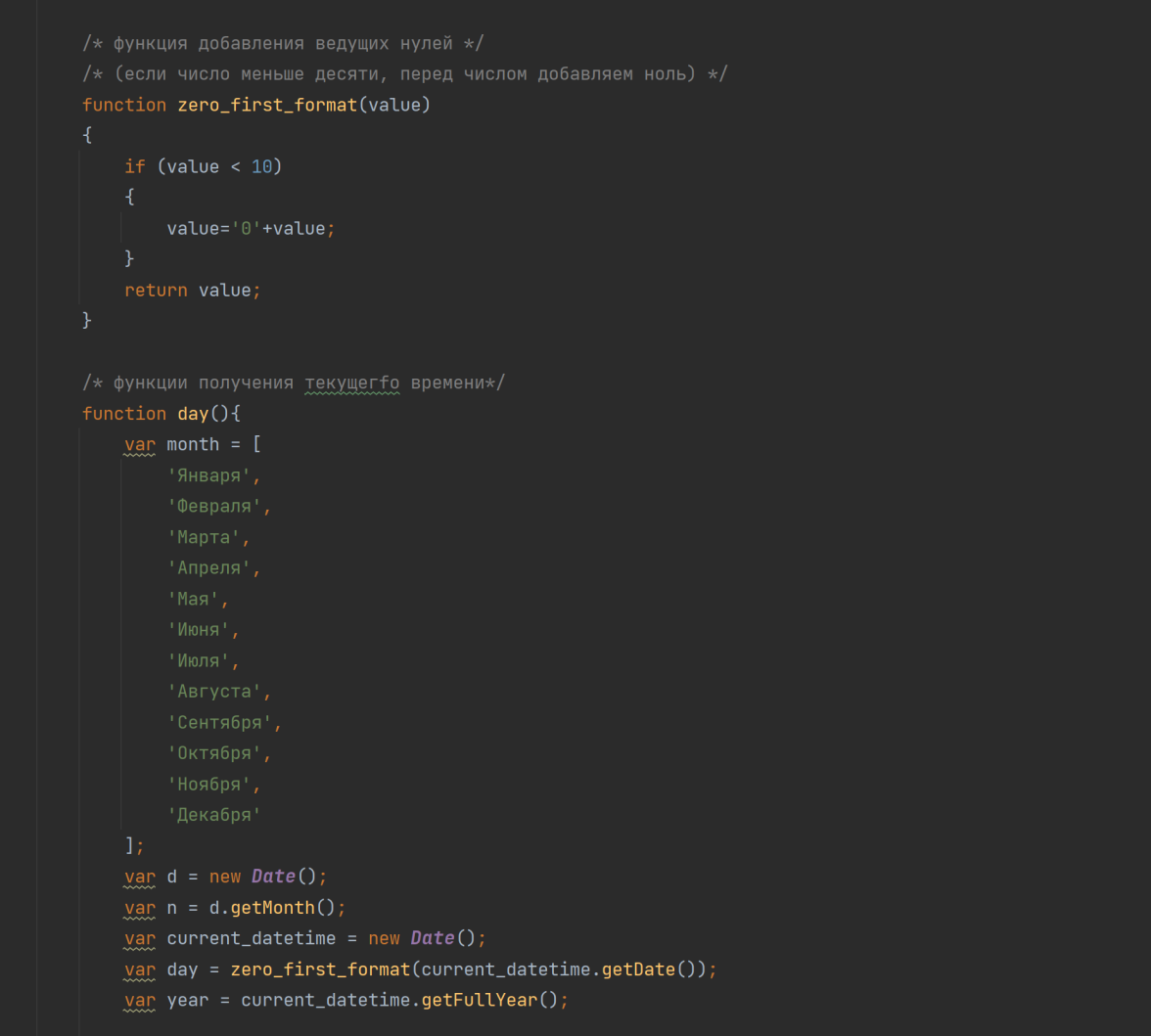


Рисунок 4 – программирование часов на языке JavaScript

## Клиентское приложения

Также были изучены вопросы разработки клиентских приложений. Более подробно акцентировано внимание на разработку клиентских приложений

Клиентской называется часть приложения, с которой напрямую взаимодействует конечный пользователь. Это может быть либо приобретенное компанией серийное коммерческое программное обеспечение, либо прикладная программа, разработанная внутри компании с помощью инструментальных средств третьих фирм.

Веб-приложение - клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются межплатформенными службами.

Была выполнена разработка клиентской части веб-приложения «Расписание/График». Данное приложение предоставляет возможность составлять график работы и дел сотрудников.

## Проектирование процессов работы приложения

В диаграмме IDEF0 (Рисунок 5) показан весь процесс составления расписания на текущий момент, а именно:

* Проверка свободности рабочей группы;
* Сверка нагрузки часов работников;
* Проверка свободности рабочей кабинета.

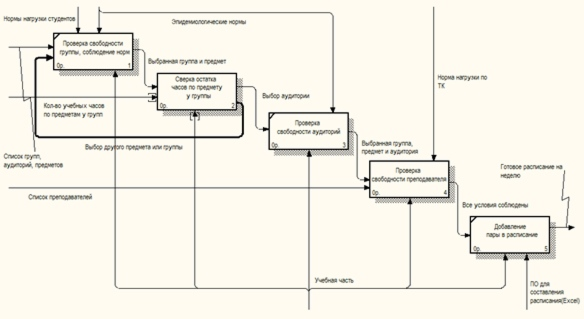


Рисунок 5 – Диаграмма в нотации IDEF0

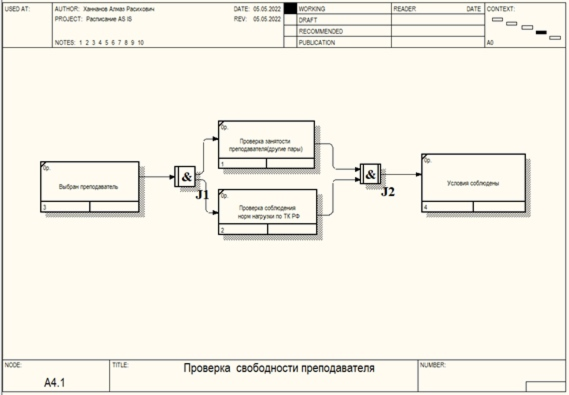


Рисунок 6 – Декомпозиции блока «Проверка свободности работников» в нотации IDEF3

На данной диаграмме(Рисунок 6) показано, как происходит проверка занятости работников другими делами, а также происходит проверка соблюдения норм нагрузки по Трудовому Кодексу РФ. Важно обратить внимание на перекрестки стрелок. Они типа «И», значит что должны соблюдаться оба условия, чтобы пройти на следующий этап.

Резюмируя, устаревший принцип тянет за собой множество проблем, которые требуют много времени и действий, как для составления графика, корректировки, так и для обычного просмотра. Возможности ЭВМ и АИС очень помогают ускорить любую деятельность, поэтому следует развиваться с ними и разрабатывать новое ПО.

## Выбор архитектурных подходов и технологий

Важным этапом в процессе разработки клиентской части является выбор ее архитектуры. Далее рассмотрим существующие варианты архитектур, на основе чего сделаем свой выбор.

Существует 2 вида архитектуры веб-приложений:

* SPA (Single Page Application) – одностраничное приложение;
* MPA (Multi Page Application) – многостраничное приложение.

Значение термина SPA (Single Page Application) кроется внутри него самого. SPA — это одностраничное приложение, содержащее HTML-страницу, которая динамически (без полной перезагрузки) обновляется в ответ на действия пользователя. Архитектура приложения устроена так, что при первоначальном запуске посетитель видит основной контент сайта в браузере, а новые данные загружаются на ходу по мере необходимости, например, при прокрутке или клике на иконку. Если вы когда-нибудь листали ленту Facebook, Twitter или Вконтакте, то вы понимаете, о чём идёт речь. Такое приложение при первой загрузке сразу отдаёт весь его внешний вид и клиентскую логику, что с одной стороны вызывает долгую первичную загрузку, но с другой стороны позволяет более не перезагружать страницу, а только лишь обновлять данные, которые приходят по запросу к серверу, используя возможности клиентской логики (JavaScript) и рендер графической части происходит прямо в клиентской части в браузере при помощи JavaScript, это называется CSR.

MPA или Multi Page Applications по принципу работы полностью противоположны SPA. MPA — это многостраничные приложения, работающие, как привычные нам веб-сайты. Они отправляют запрос на сервер и полностью обновляют страницу, когда с ней совершается какое-либо действие (переход на другую страницу, внесение и изменение данных). Подобная архитектура приложения значительно влияет на скорость и производительность, поскольку большая часть данных подгружается повторно при каждом переходе, в том числе и полностью весь HTML документ (SSR).

Более разумным решением стал выбор архитектуры SPA и вместе с ним идет выбор подхода REST API по нескольким причинам:

* Быстрая работа без перезагрузки страницы;
* Передача с клиента на сервер и обратно только чистых данных, исключая HTML, CSS и JavaScript;

Один из самых популярных фреймворков который диктует SPA архитектуру это Vue, конечно же еще существует REACT и ANGULAR, но REACT не подходит к этому проекту поскольку не реализует реактивность, а ANGULAR в свою очередь не показывает себя полностью в таком маленьком проекте.

Vue - это **прогрессивный фреймворк** для создания пользовательских интерфейсов. Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications).

Для того чтобы полностью использовать возможности Vue нужно воспользоваться дополнительными инструментами и библиотеками, таковыми являются Vuex, bootstrap.

Vuex - это библиотека управления состоянием, специально предназначенная для создания сложных крупномасштабных приложений Vue.js. Она использует глобальное централизованное хранилище для всех компонентов в приложении, используя преимущества собственной системы реактивности для мгновенных обновлений.

По сути, состояние Vuex можно рассматривать как единственный источник истины для всего приложения.

Bootstrap в свою очередь является очень полезной библиотекой стилей который используется для быстрой верстки адаптивных дизайнов сайтов и веб - приложений

Итоги выбора архитектуры и технологий:

* Была выбрана архитектура приложения SPA;

Были выбрана библиотеки такие как Vue и Vuex и дополнительный инструмент Bootstrap.

## Процесс разработки клиентского приложение

Этапы разработки:

* Загрузка библиотек, фреймворков и начальная настройка.
* Разметка HTML
* Корректировка стилей и классов.
* Разработка логики работы с данными с сервера «Vuex»
* Разработка логики компонентов «Vue»
* Корректировка адаптивности дизайна сайта.

1).Загрузка библиотек, фреймворков и начальная настройка.

Для того чтобы установить все нужные пакеты, необходимо обратиться к менеджеру пакетов (npm), Строкой в терминале «npm install Vue» мы начинаем скачивать и устанавливать библиотеку «Vue» (Рисунок 7)

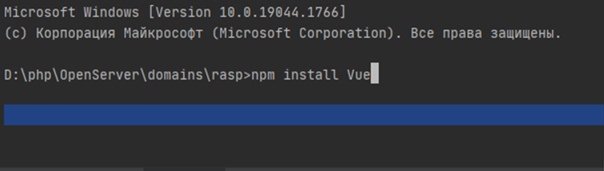


Рисунок 7 - Установка Vue

Точно таким же способом мы устанавливаем «Vuex» и «Bootstrap»

Начальная настройка заключается в том чтобы привязать все библиотеки в главном файле приложения (Рисунок 8)

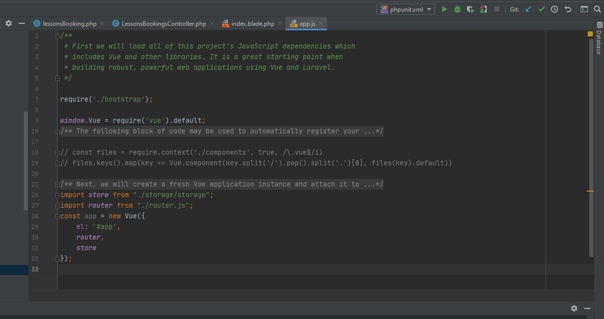


Рисунок 8 - Настройка и подключение всех библиотек.

2).Разметка HTML

Разметка HTML в Vue происходит внутри его компонентов, тега template(Рисунок 9)

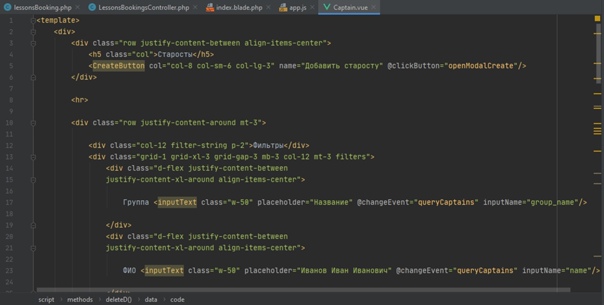


Рисунок 9 - Разметка HTML на Vue

3).Корректировка стилей и классов.

Не все стили от Bootstrap могут подойти поэтому в редких случаях их необходимо писать самому. (Рисунок 10)

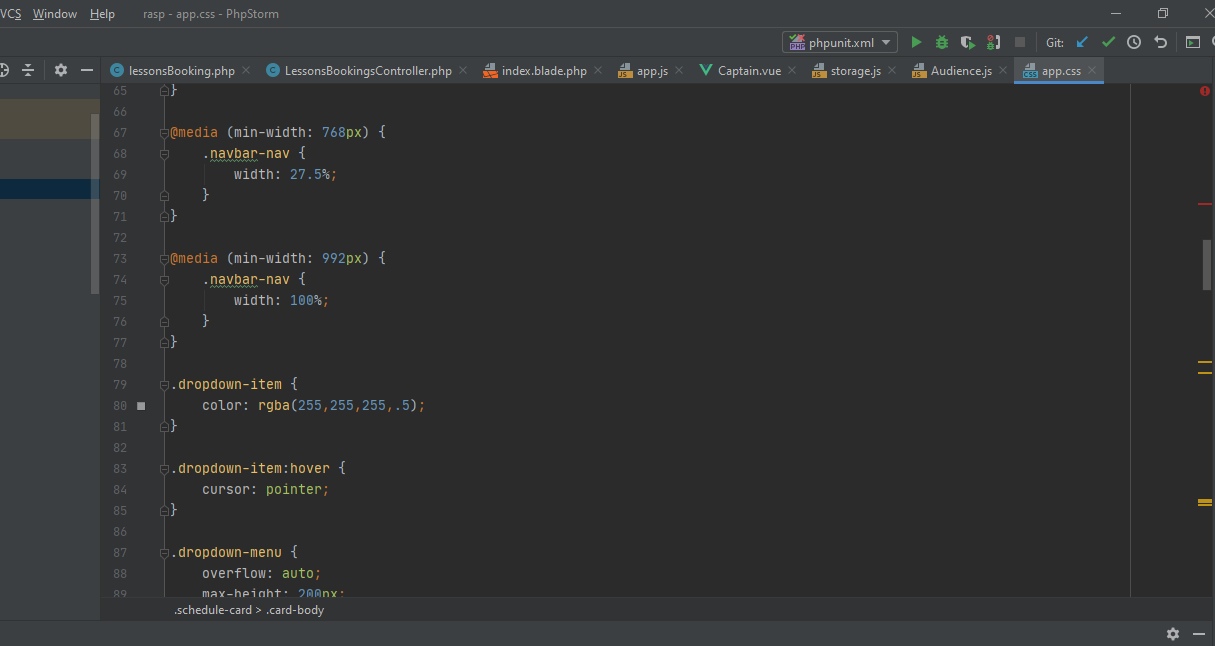


Рисунок 10 - Корректировка стилей

4). Разработка логики работы с данными «Vuex»

Логика на Vuex представляет из себя создания хранилищ для данных что в свою очередь представляет из себя различного рода переменные которые хранят в себе данные для реактивного обновления, а также действия над ними и мутации

Действия это асинхронные события а мутации синхронные события

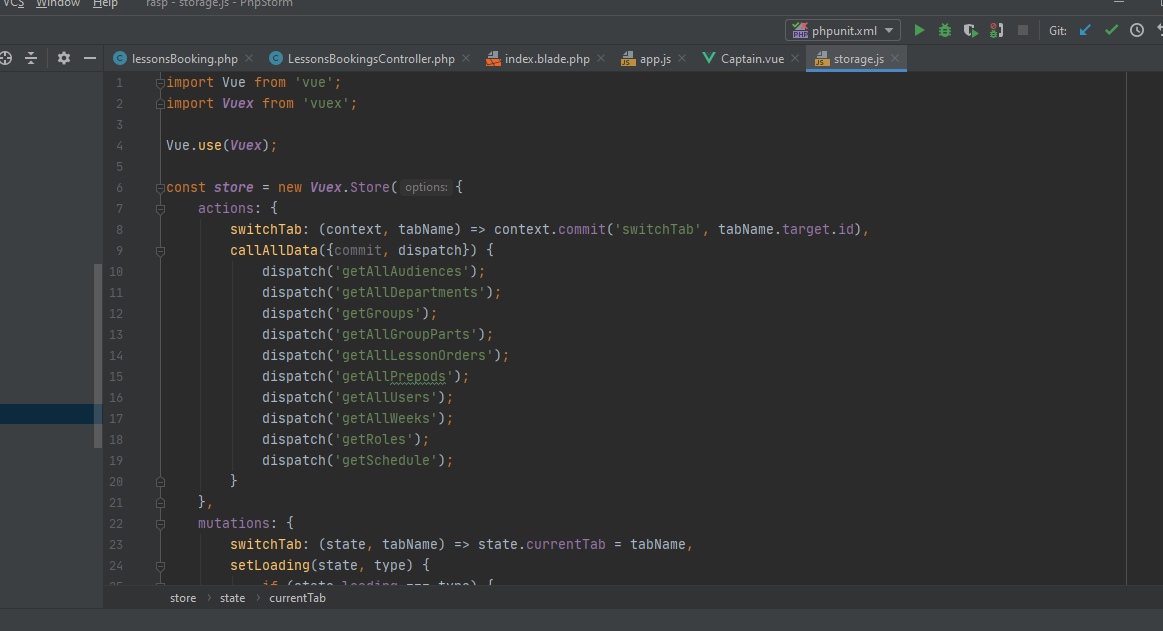


Рисунок 11 - Разработка логики с данными

5).Разработка логики компонентов.

Vue диктует принцип декомпозиции, а это значит что слишком большие файлы следует делить на более маленькие компоненты и из за этого весь исходный код на Vue состоит из множества маленьких компонентов.

А компоненты в свою очередь состоят из 3 частей:

* HTML разметка
* Логическая часть, а то есть скрипты
* Стили

Но стили которые прописываются внутри компонентов мы не будем использовать поскольку гораздо удобное воспользоваться готовыми стилями либо создать свои в отдельном CSS файле.

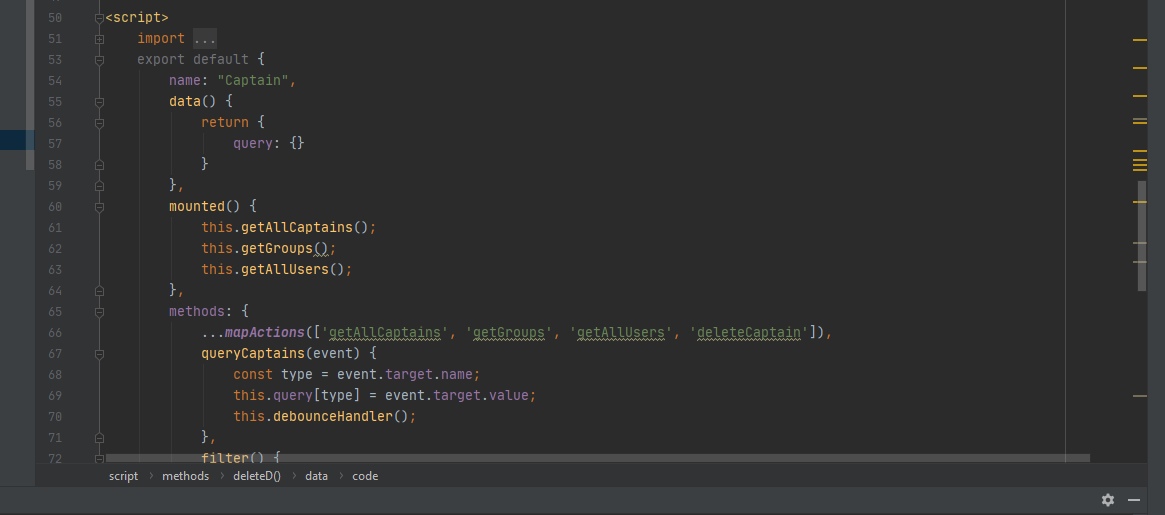


Рисунок 12 – Разработка логики с компонентами

6) Корректировка адаптивности дизайна сайта

Адаптивность — это способность сайта «подстраиваться» под различные технические условия (а именно, под размеры экрана пользовательского устройства. Причем качество отображения не зависит ни от диагонали, ни от позиционирования экрана.

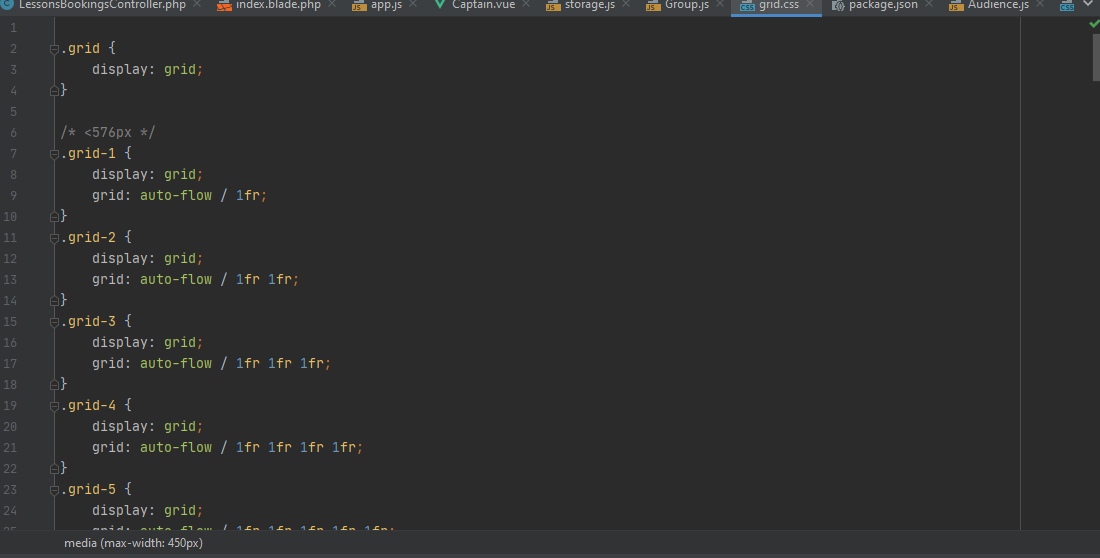


Рисунок 13 – Библиотека стилей для адаптации

Цель адаптивной верстки — добиться того, чтобы сайт оставался удобным и обеспечивал конверсию при загрузке на разных устройствах.

Для адаптивности была написана собственная небольшая библиотека стилей основанная на технологии Grid CSS

После проведения всех работ по разработки клиентской части мы получили новый современный дизайн, темная тема которая не будет слепить глаза, только нужная информация

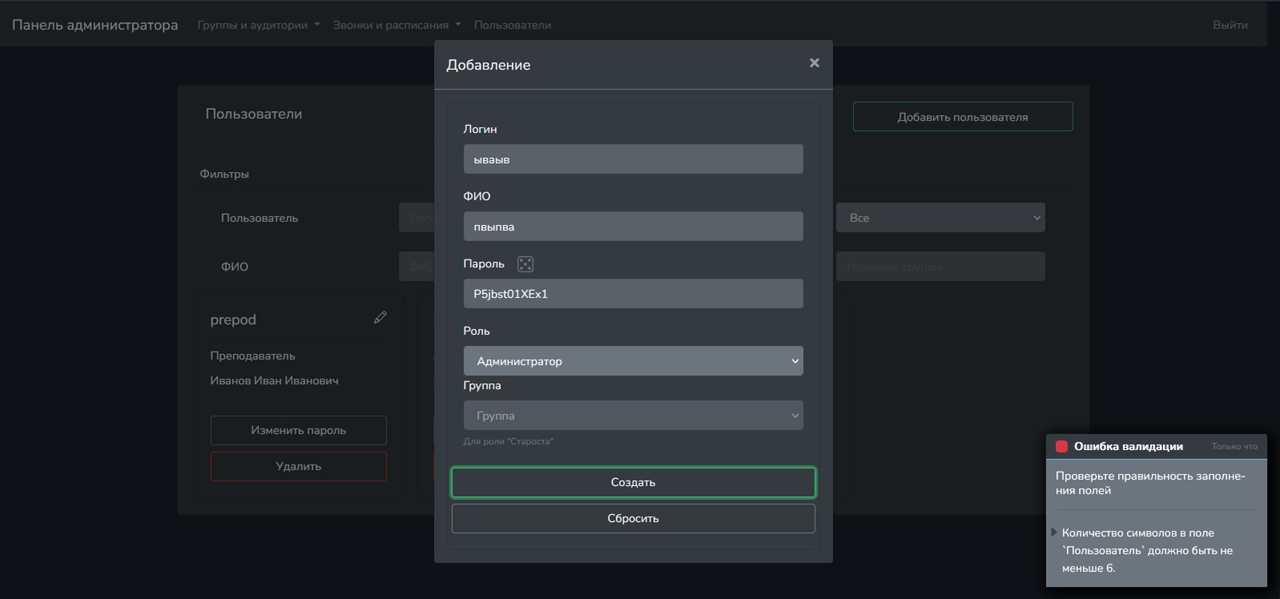


Рисунок 15 – Результаты разработки пользовательского интерфейса

После проведения всех частей клиентской части были проведено тестирования

Тестирование были проведены несколькими методами:

* В первую очередь были написаны Unit тесты, которые проверяют правильность работы информационной системы
* Тест на проверку адаптивности экрана на разных устройствах.

# Заключение

В ходе производственной практики была разработана:

* Локальное приложение для информационного стенда в браузерном окружении;
* Техническая документация по модификации локального приложения;
* Клиентская часть веб-приложения «Расписание и график», на языках CSS ,JavaScript, Html.

Диаграмма процессов работы веб-приложения «Расписание и график»

При разработке локального приложения были применены языки CSS ,JavaScript, Html.

Так же использованы библиотеки написанные на языке JavaScript

* JQuery;
* Facybox.

При разработке клиентской части веб-приложения были применены такие языки и фреймворки как

* JavaScript;
* Html;
* Css;
* Vue;
* VueX.

Был применен шаблон проектирования MVVM. После окончания разработки были проведены испытания и тестирования информационной системы и введение в действие.

# Литература

1. Информационный портал об образовании в России [Электронный ресурс]. Режим доступа: URL: <https://edugid.ru/spravochnik-abiturientov/507-chto-takoe-praktika-v-vuze-ili-kolledzhe-vidy-praktiki> (Дата обращения: 15.12.2022)
2. Википедия [Электронный ресурс].Режим доступа: URL: <https://ru.wikipedia.org/wiki> (Дата обращения: 15.12.2022)
3. Хабр [Электронный ресурс]. Режим доступа: URL: <https://habr.com/ru/all/> (Дата обращения: 15.12.2022)
4. Блог Кинзябулатов Рамиля [Электронный ресурс]. Режим доступа: URL: <https://trinion.org/blog/idef0-znakomstvo-s-notaciey-i-primer-ispolzovaniya> (Дата обращения: 15.12.2022)
5. Корпоративный менеджмент [Электронный ресурс]. Режим доступа: URL: <https://www.cfin.ru/vernikov/idef/idef3.shtml> (Дата обращения: 15.12.2022)
6. Базовые стили CSS [Электронный ресурс] Режим доступа: URL: <https://myrusakov.ru/kak-zadat-css-stil.html> (Дата обращения: 15.12.2022)
7. Основы работы с Vue [Электронный ресурс] Режим доступа: URL: https://css-live.ru/articles/kak-sozdavat-vue-komponenty-slovno-profi.html (Дата обращения: 15.12.2022)