содержание

[Введение 4](#_Toc122182689)

[Основная часть 5](#_Toc122182690)

[Локальное приложение 5](#_Toc122182691)

[Серверное приложение 8](#_Toc122182692)

[Выбор архитектурных подходов и технологий 9](#_Toc122182693)

[Процесс разработки серверного приложения 11](#_Toc122182694)

[Оптимизированные процессы составления графика работы 15](#_Toc122182695)

[заключение 16](#_Toc122182696)

[Литература 17](#_Toc122182697)

# 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 back-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.

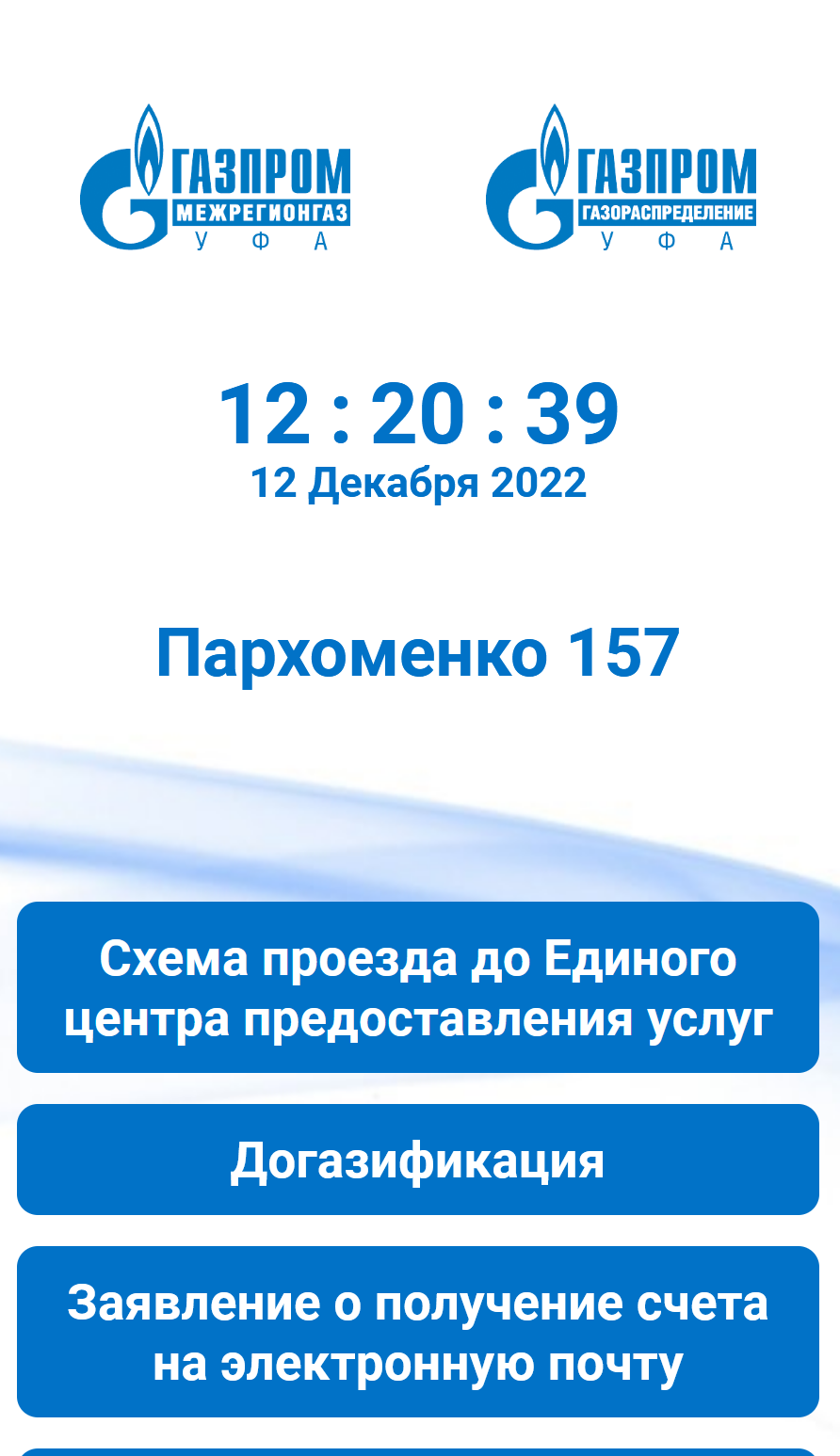


Рисунок 1 – Внешний вид локального приложения

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.

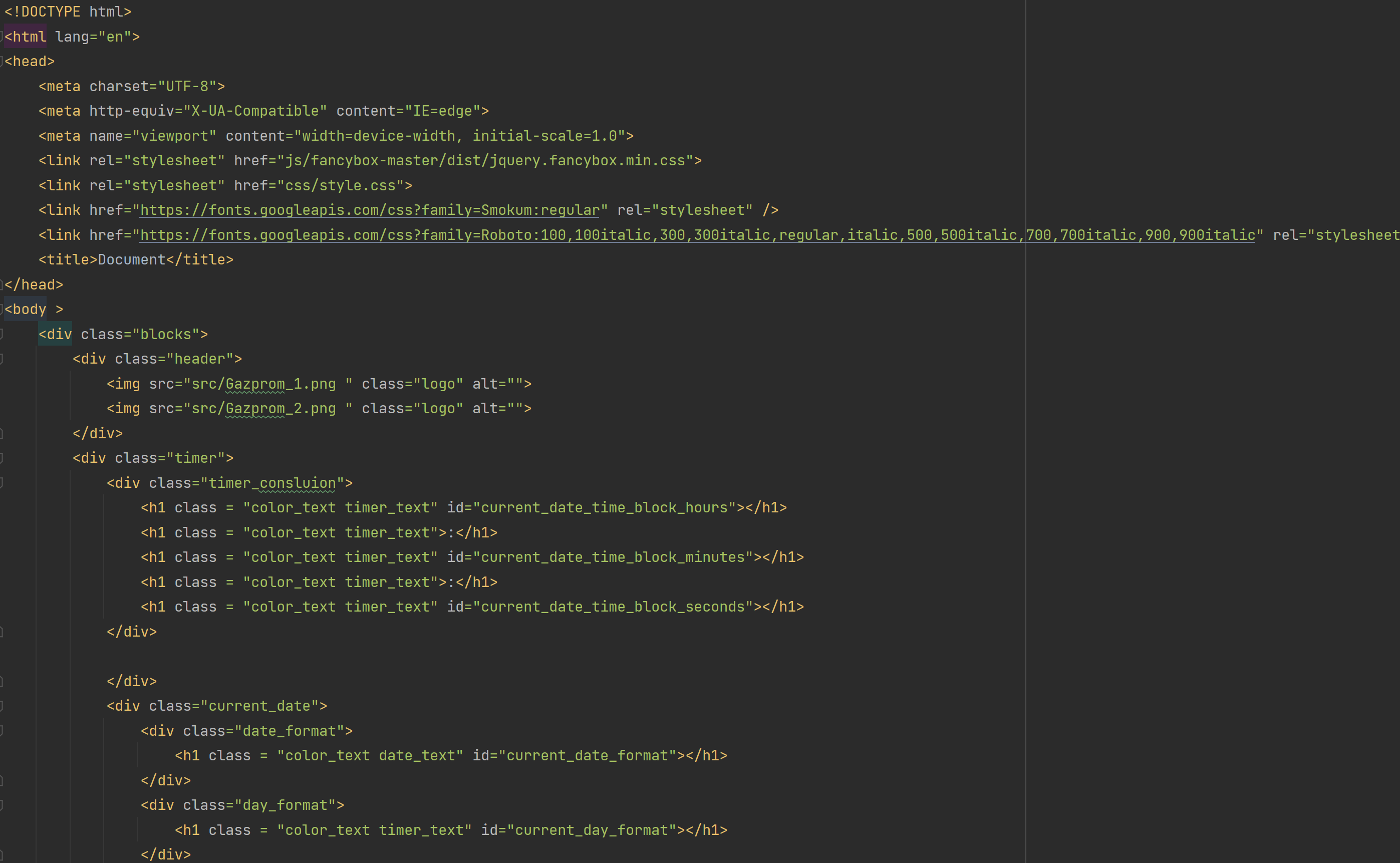


Рисунок 2 – HTML вёрстка

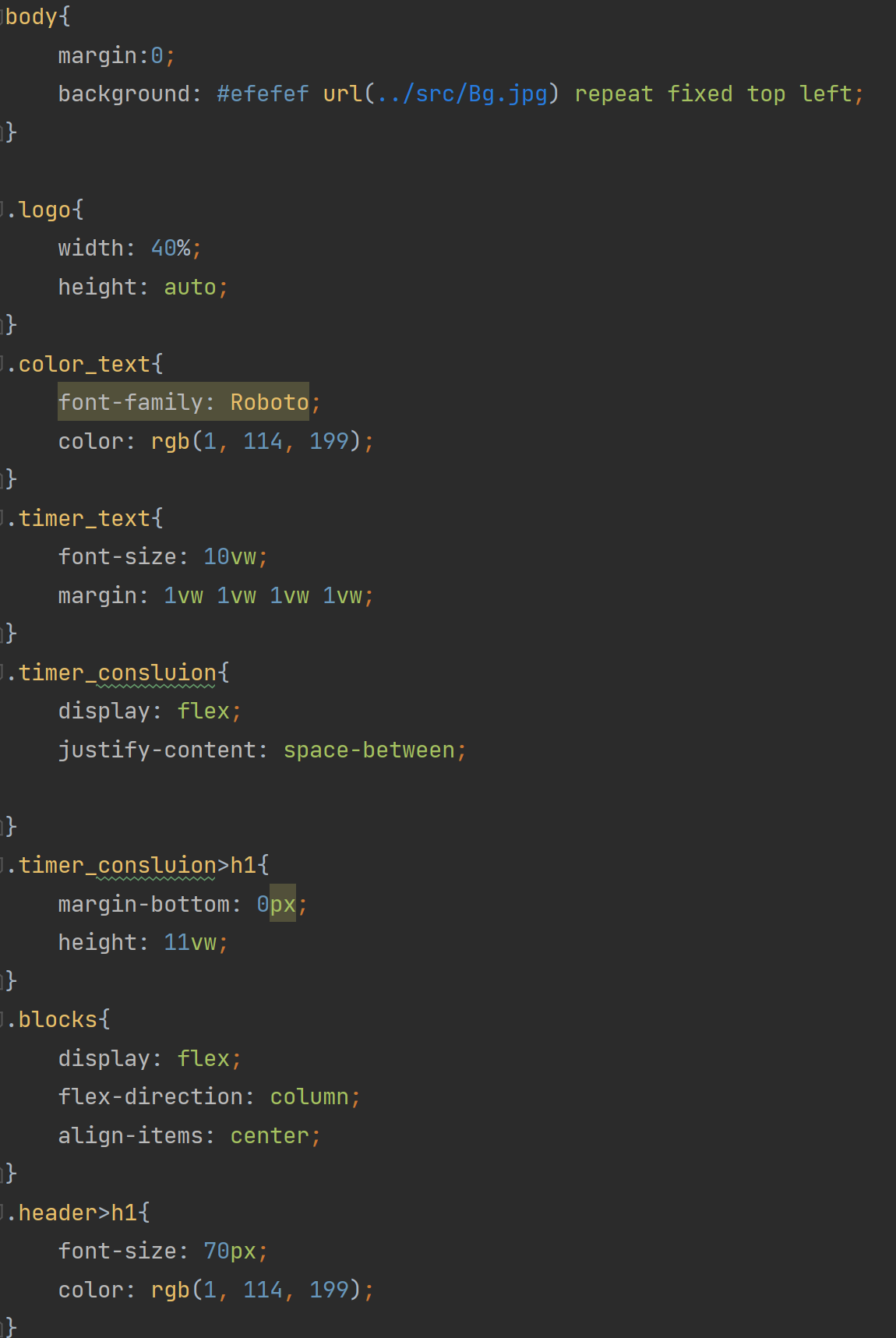


Рисунок 3 – стилизация графического интерфейса

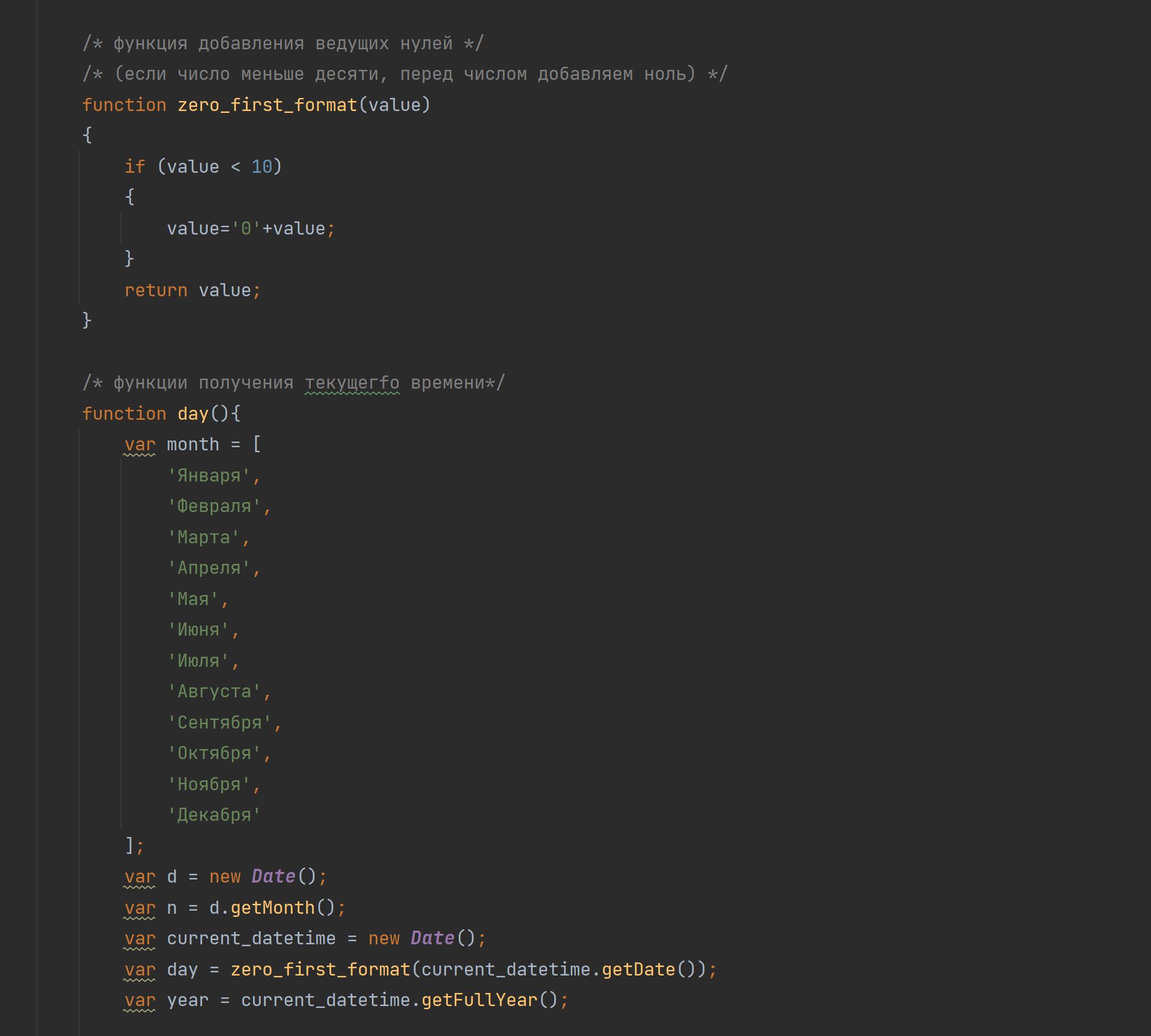


Рисунок 4 – программирование отображения настоящего времени в приложении на языке JavaScript

## Серверное приложение

Также были изучены вопросы разработки клиентских приложений. Более подробно акцентировано внимание на разработку серверных 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 серверной части веб-приложения «Расписание/График». Данное приложение предоставляет возможность составлять график работы и дел сотрудников.

## Выбор архитектурных подходов и технологий

Важным этапом в процессе разработки ИС 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 сервера на клиент и обратно только чистых данных, исключая HTML, CSS и JavaScript;
* Возможность составление понятного и наглядного API с возможностью его использования в самой разработанной информационной системе, так и разработки дополнительных АИС, как силами разработчиков-энтузиастов, которые хотят реализовывать собственные идеи, так и силами собственных разработчиков, которые ведут поддержку данной ИС. Например, бот ВКонтакте, который будет самостоятельно уведомлять всех, кто захочет на рассылку расписания.

На стороне самого сервера, внутри его логики была выбрана архитектура MVC (model-view-controller), поскольку это просто самый популярный и часто используемый подход.

Шаблон проектирования MVC предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: Модель, Представление и Контроллер – таким образом, что модификация каждого компонента может осуществляться независимо:

* Представление (View) отвечает за взаимодействие с пользователем. То есть код компонента view определяет внешний вид приложения и способы его использования;
* Модель (Model) отвечает за данные, а также определяет структуру приложения. Например, если вы создаете To-Do приложение, код компонента model будет определять список задач и отдельные задачи;
* Контроллер (Controller) отвечает за связь между model и view. Код компонента controller определяет, как сайт реагирует на действия пользователя. По сути, это мозг MVC-приложения.

Один из самых популярных фреймворков, который диктует MVC подход это Laravel - это относительно новый веб фреймворк написанный на языке PHP (первый выпуск произошел в 2011 году). Laravel взял лучшее из другого бэкэнд-фреймворка, такого как Rails, однако сейчас уже Laravel сильно обогнал его по распространенности.

Итоги выбора архитектуры и технологий:

* Была выбрана архитектура приложения SPA;
* Архитектура самой back-end части приложения MVC и метод общения сервера и клиента с помощью REST API;
* Фреймворк, реализующий шаблон MVC – Laravel.
* Спроектированы оптимизированные процессы составления расписания

## Процесс разработки серверного приложения

Этапы разработки:

* Загрузка библиотек и фреймворков и начальная настройка
* Разметка таблиц базы данных и миграция
* Разработка логики работы с данными (model)
* Разработка логики обработки запросов (controller)
* Разработка логики представления пользователю (view)

1) Загрузка библиотек и фреймворков и начальная настройка

Для того, чтобы установить все нужные пакеты для разработки на PHP нужно обратится к сборщику пакетов Composer и указать ему скачать всё что вам нужно командой «composer install <название пакета>» в терминале.

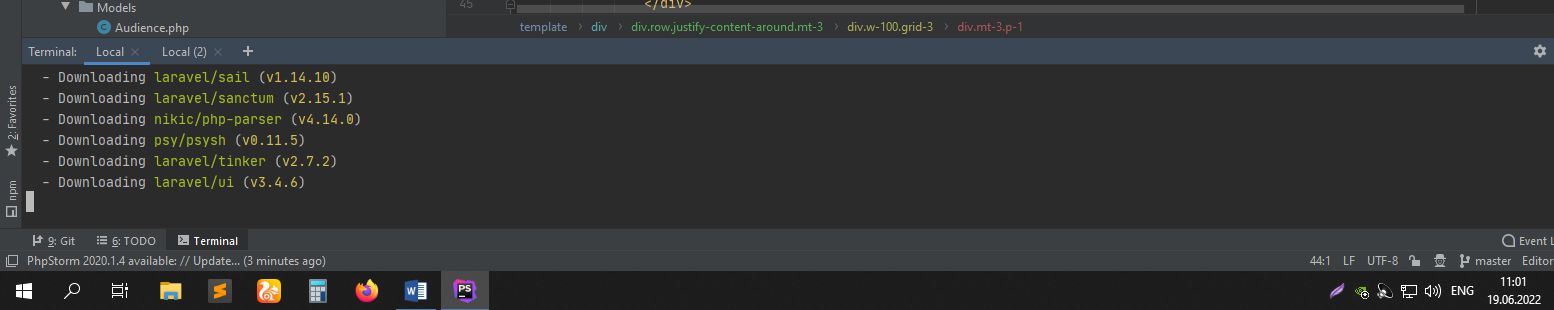


Рисунок 5 - Терминал

Далее требуется настроить проект и связать проект с базой данных.

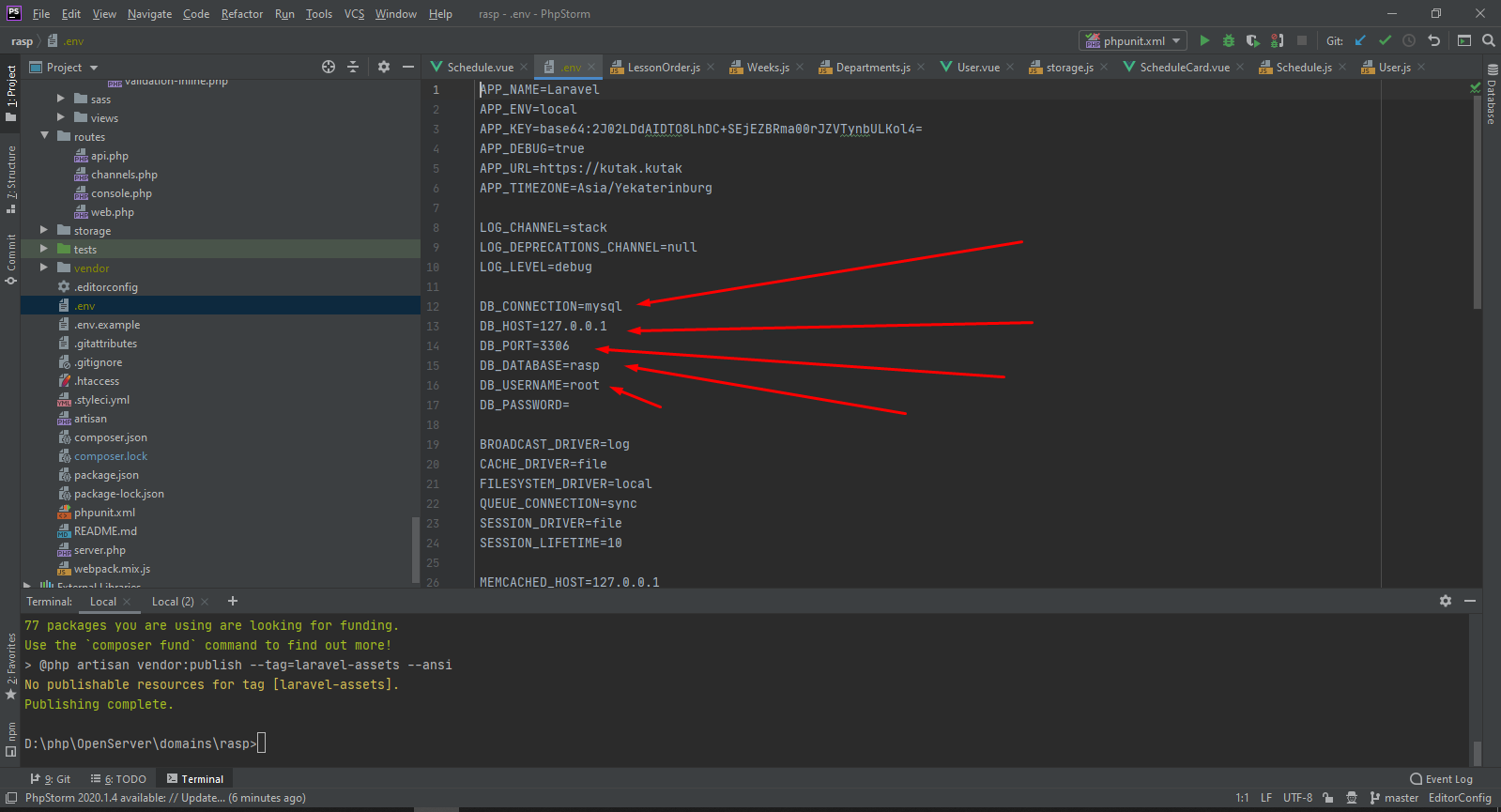


Рисунок 6 - Конфигурационный файл

2) Разметка таблиц базы данных и миграция

Теперь можно приступить к разработке, в первую очередь надо продумать структуру и связи между таблицами в базе данных MySQL.

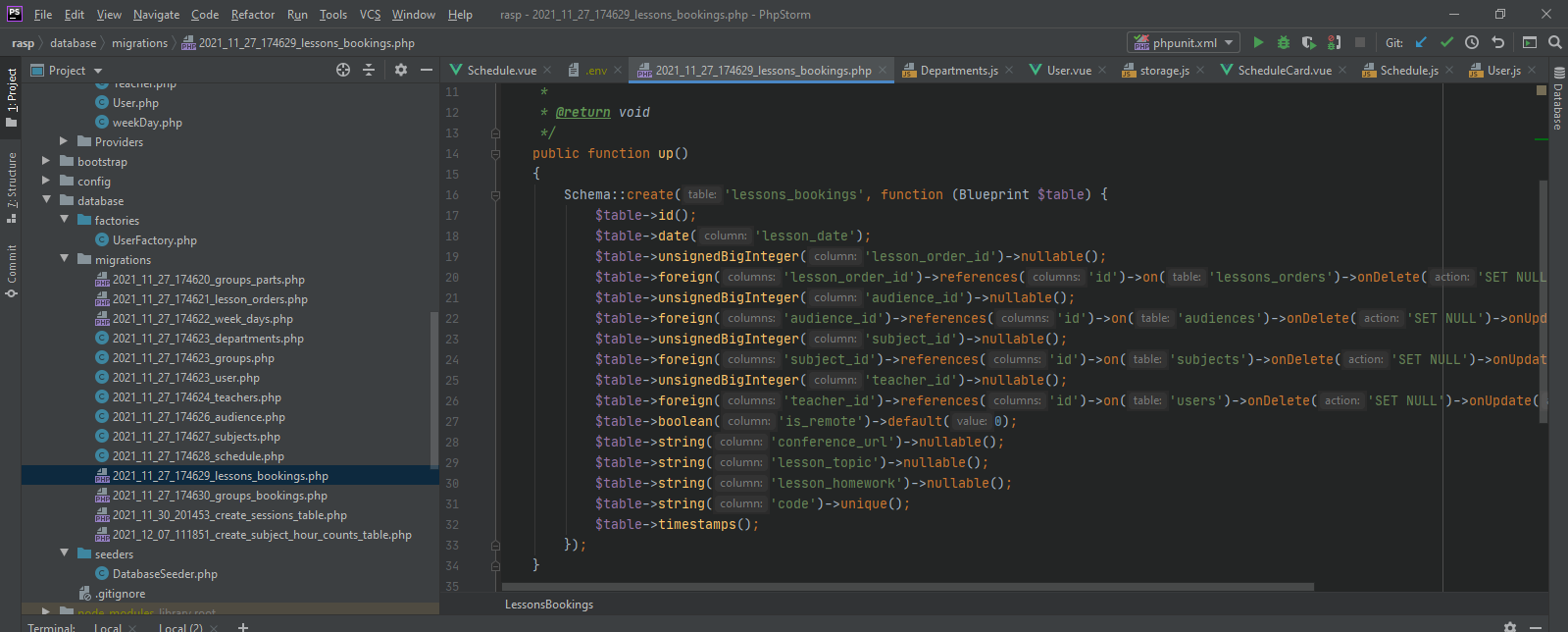


Рисунок 7 – Пример файла миграций

В фреймворке Laravel нет прямого взаимодействия с базой данных, все таблицы записываются в миграции, а затем эти же самые миграции запускаются и заполняют БД таблицами и связями.

* 3) Разработка логики работы с данными (model)

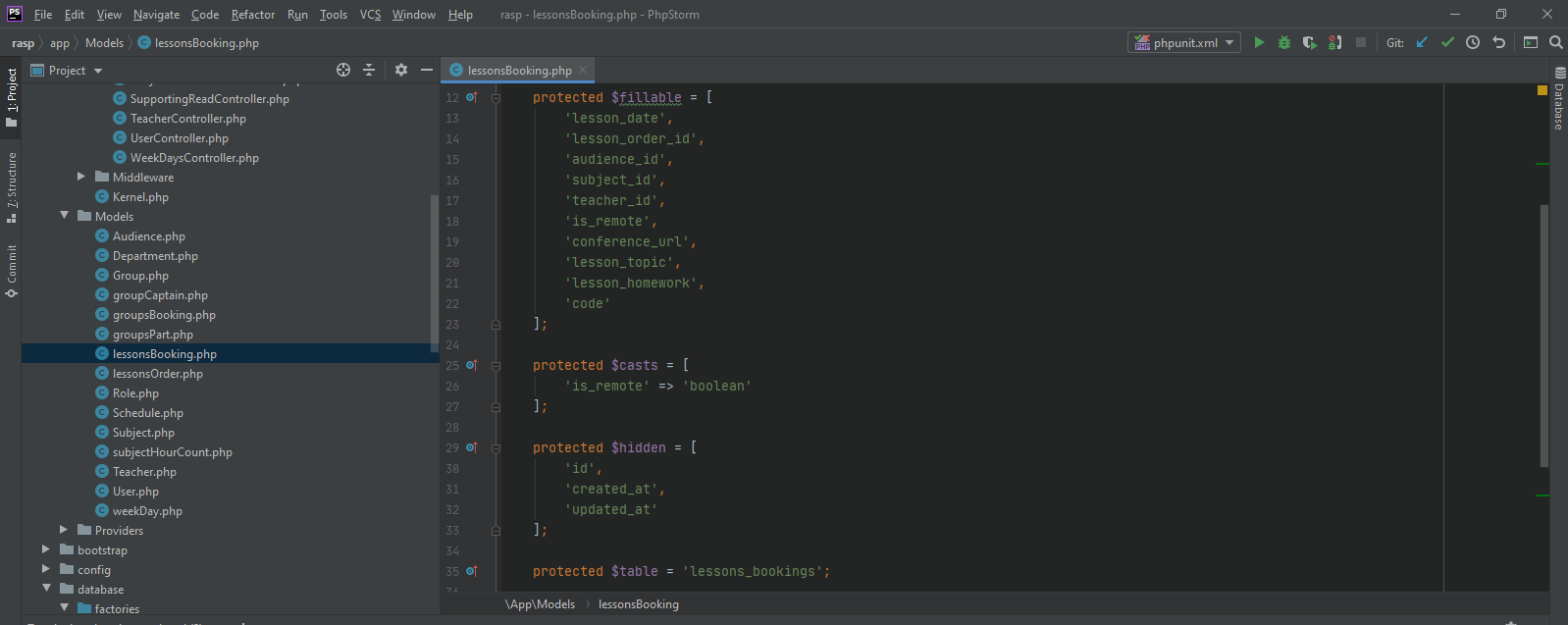


Рисунок 8 – Пример модели

При разработке моделей описываются доступные поля таблиц и некоторые скрипты для взаимодействия с данными при их выборке

4) Разработка логики обработки запросов (controller)

Контроллеры - это центральные мозги, через них проходят все запросы и обработки, без них ничего бы не работало.

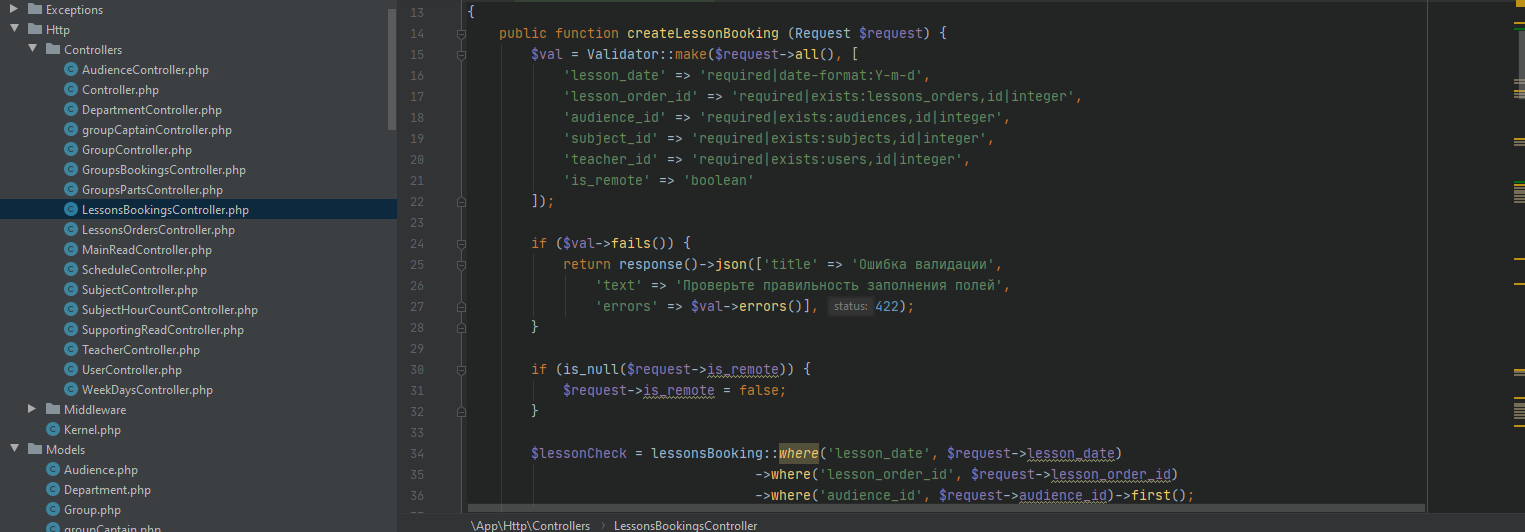


Рисунок 9 – Пример контроллера

Контроллер проводит валидацию всех данных, проводит все проверки с помощью операторов «if», так же обращается к базе данных для проверок, и только после всего этого выполняет нужный запрос и отдаёт либо данные, либо только обратный ответ с успешным выполнением

5) Разработка логики представления пользователю (view)

Поскольку разработкой клиентской части занимается отдельный человек и идет разработка SPA, а это значит что всё представление рендерится в клиентской части, то представления на стороне back-end являются почти пустыми.

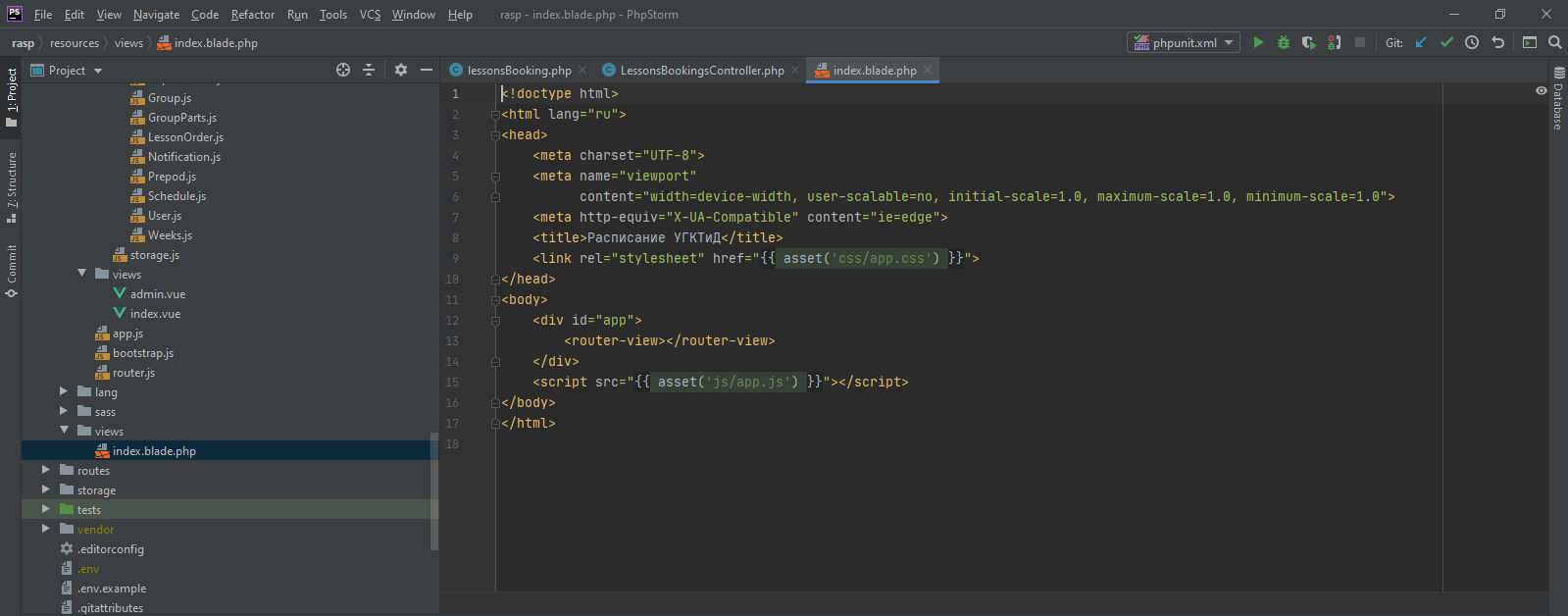


Рисунок 10 – Представление

После разработки всех частей информационной системы почти всё готова, последним этапом является тестирование.

Тестирование было совершено несколькими методами:

* В первую очередь написаны unit тесты, которые проверяют правильность работы информационной системы
* Затем также проведено ручное тестирование

## Оптимизированные процессы составления графика работы

На данных диаграммах показаны оптимизированные процессы разработанной информационной системы, они наглядно демонстрируют, как данная ИС сокращает количество ручных операций.

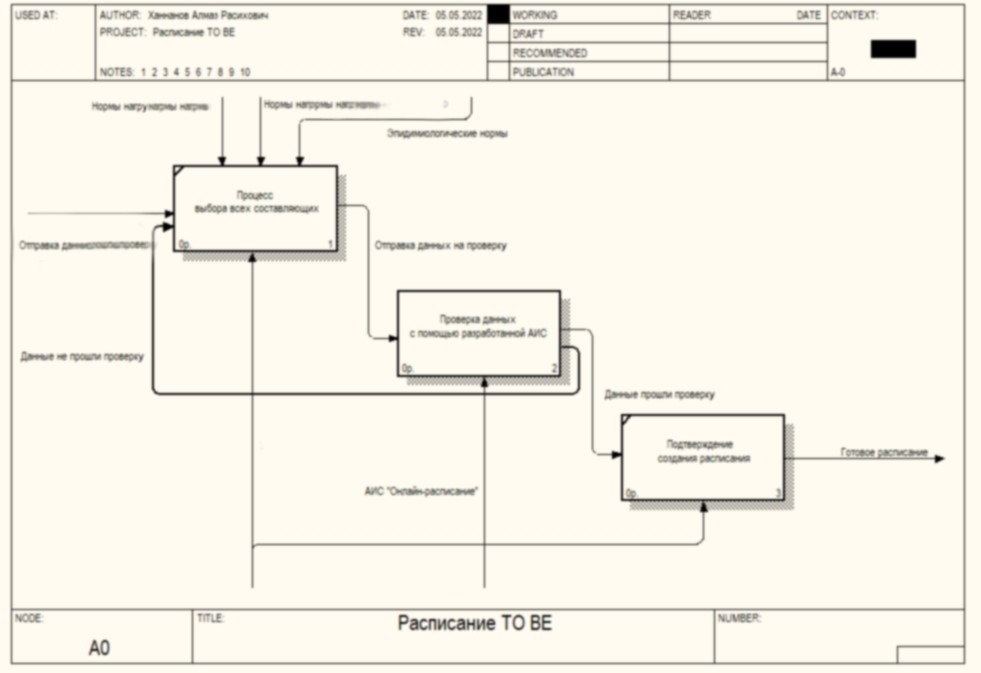


Рисунок 11 – Диаграмма спроектированной ИС в нотации IDEF0

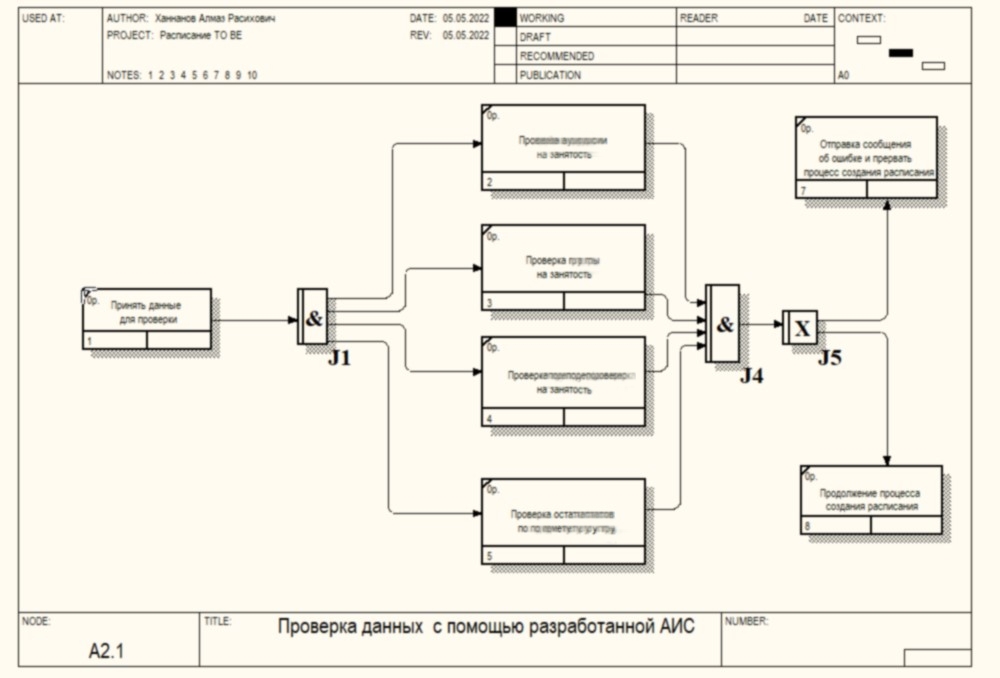


Рисунок 12 – Декомпозиция блока «Проверка данных» в нотации IDEF3

# заключение

В ходе производственной практики были разработаны:

* Локальное приложение для информационного стенда в браузерном окружении;
* Техническая документация по модификации локального приложения;
* Серверная часть веб-приложения «Расписание/График» на языке программирования PHP, исполняемая веб-сервером Apache;
* Диаграммы процессов работы веб-приложения «Расписание/График»;
* Документация программного интерфейса серверной части приложения «Расписание/График».

При разработке локального приложения были применены языки HTML, CSS, JavaScript. Также использованы библиотеки, написанные на языке JavaScript:

* JQuery;
* Fancybox.

При разработке серверной части веб-приложения были применены язык PHP, фреймворк Laravel, основанный на языке PHP, шаблон проектирования MVC. Разработка основывалась на архитектуре веб-приложений SPA. В качестве базы данных использована реляционная MariaDB. После окончания разработки были проведены испытания и тестирование информационной системы и введение АИС 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: 13.12.2022)
2. Википедия [Электронный ресурс].Режим доступа: URL: <https://ru.wikipedia.org/wiki> (Дата обращения: 14.12.2022)
3. Хабр [Электронный ресурс]. Режим доступа: URL: <https://habr.com/ru/all/> (Дата обращения: 12.12.2022)
4. Блог Кинзябулатов Рамиля [Электронный ресурс]. Режим доступа: URL: https://trinion.org/blog (Дата обращения: 12.12.2022)
5. Индикатор [Электронный ресурс]. Режим доступа: URL: <https://indicator.ru> (Дата обращения: 12.12.2022)
6. Файловый архив студентов [Электронный ресурс]. Режим доступа: URL: <https://studfile.net> (Дата обращения: 13.12.2022)