Страница на всякий случай

содержание

1 [Введение 5](#_Toc106894389)

2 [Основная часть 6](#_Toc106894390)

2.1 [Анализ нынешнего принципа составления расписания 6](#_Toc106894391)

2.2 [Выбор архитектурных подходов и технологий 8](#_Toc106894392)

2.3 [Разработка 11](#_Toc106894393)

2.4 [Оптимизированные процессы составления расписания 15](#_Toc106894394)

3 [заключение 16](#_Toc106894395)

4 [Литература 17](#_Toc106894396)

# 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 пройдена в Центре прототипирования радиоэлектронного кластера республики Башкортостан в должности back-end разработчик.

Объектом исследования является разработка распределенной клиент-серверной информационной системы «Онлайн расписание».

Предмет исследования — разработка серверной части(back-end) распределенной клиент-серверной информационной системы «Онлайн расписание» с применением современных технологий и подходов.

Актуальность выбранной работы заключается в том, что в Государственном бюджетном профессиональном образовательном учреждении «Уфимский государственный колледж технологии и дизайна» действует устаревший, неэкологичный, трудоёмкий, требующий много времени и внимания человека принцип составления, корректировки и размещения расписания учебных занятий в данном учреждении. В связи с этим было поручено модифицировать данный принцип и разработать серверную часть нового веб-приложения «Онлайн расписание»

Целью данной работы является проанализировать требования к разрабатываемому ПО, выбрать архитектуру, применяемые технологии и подходы, разработать, провести тестирование и внедрить разработанное ПО.

Используемые методики:

* Архитектурный подход MVC (model-view-controller)
* Архитектурный стиль REST

# Основная часть

## Анализ нынешнего принципа составления расписания

Для начала анализа воспользуемся положением об учебной части Новосибирского торгово-экономического колледжа в качестве примера.

Согласно Положению об учебной части ГБПОУ Новосибирского торгово-экономического колледжа:

«Учебная часть является структурным подразделением учебного заведения, осуществляющим планирование, организацию, учет и контроль за ходом учебного процесса по формам обучения и учебным дисциплинам, определенным ФГОС СПО по специальностям и профессиям колледжа.

В своей работе учебная часть руководствуется Федеральным законом Российской Федерации от 29 декабря 2012 г. No 27Э-ФЗ «Об образовании в Российской Федерации», Порядком организации и осуществления образовательной деятельности по образовательным программам среднего профессионального образования, утвержденного приказом Министерства образования и науки Российской Федерации от 14 июня 2013 г. N 464; постановлениями, приказами, распоряжениями и инструктивными письмами Министерства труда, занятости и трудовых ресурсов НСО, Уставом колледжа и правилами внутреннего распорядка, приказами директора, на стоящим Положением.

В своей деятельности учебная часть взаимодействует с зам. директора по научно-методической работе, зам. директора по воспитательной работе, председателями цикловых комиссий, отделом кадров, бухгалтерией, библиотекой по вопросам планирования, организации учета и контроля учебного процесса.»

Исходя из данного примера можно установить первую причину для автоматизации и упрощения принципа составления расписания, поскольку учебная часть колледжа взаимодействует и заведует многими функциями образовательного учреждения, а разработанное ПО позволит ускорить работу и уменьшить затраты, как и на материалы (бумага и прочее), так и на человеко-часы.

Сравним нынешний вид расписания с уже введенными новыми системами и принципами, зайдём в любой подручной браузер, возьмем для примера ИС «Онлайн расписание» Уфимского колледжа радиоэлектроники, телекоммуникаций и безопасности, в FireFox напишем «расписание УКРТБ» перейдем по результатам поиска (Рисунок 1).

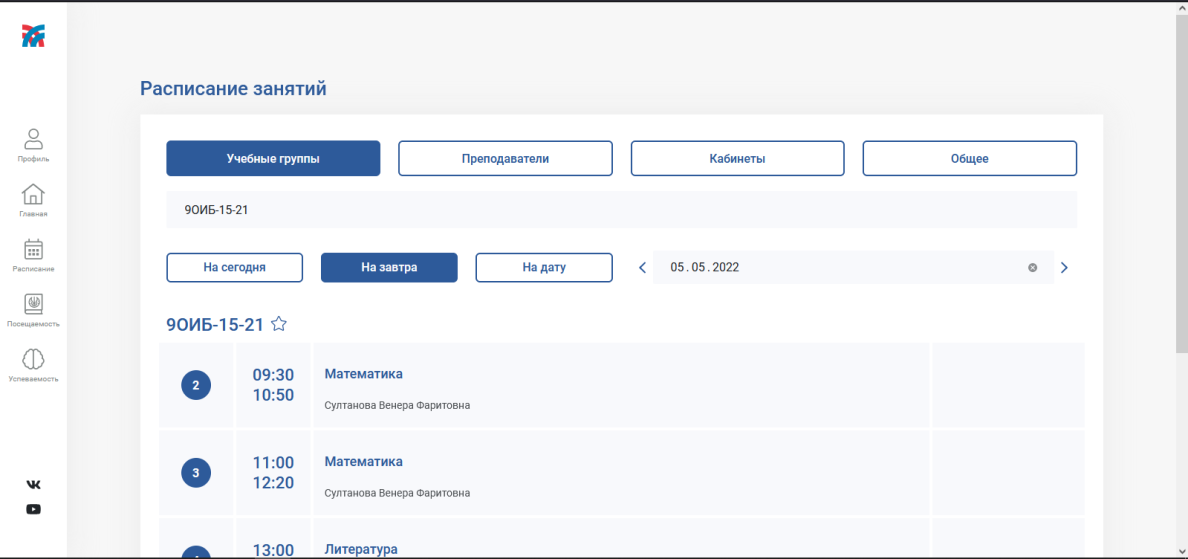


Рисунок 1 – Пример онлайн расписания.

Как мы можем видеть на данном рисунке, это очень удобная и простая для пользования ИС, которая даже имеет возможность расширения и дополнительной автоматизации с помощью бота, который будет способен сам напоминать студентам и преподавателям о занятиях.

На данный момент в Уфимском колледже технологии и дизайна действует архаичное расписание в неудобном формате, которое следует модернизировать (Рисунок 2), поскольку даже студенту приходится выполнять много действий на обычный просмотр расписания, а точнее:

* Взглянуть на расписание;
* Понять с какой пары начинаются занятия;
* Перевести взгляд на расписание звонков;
* Узнать время начала занятий.

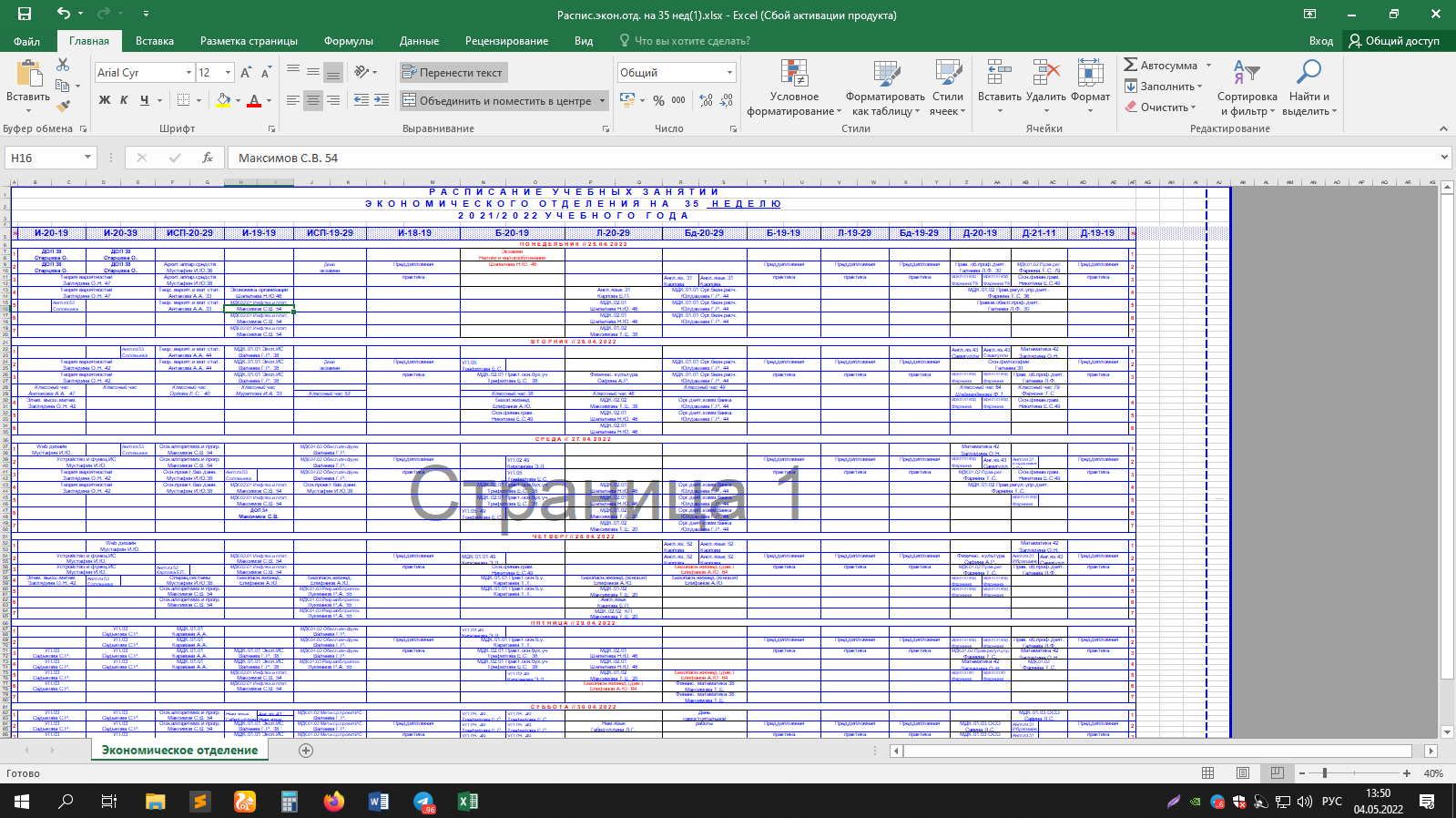


Рисунок 2 – Расписание УГКТиД

В связи с этим будет разработано веб-приложение, которое будет удовлетворять всем базовым запросам пользователя (удобность и простота использование) и автоматизировать такие вещи как:

* Составление расписание, а точнее автоматизация проверки свободных аудиторий, групп и преподавателей, благодаря возможностям разработанной ИС и ЭВМ;
* Просмотр расписания;
* В перспективе, процесс уведомления о предстоящих занятиях, а точнее расширение разработанной ИС с помощью дополнительной системы уведомления (ВК бот, телеграм бот).

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

## Выбор архитектурных подходов и технологий

Важным этапом в процессе проектирования 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[[1]](#footnote-1).

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[[2]](#footnote-2)).

Более разумным решением стал выбор архитектуры SPA и вместе с ним идет выбор подхода REST API[[3]](#footnote-3) 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[[4]](#footnote-4), 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

## Разработка

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.

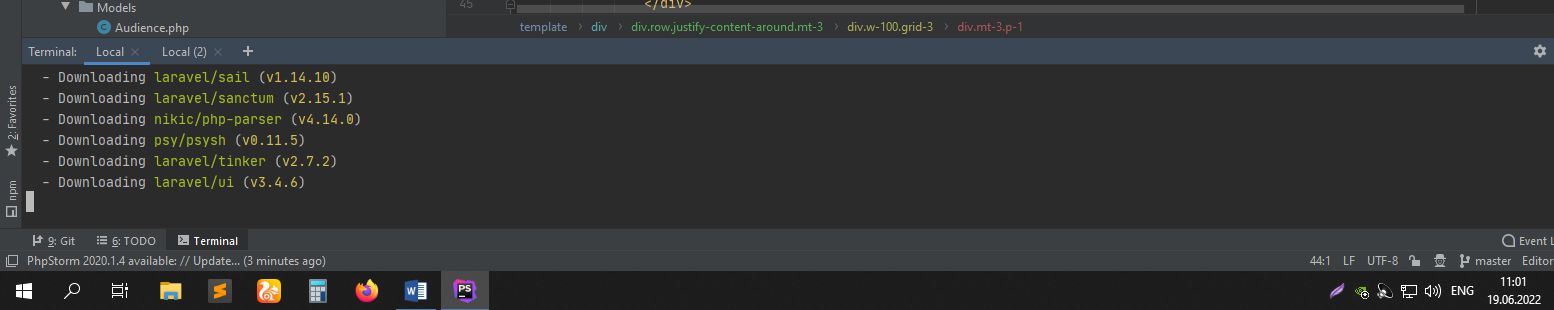


Рисунок 3 - Терминал

Далее требуется настроить проект и связать проект с базой данных.

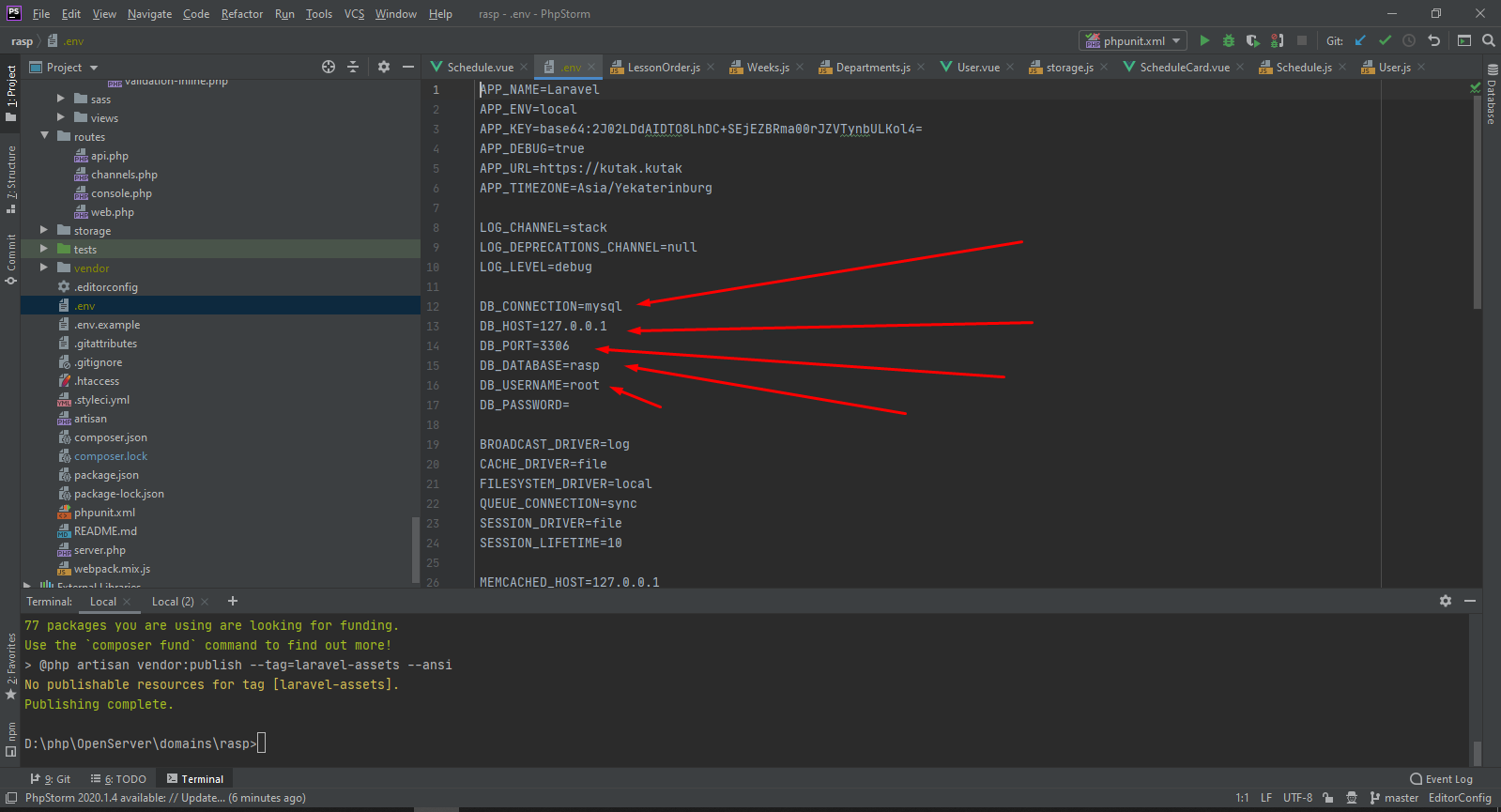


Рисунок 4 - Конфигурационный файл

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.

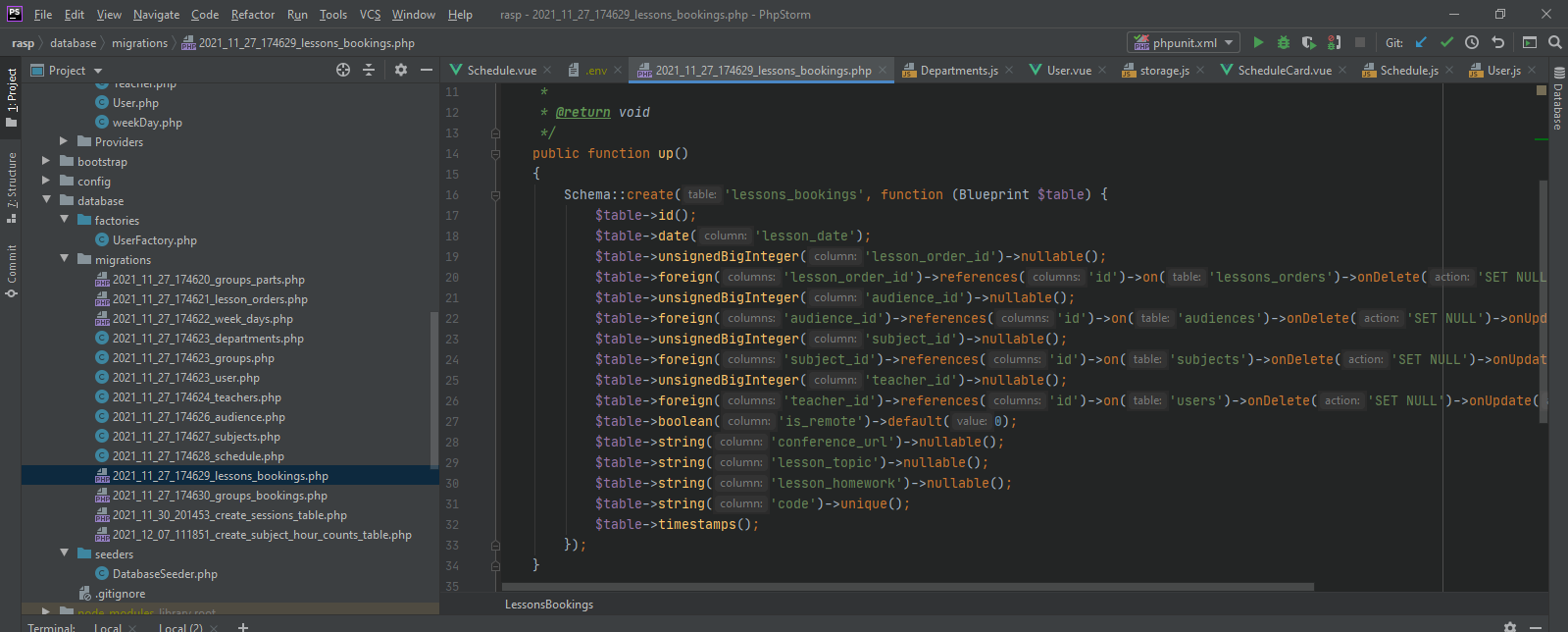


Рисунок 5 – Пример файла миграций

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)

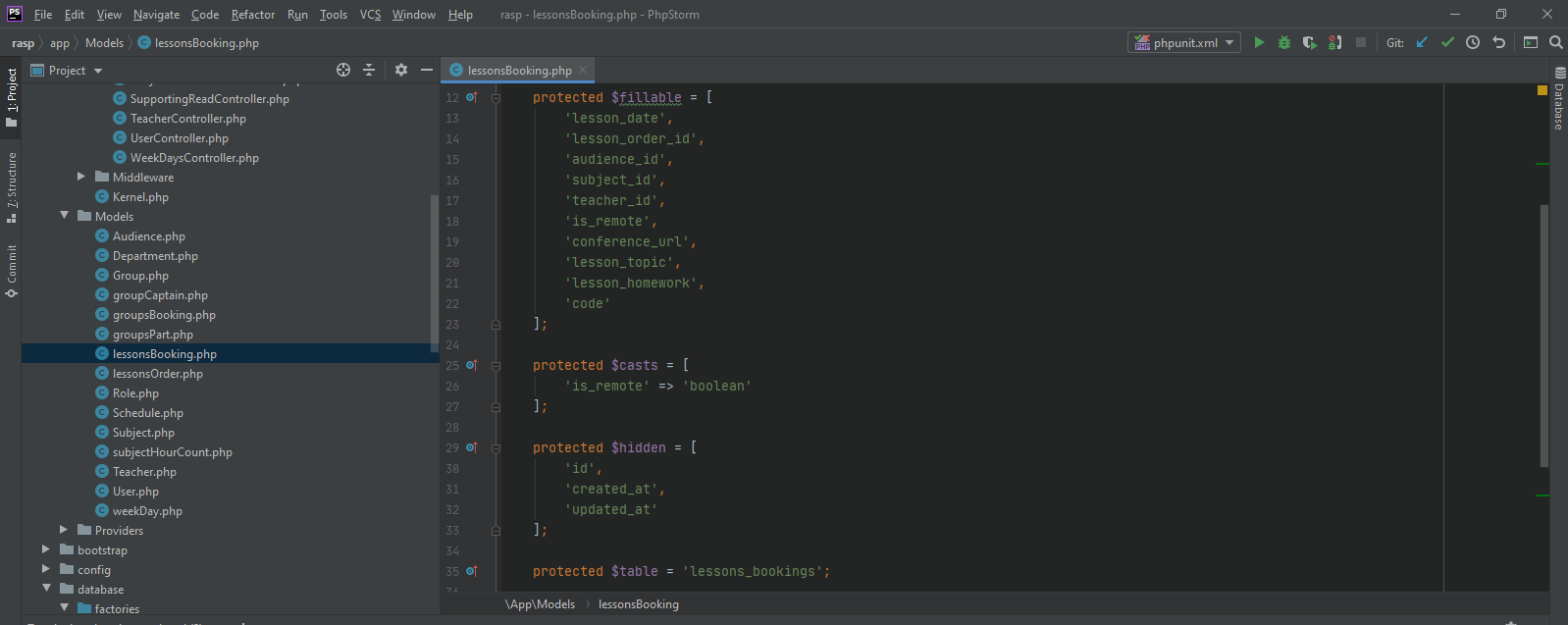


Рисунок 6 – Пример модели

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.

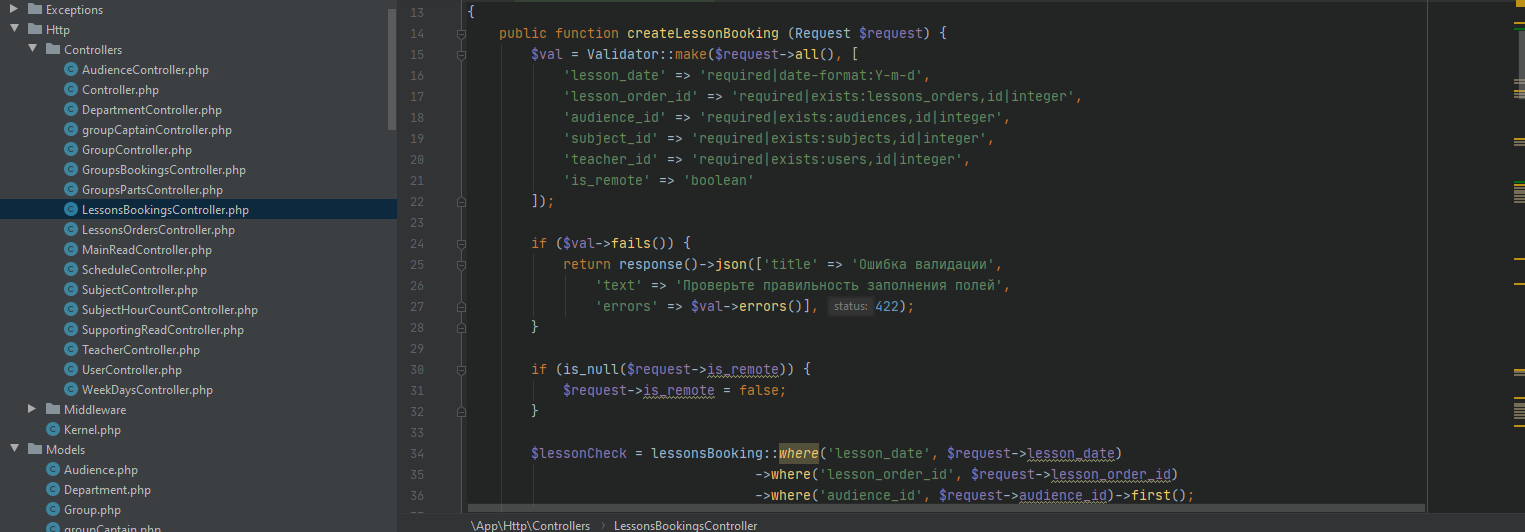


Рисунок 7 – Пример контроллера

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.

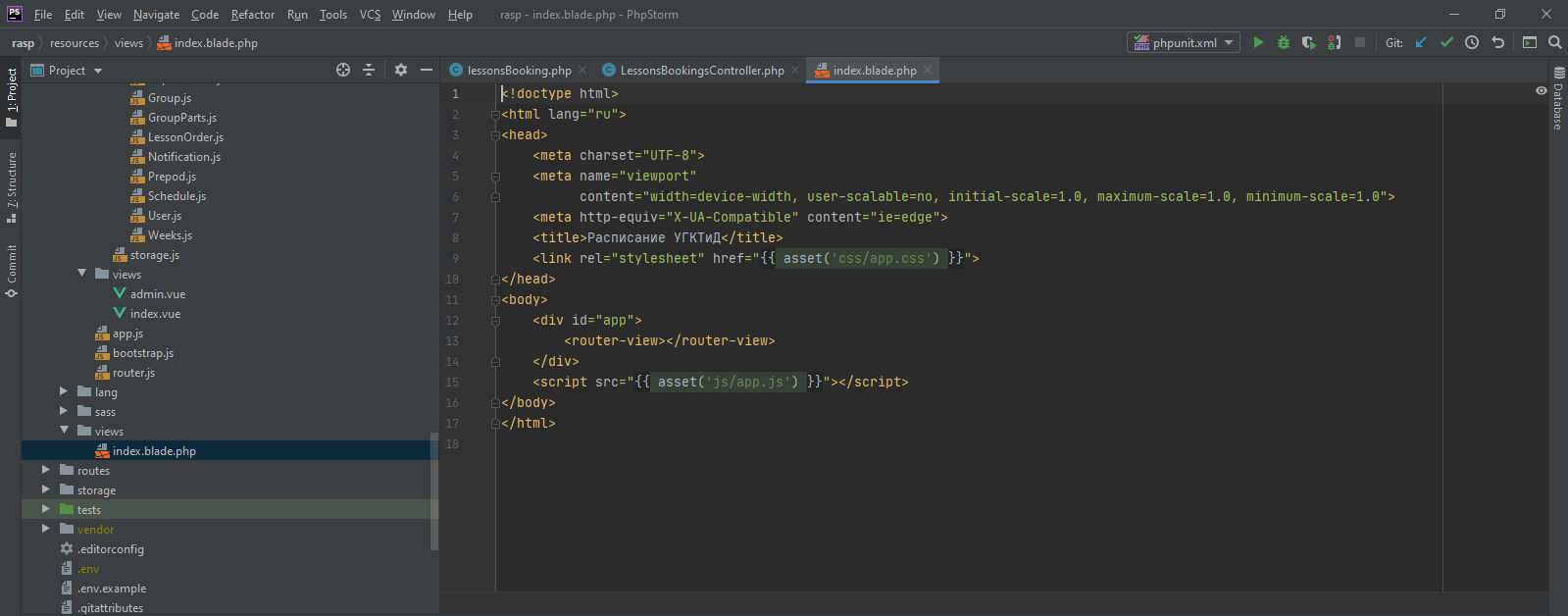


Рисунок 8 – Представление

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 расписания

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

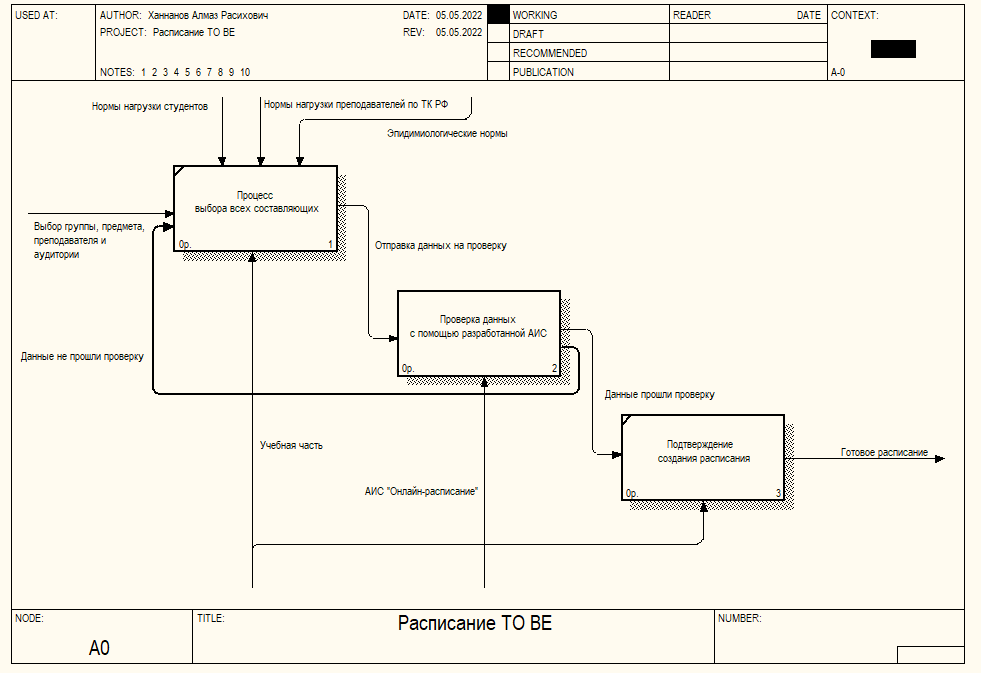


Рисунок 9 – Диаграмма спроектированной ИС в нотации IDEF0

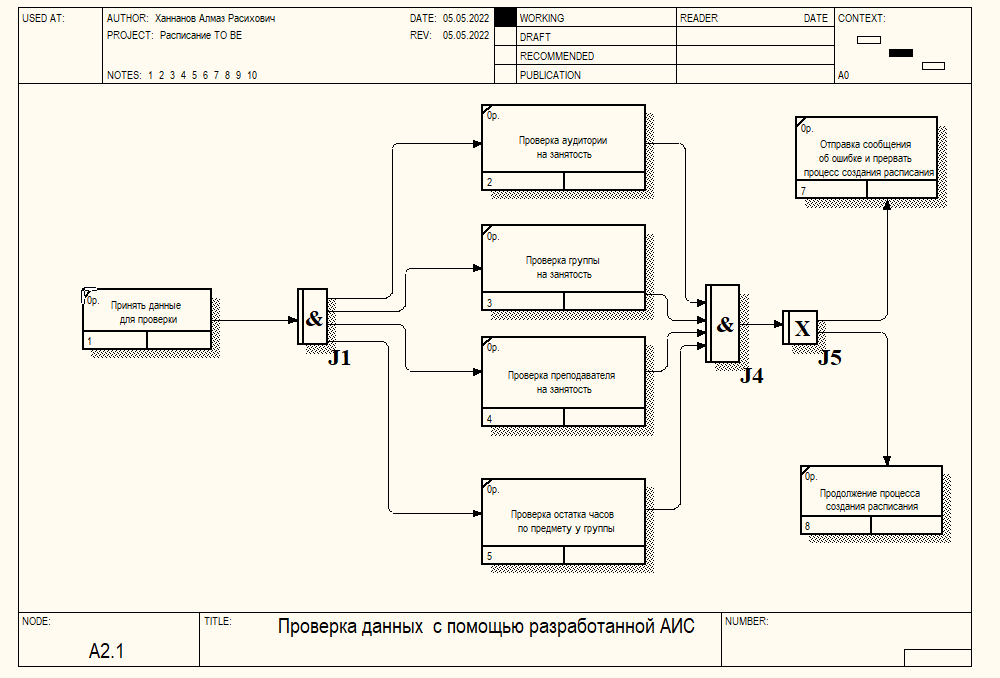


Рисунок 10 – Декомпозиция блока «Проверка данных» в нотации IDEF3

# заключение

В ходе производственной практики были изучены различные автоматизированные информационные системы, проведены анализы и сравнения наиболее подходящих решений для разработки новой информационной системы, проведен подбор архитектуры приложения и подходов для разработки, разработана серверная часть распределенной клиент-серверной информационной системы, проведено тестирование.

# 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: 20.12.2021)
2. Википедия [Электронный ресурс].Режим доступа: URL: <https://ru.wikipedia.org/wiki> (Дата обращения: 20.12.2021)
3. Учебный портал УКРТБ [Электронный ресурс]. Режим доступа: URL: <https://study.ukrtb.ru/timetable> (Дата обращения: 20.12.2021)
4. Хабр [Электронный ресурс]. Режим доступа: URL: <https://habr.com/ru/all/> (Дата обращения: 20.04.2022)
5. Блог Кинзябулатов Рамиля [Электронный ресурс]. Режим доступа: URL: https://trinion.org/blog (Дата обращения: 05.05.2022)
6. Корпоративный менеджмент [Электронный ресурс]. Режим доступа: URL: https://www.cfin.ru/ (Дата обращения: 06.05.2022)

1. CSR (Client Side Rendering) – рендеринг на стороне клиента (браузер) [↑](#footnote-ref-1)
2. SSR (Server Side Rendering) – рендеринг на стороне сервера [↑](#footnote-ref-2)
3. REST API – подход, при котором клиент и сервер общаются между собой только передавая чистые данные [↑](#footnote-ref-3)
4. Ruby on Rails — фреймворк, написанный на языке программирования Ruby [↑](#footnote-ref-4)