Содержание

[Введение 4](#_Toc119062679)

[1 Теоретическая часть **Ошибка! Закладка не определена.**](#_Toc119062680)

[1.1 Анализ предметной области **Ошибка! Закладка не определена.**](#_Toc119062681)

[1.1.1 Анализ действующей ИС **Ошибка! Закладка не определена.**](#_Toc119062682)

[1.1.2 Рассмотрение аналогов ИС **Ошибка! Закладка не определена.**](#_Toc119062683)

[1.1.3 Анализ рисков **Ошибка! Закладка не определена.**](#_Toc119062684)

[1.2 Выбор программного обеспечения для управления проектами **Ошибка! Закладка не определена.**](#_Toc119062685)

[1.2.1 Особенности Hygger **Ошибка! Закладка не определена.**](#_Toc119062686)

[1.2.2 Особенности Monday.com **Ошибка! Закладка не определена.**](#_Toc119062687)

[1.2.3 Особенности Smartsheet **Ошибка! Закладка не определена.**](#_Toc119062688)

[2 Практическая часть **Ошибка! Закладка не определена.**](#_Toc119062689)

[2.1 Разработка технического задания **Ошибка! Закладка не определена.**](#_Toc119062690)

[2.1.1 Понятие ТЗ **Ошибка! Закладка не определена.**](#_Toc119062691)

[2.1.2 Содержание ТЗ **Ошибка! Закладка не определена.**](#_Toc119062692)

[2.1.3 Описание функций продукта **Ошибка! Закладка не определена.**](#_Toc119062693)

[2.2 Создание диаграмм **Ошибка! Закладка не определена.**](#_Toc119062694)

[2.2.1 Анализ и сравнение процессов ИС в стадиях AS IS и TO BE **Ошибка! Закладка не определена.**](#_Toc119062695)

[2.2.2 Проектирование процессов взаимодействия с разрабатываемой информационной системой **Ошибка! Закладка не определена.**](#_Toc119062696)

[2.2.3 Проектирование web-сайта **Ошибка! Закладка не определена.**](#_Toc119062697)

[2.3 Выбор архитектурных подходов и технологий **Ошибка! Закладка не определена.**](#_Toc119062698)

[2.4 Составление эскизного проекта **Ошибка! Закладка не определена.**](#_Toc119062699)

[2.5 Управление проектом в Hygger **Ошибка! Закладка не определена.**](#_Toc119062700)

[Заключение **Ошибка! Закладка не определена.**](#_Toc119062701)

[Список литературы **Ошибка! Закладка не определена.**](#_Toc119062702)

# Введение

В настоящее время наблюдается развитие информационных технологий в организациях разного рода деятельности. Это обусловлено стремлением к сокращению времени обработки и анализа данных. Особенно оперативность обработки и распространения информации важна на предприятиях со сложной технологической цепочкой, так как с увеличением количества подразделений предприятия наблюдается рост временных затрат на прохождения документации, что приводит замедлению темпов реализации управленческих и технологических решений.

В рамках данной работы проведено проектирование информационной системы для предметной области «Составление расписаний занятий». Технология составления расписаний учебных занятий представляет собой достаточно трудоемкий процесс, так как необходимо учитывать множество факторов, включающих: занятость преподавателей, учебных кабинетов, уровень учебной нагрузки на учащихся, необходимость проведения лабораторных занятий в специально отведенных для этого кабинетах и др. Когда учебное заведение включает достаточно большое количество учебных классов, кабинетов, преподавателей, то при составлении расписаний зачастую невозможно вручную провести оптимальное распределение нагрузки. Использование информационных технологий для решения поставленной задачи обеспечит возможности сокращения трудозатрат на составление расписаний, минимизации ошибок.

Предмет исследования: разработка информационной системы «Расписание занятий».

Целью работы является разработка информационной системы «Расписание занятий».

# Общая часть

## Описание предметной области

Расписания можно использовать для организации и координации деятельности в различных условиях, таких как школа, бизнес и личная жизнь. Они могут помочь людям оставаться организованными и обеспечивать эффективное и своевременное выполнение задач.

Расписание занятий - это план выполнения конкретной задачи или действия, часто в течение определенного периода времени. План включает в себя список задач или событий, которые необходимо выполнить, и время, в которое они должны произойти.

Существует несколько типов расписаний, которые можно использовать в зависимости от потребностей и целей человека или организации:

* годовой график - это план, в котором излагаются действия и задачи, которые необходимо выполнить в течение года. Он может включать ежемесячные расписания, а также любые повторяющиеся события или действия, которые происходят ежегодно;
* график проекта - это план, в котором излагаются задачи и этапы, которые необходимо выполнить в рамках конкретного проекта. Он включает график выполнения каждой задачи, а также может включать необходимые ресурсы, зависимости между задачами и крайние сроки;
* сменный график - это план, в котором указывается время, в которое должны работать разные сотрудники. Он может включать разные смены для разных дней недели, а также может включать выходные и праздничные дни;
* расписание занятий - это план, в котором указаны время и дни, в которые запланированы занятия для различных занятий или разделов курса. Он обычно используется в школах и университетах для организации времени и места занятий;
* расписание собраний - это план, в котором указаны время и место запланированных собраний. Это могут быть регулярные встречи, а также разовые или специальные встречи;
* график поездок - это план, в котором излагается маршрут поездки, включая время и местонахождение рейсов, гостиниц и других мероприятий, связанных с поездками;
* расписание событий - это план, в котором указаны сроки и детали события, например конференции, концерта или торговой выставки. Он может включать время и место проведения различных сессий или мероприятий, а также любые специальные меры или приспособления, которые необходимо сделать;
* график технического обслуживания - это план, в котором излагаются задачи и действия, которые необходимо выполнить для технического обслуживания и ремонта оборудования или помещений. Он может включать график выполнения этих задач, а также сведения о ресурсах и персонале, необходимых для их выполнения;
* производственный график - это план, в котором излагаются задачи и действия, связанные с производством товаров или услуг. Он может включать график выполнения каждой задачи, а также сведения о ресурсах и персонале, необходимых для их выполнения.

Специфика предметных областей определила различия моделей и методов решения задач формирования расписаний. По схеме работы алгоритмы можно разделить на следующие типы:

* жадные алгоритмы, выбирающие очередные заявки/действия по некоторому правилу без дальнейшего их переназначения;

Задача формирования начального расписания решается последовательным выбором заявки или совокупности заявок и последующим ее включением в расписание в выбираемое время начала выполнения действия. То есть, на каждом шаге решения задачи формирования начального расписания присутствуют две операции выбора, после чего принимаются некоторые решения.

Задача оптимизации начального расписания решается последовательным выбором действия или совокупности действий и последующей его перестановкой в расписании в выбираемое время начала выполнения действия. То есть, на каждом шаге решения задачи оптимизации начального расписания также присутствуют две операции выбора, после чего принимаются некоторые решения.

Такой подход на обоих этапах характерен для жадных алгоритмов и широко используется для задач формирования расписаний и связанных с этим задач распределения ресурсов.

Жадный подход строит решение посредством последовательности шагов, на каждом из которых получается частичное решение поставленной задачи, пока не будет получено полное решение. При этом на каждом шаге – и это является главным в рассматриваемом методе – выбор должен быть:

* допустимым, то есть удовлетворять ограничениям задачи;
* локально оптимальным, то есть наилучшим локальным выбором среди всех допустимых вариантов, доступных каждом шаге;
* окончательным, то есть, будучи сделан, он не может быть изменен последующими шагами алгоритма.

Эти требования поясняют название метода: на каждом шаге он предполагает «жадный» выбор наилучшей доступной альтернативы в предположении, что последовательность локально оптимальных выборов приведет к глобально оптимальному решения всей задачи. Использование идеологии жадных алгоритмов предполагает цикличность алгоритмов для обоих этапов решения задачи формирования расписания.

* алгоритмы неполного перебора с возвратом, удаляющие ранее включенные в расписание действия/заявки, если на каком-то шаге включения очередной заявки/действия невозможно;
* алгоритмы, ищущие возможность включения очередной заявки/действия в расписание и варианты перестановок ранее включенных в расписание заявок/действий, конфликтующих с очередной заявкой/действием. Если перестановку одной заявки/действия найти не удается, некоторые методы ищут цепочки перестановок ранее включенных в расписание заявок/действий;
* декомпозиционные алгоритмы, разделяющие исходную задачу на подзадачи меньшего размера и применяющие к ним другие методы или дальнейшие шаги декомпозиции.

Среди многочисленных методов, применяющихся в автоматизированном формировании расписаний, наиболее распространёнными являются: метод моделирование отжига (simulated annealing), метод штрафов и запретов (tabu search algorithms) и эволюционные алгоритмы (evolutionary algorithms). Перечисленные методы применяются для оптимизации расписаний, а общим для них является введение критериев оптимальности и целевой функции оптимизации взамен требований к расписанию. Методы формирования начальных расписаний, к которым будут применены методы оптимизации, различны и определяются предметной областью.

Различия в задачах расписания и методах их решении определяются наличием и видом связей между заявками на формирование расписания. Это позволяет ввести следующую классификацию:

* расписание является множеством независимых друг от друга действий. То есть, любое действие, являющееся элементом расписания, может быть расположено в любом таймслоте интервала расписания с учетом обязательных ограничений. К этому типу относятся расписания экзаменов, занятий и т.п. Для этого типа задач расписания характерна возможность произвольного выбора заявок при его формировании. То же самое относится к выбору элементов расписания при его оптимизации. Для визуализации в расписании одновременно происходящих событий используется двумерной представление времени;
* расписание является множеством независимых друг от друга векторов действий, являющихся элементами расписания. Например, расписание движения пассажирского транспорта. Расписание будет формироваться из векторов заявок, каждая из которых включает прохождение одного перегона между станциями и пребывание на конечной станции перегона. Формирование расписания основано на возможности произвольного выбора векторов заявок и их включении в расписание на произвольное время начала первого действия с учетом обязательных ограничений. Оптимизация расписания обеспечивается перестановкой элементов расписания – векторов действий, то есть изменением времени начала первого действия. Круговое представление расписания для любого ресурса системы в данном случае позволяет наглядно оценить возникающую неравномерность действий расписания во времени;
* расписание является множеством независимых друг от друга иерархий действий. К этому типу расписаний относится, например, календарный график малоэтажного строительства поселка, где каждая иерархия представляет необходимые работы (действия) для возведения и обустройства одного здания. При формировании расписания и его оптимизации должны рассматриваться иерархии заявок и действий. Элементами расписания будут иерархии действий. Наиболее удобными для работы с расписаниями были бы диаграммы Ганта;
* расписание является множеством независимых друг от друга сетевых структур действий. К этому типу расписаний относится большинство календарных графиков мультипроектного планирования. Каждая сеть представляет отдельных проект. Проекты могут быть технологически независимыми, но объединенными по потребляемым ресурсам, прежде всего по возобновляемым ресурсам. При формировании расписания и его оптимизации должны рассматриваться сетевые структуры заявок и действий. Элементами расписания будут сетевые структуры действий. Как и в предыдущем случае диаграммы Ганта наиболее предпочтительны для представления календарного графика (расписания). Так как иерархические структуры являются частным случаем сетевых структур, то методы формирования расписания в обоих случаях очень близки.

Расписание занятий может включать следующую информацию:

* название и номер курса;
* имя преподавателя;
* дни и время встречи;
* местонахождение, например, номер кабинета;
* название предмета.

Расписания занятий могут также включать сведения о формате занятия, например, очная форма обучения, онлайн или их сочетание. Расписание занятий обычно распространяется среди студентов в начале семестра и может быть изменено из-за непредвиденных обстоятельств или обновлений.

Виды расписания по способу представления информации:

* расписание, которое распечатывается на бумаге и распространяется среди учащихся, учителей (преподавателей) и других заинтересованных лиц. Этот тип расписания удобен для предоставления физической ссылки, к которой можно легко получить доступ и обратиться к ней, но может потребоваться больше усилий для обновления и распространения;
* расписание, которое представлено в электронном виде либо на веб-сайте, либо через программное приложение. Этот тип расписания удобен для предоставления более динамичной и интерактивной справочной информации, которую можно легко обновлять и получать к ней доступ, но может потребоваться доступ к компьютеру или другому устройству;
* расписание, которое представлено через мобильное приложение или другую мобильную платформу, что позволяет получить к нему доступ со смартфона или другого мобильного устройства. Этот тип расписания удобен для предоставления удобного и переносимого справочника, к которому можно легко получить доступ на ходу, но для которого может потребоваться подключение для передачи данных или другие ресурсы;
* расписание, представленное в комбинации печатного и электронного форматов, что позволяет получить к нему доступ с помощью различных средств. Этот тип расписания удобен для обеспечения гибкости и доступности, но может потребовать больше усилий для координации и обслуживания.

Также расписание может быть представлено в различных форматах:

* таблица: расписание, представленное в виде таблицы с различными столбцами, такими как время, место, участники;
* график: расписание, представленное в виде графика, где каждое событие представлено в виде точки на осях времени и темы;
* список: расписание, представленное в виде простого списка событий с указанием времени и места каждого события;
* календарь: расписание, представленное в виде календаря, где каждое событие отображается в соответствующую дату;
* недельное расписание, где каждая колонка – учебная группа, строки это конкретный временной отрезок, группа строк – один день.

Расписание может быть составлено на различный период времени:

* еженедельное расписание: составляется каждую неделю;
* расписание семестра: это расписание, в котором излагаются занятия, которые необходимо выполнить в течение семестра или академического семестра. Это может включать время занятий, перерывы, экзамены и другие действия, такие как проекты или презентации;
* расписание курса: это расписание, в котором указаны занятия и задачи, которые необходимо выполнить в течении курса.

Составление расписания занятий в образовательных учреждениях среднего профессионального образования осуществляет учебная часть данной организации.

Работа по составлению расписания проводится заведующим учебной частью, либо курирующим учебную работу заместителем директора при отсутствии должности заведующего учебной частью, на основе сведений о распределении учебной нагрузки между педагогическими работниками колледжа.

При составлении расписания учитывается участие педагогических работников в научной, учебно-методической и воспитательной работе структурного подразделения колледжа, реализующего образовательные программы СПО. Для внешних совместителей и педагогических работников, привлеченных на основе договора гражданско-правового характера, учитывается занятость по основному месту работы.

При составлении расписания учитываются факторы с точки зрения возможности проведения занятий в заданный промежуток времени, в конкретном учебном кабинете и с определенным преподавателем, учитывая учебную нагрузку на группы и количество часов по отдельным учебным предметам.

Методы составления расписания:

* создание расписания вручную путем записи сведений о каждом занятии на листе бумаги или в электронной таблице.
* использование предварительно разработанного шаблона для создания расписания, например таблицы или шаблона календаря в текстовом процессоре или программе для работы с электронными таблицами.
* использование специализированного программного обеспечения, предназначенного для создания расписаний, такого как программное обеспечение для управления проектами или приложения-календари.
* использование онлайн-инструментов, позволяющих создавать расписание путем перетаскивания событий в представление календаря.

При использовании специального программного обеспечения расписание составляется на основе базы данных, которую пользователь изначально наполняет необходимыми параметрами, выполнив следующие шаги:

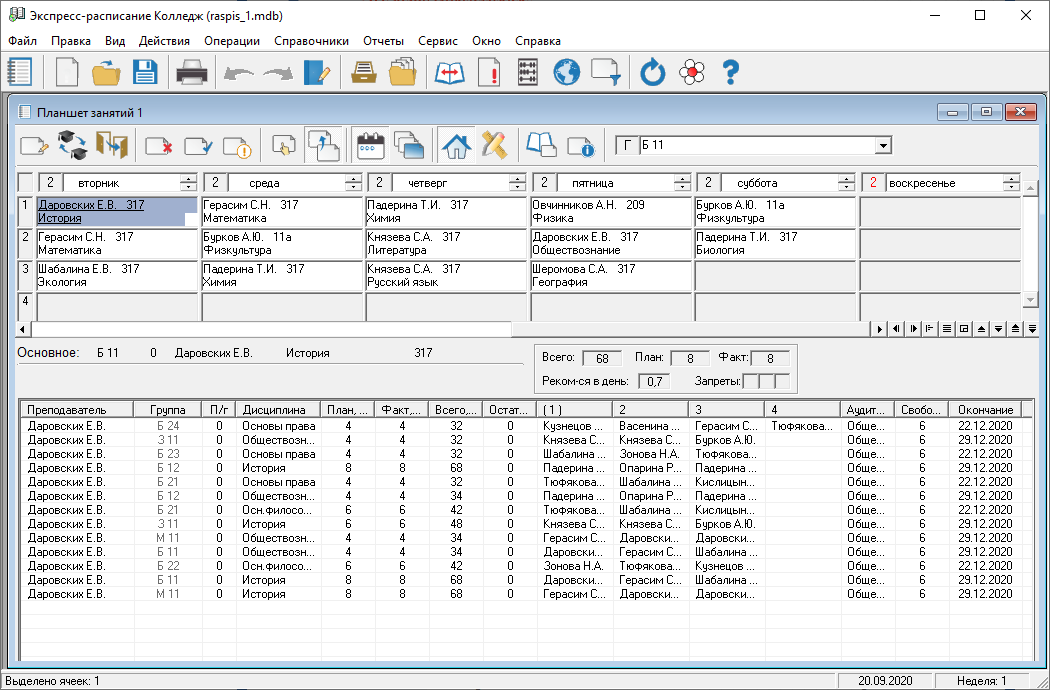
* сбор информации: первым шагом в создании расписания занятий является сбор всей необходимой информации, включая названия и номера групп, имена преподавателей, дни и время собраний, аудитории и предметы.
* после того, как вся необходимая информация будет собрана, ее можно ввести в программное обеспечение. Может включать создание отдельных записей для каждой группы и ввод соответствующих сведений;
* установка ограничений и предпочтений: программное обеспечение может позволить пользователям устанавливать ограничения и предпочтения, которые помогают гарантировать, что расписание выполнимо и соответствует их потребностям. Например, пользователи могут указать, что определенные занятия должны или не должны быть запланированы на определенное время, или что они предпочитают, чтобы определенные занятия были запланированы на определенное время;
* настройка представления расписания: после ввода всей необходимой информации и настройки любых ограничений и предпочтений программное обеспечение может создать один или несколько вариантов расписания. Расписание может отображаться в виде календаря или списка, можно просматривать и сравнивать для определения наилучшего соответствия;
* доработка расписания: после того, как желаемое расписание определено, его можно просмотреть и доработать. Этот шаг может включать в себя внесение любых необходимых корректировок или изменений и обеспечение правильности всей необходимой информации.

Когда расписание готово, оно публикуется, чтобы учащиеся, преподаватели и другие заинтересованные стороны могли получить к нему доступ. Это может включать экспорт расписания в другой формат, например, PDF или электронную таблицу, или предоставление доступа к нему в Интернете через веб-интерфейс.

## Обзор аналогов

В настоящее время существует большое количество автоматизированных информационных систем, осуществляющих решение задачи автоматизации составления расписания в образовательном учреждение.

«Экспресс-расписание Колледж» - программа для автоматизации составления расписания учебных занятий в училищах, колледжах и профессиональных лицеях. Программа автоматически составляет основное расписание, позволяет вести учет выполненных часов, ежедневные изменения расписания, формирует разнообразные отчеты. Интерфейс программы представлен на рисунке 1.



1. Интерфейс программы «Экспресс-расписание Колледж»

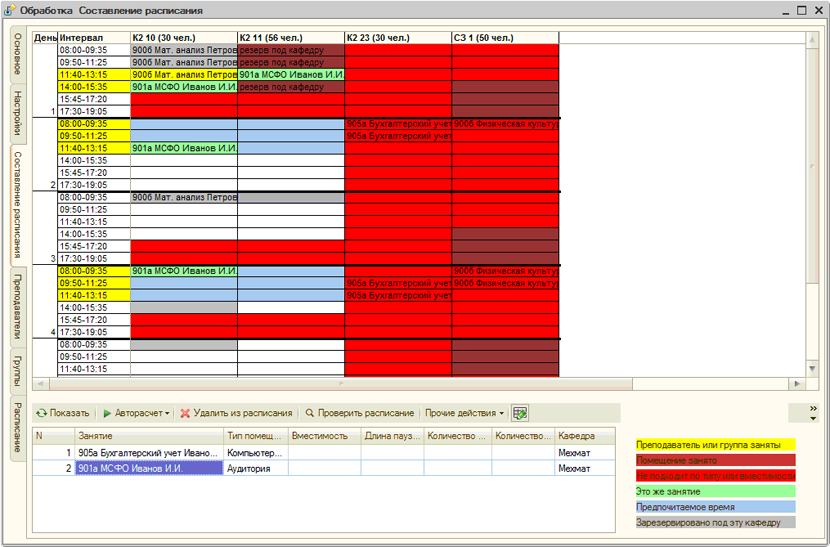
Преимущества:

* при составлении расписания программа помогает избежать накладок;
* можно в любое время отследить нагрузку по преподавателю, по группе, по аудитории;
* недорогая и работает в сети.

Недостатки:

* не может учесть специфику некоторых учреждений, расписание приходится составлять вручную.

«1С:Автоматизированное составление расписания. Колледж» - продукт является самостоятельным программным продуктом, предназначенным для решения задач автоматизированного составления учебных расписаний и оперативного управления помещениями в образовательных организациях среднего профессионального образования. Интерфейс программы представлен на рисунке 2.



1. Интерфейс программы «1С:Автоматизированное составление расписания. Колледж»

Преимущества:

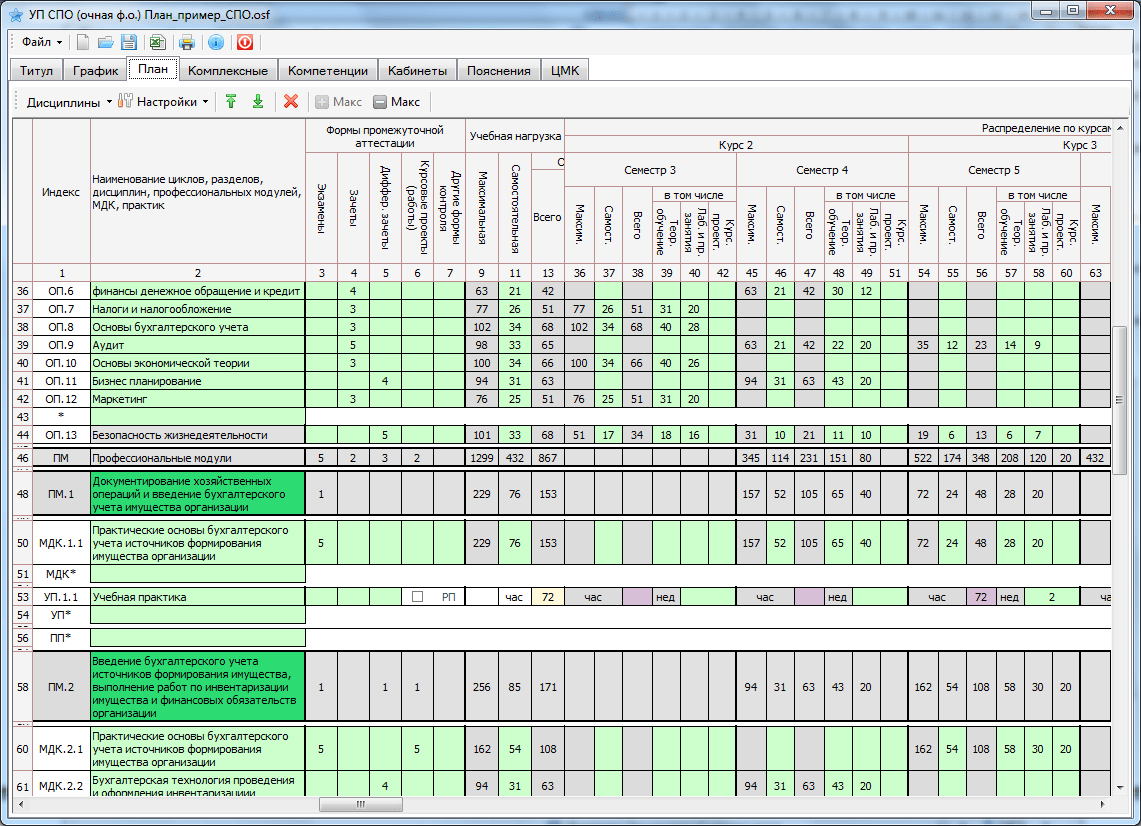
* решение прорабатывается под конкретную организацию;
* интегрируется с другими решениями на базе 1С.

Недостатки:

* высокая стоимость;
* отсутствует удобная для интеграции программа разработки учебных планов.

Система «АВТОРасписание» предназначена для быстpого, удобного и качественного составления расписаний занятий и сопровождения их в течение всего учебного года. Программа достаточно проста в освоении. Имеется подробное руководство пользователя и справочная система, где описаны все возможности и способы работы с программой.

Программа отличается уникальным, мощным алгоритмом построения и оптимизации расписания. Этот алгоритм является оригинальной авторской разработкой. Он позволяет находить оптимальные решения даже при очень сложных исходных данных. Интерфейс программы представлен на рисунке 3.



1. Интерфейс программы «АВТОРасписание»

Преимущества:

* сохранение расписание в различных форматах;
* возможность выбора режима составления расписания.

Недостатки:

* автоматически расписание составляется долго, и требует большой корректировки вручную;
* программа требует достаточно продвинутых способностей от завуча.

# Специальная часть

## Аналитическая часть

### Постановка задачи

### Разработка диаграмм

### Обоснование выбора программных средств

### Обоснование выбора языка программирования

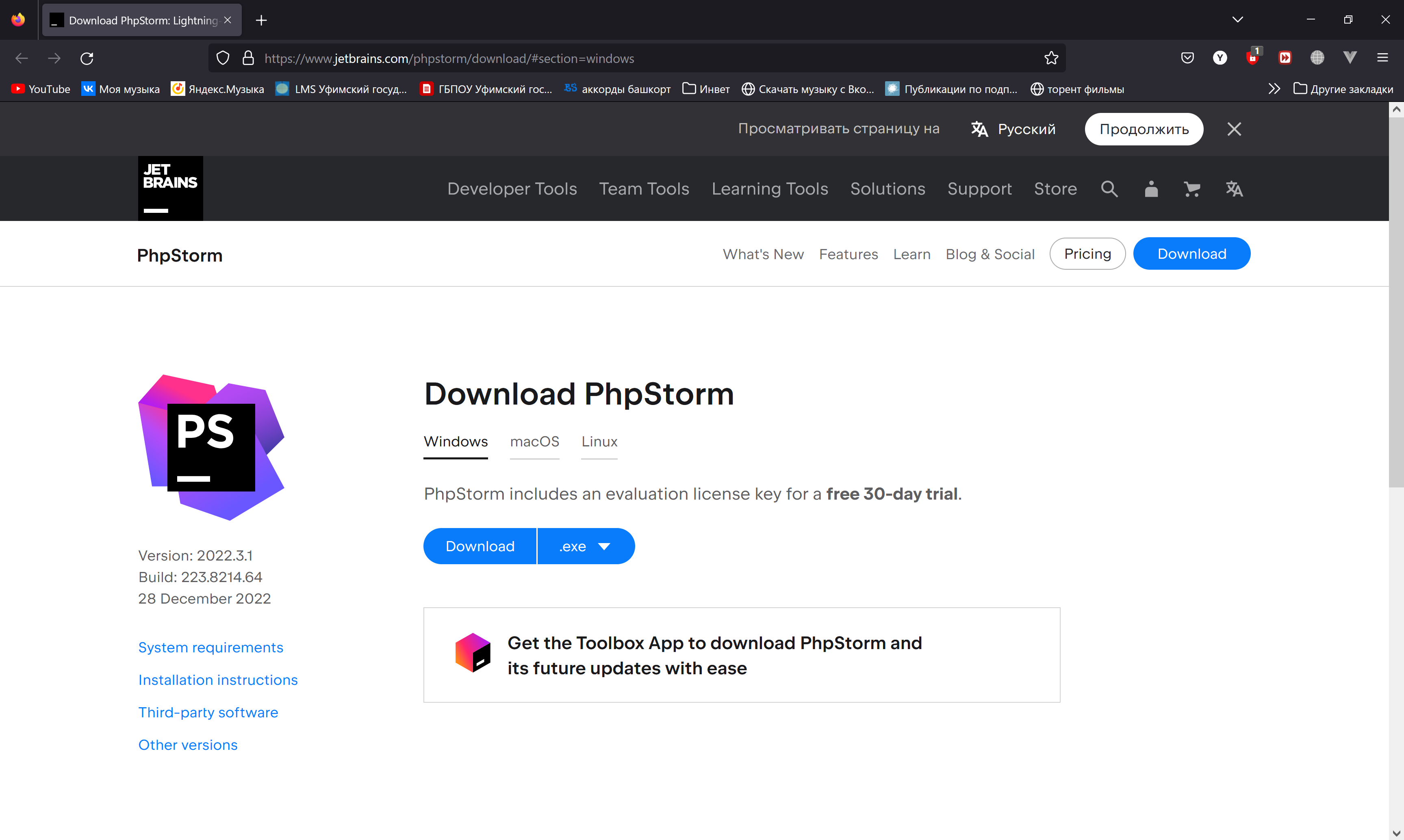
## Практическая часть

### Описание процесса разработки веб-приложения

Исходя из поставленной задачи были проведены следующие работы:

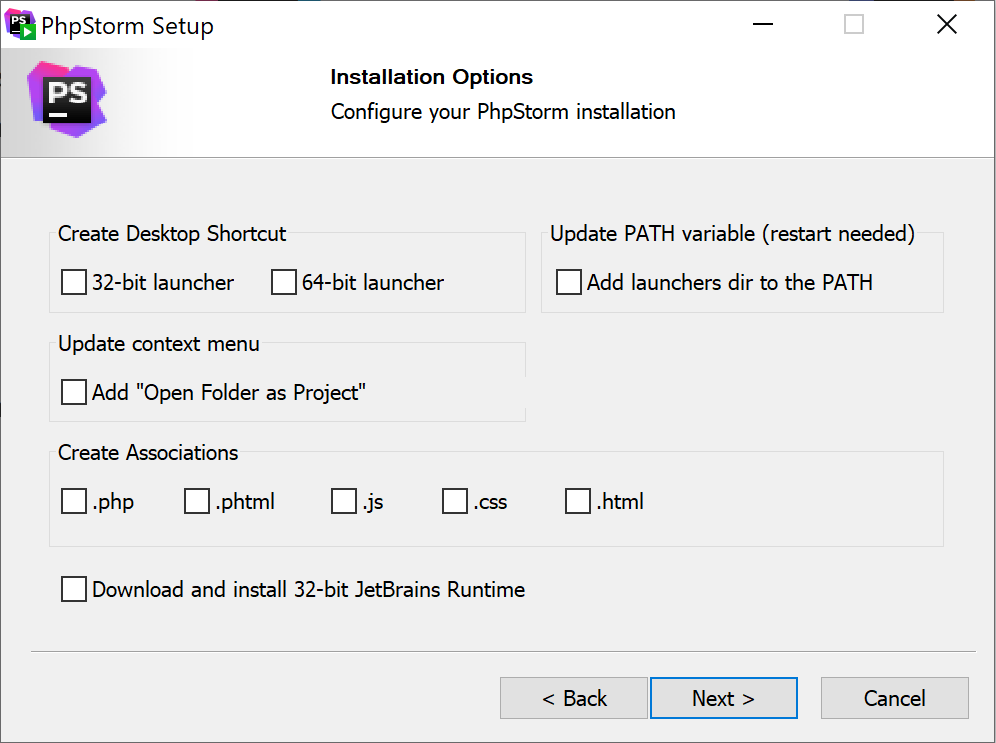
* загрузка и установка интегрированной среды разработки JetBrains PhpStorm;
* настройка интегрированной среды разработки JetBrains PhpStorm;
* загрузка и установка программы Open Server, реализующей функции для локальной разработки веб-приложений, а именно: веб-сервер, сервер базы данных, интерпретатор языка программирования PHP;
* настройка модулей Open Server, выбор версии языка программирования PHP, выбор веб-сервера, выбор модуля СУБД.
* создание нового проекта;
* настройка зависимостей и установка программного обеспечения для разработки приложения с помощью терминального пакетного менеджера «composer»;
* конфигурирование проекта и фреймворка Laravel;
* настройка зависимостей и установка фреймворка Vue JS, вспомогательной библиотеки Vuex с помощью пакетных менеджеров «composer» и «npm»;
* разработка программного кода;
* тестирование и отладка приложения.

Для загрузки интегрированной среды разработки JetBrains PhpStorm необходимо перейти на официальный сайт разработчиков, перейти в раздел загрузки и начать скачивание. Важно, что разработчики распространяют свою среду разработки на платной основе. Имеется 30-дневный пробный период, чего вполне достаточно для разработки. На рисунке ? представлена страница для загрузки JetBrains PhpStorm.

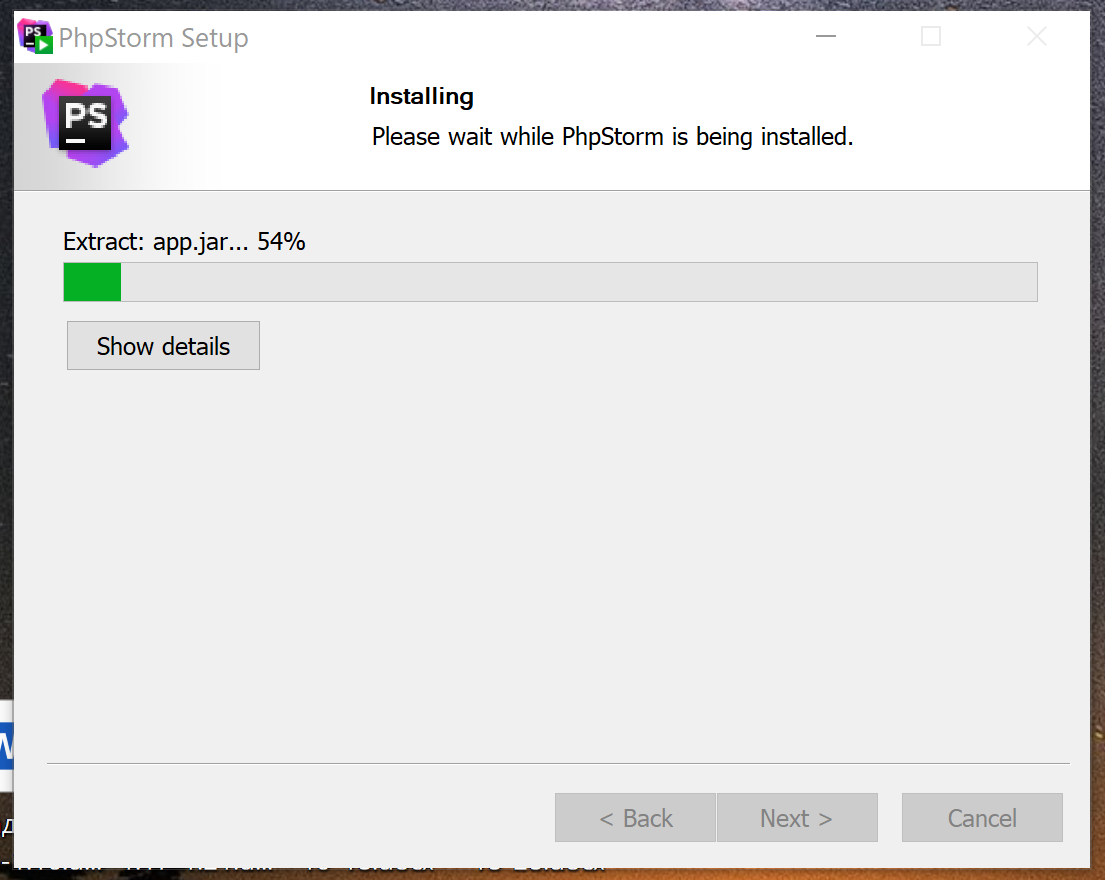


1. Страница загрузки JetBrains PhpStorm

После скачивания установочной программы с расширением «exe» был произведен запуск установки интегрированной среды разработки JetBrains PhpStorm. На рисунках ? и ? изображены процесс настройки комплектации среды разработки и установки.

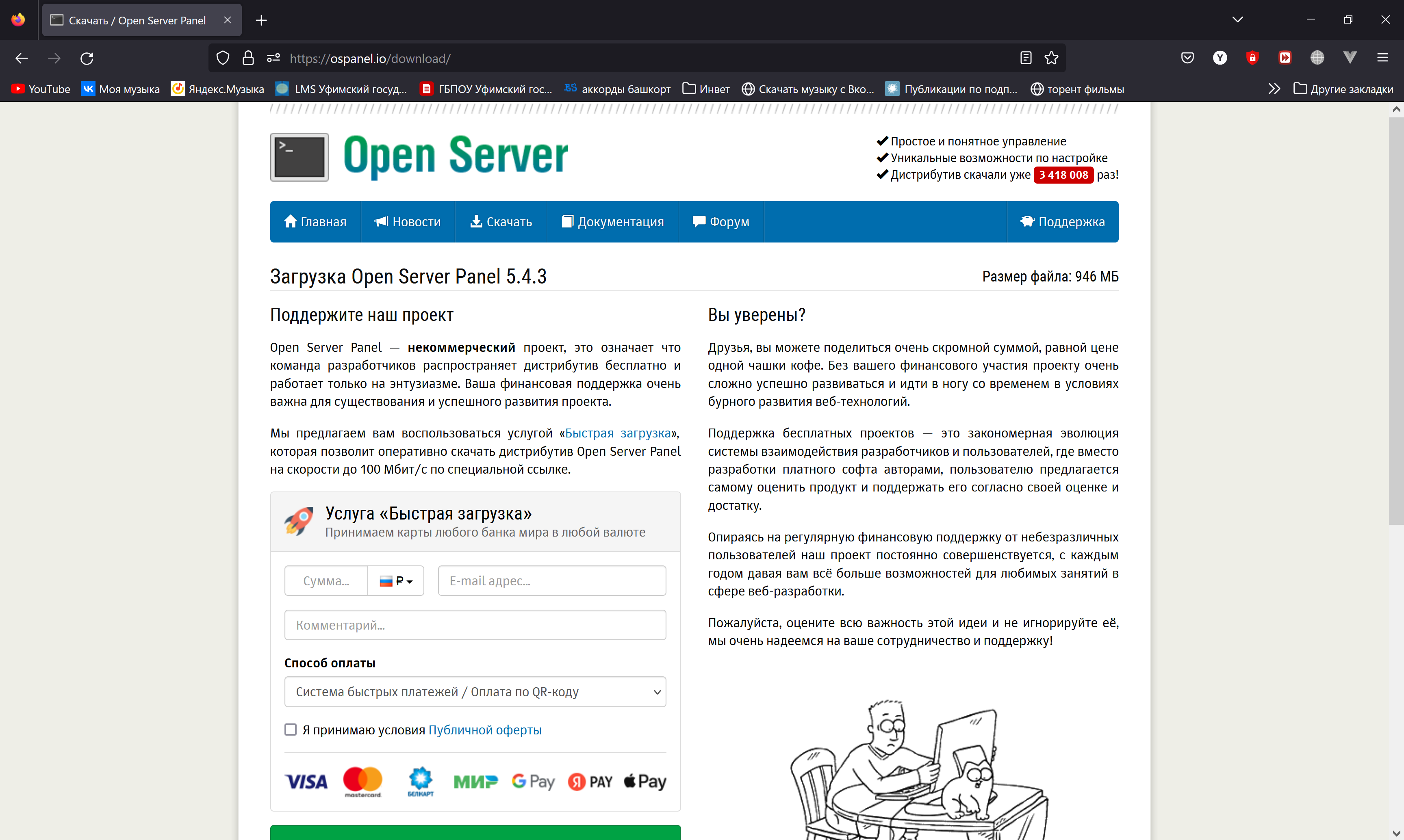


1. Настройка комплектации JetBrains PhpStorm



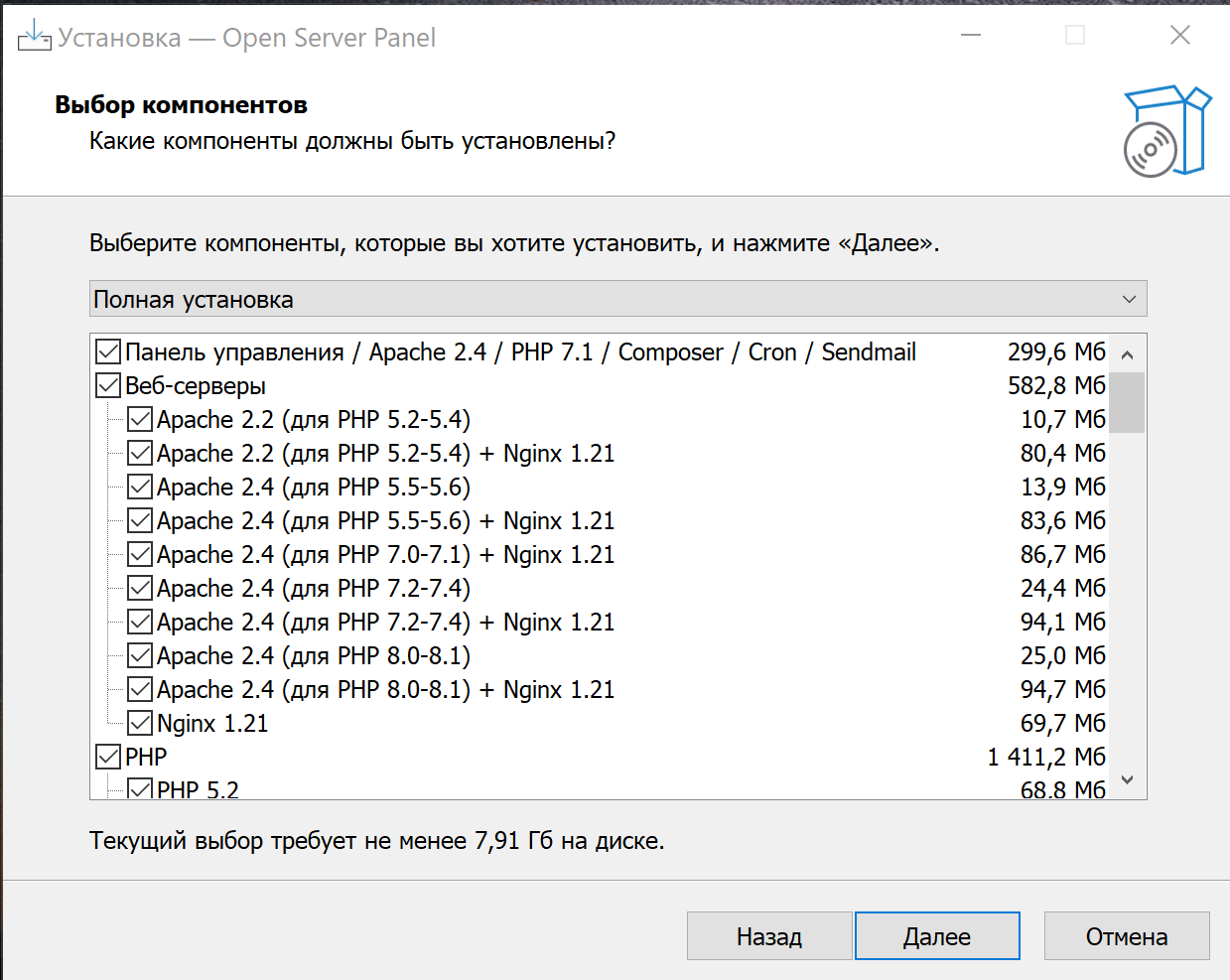
1. Окно установки JetBrains PhpStorm

Для разработки и тестирования веб-приложения требуется интерпретатор языка программирования PHP, веб сервер и сервер базы данных. Программа Open Server для локальной разработки веб-приложений предоставляет все эти функции из коробки. На рисунке ? изображена страница загрузки установочного файла данной программы на официальном сайте разработчиков.



1. Страница загрузки OpenServer

После выполнения загрузки установочного файла следует установка и конфигурирование программы Open Server. Конфигурирование представлено на рисунке ?.



1. Конфигурирование Open Server

На рисунке ? представлен процесс установки Open Server.

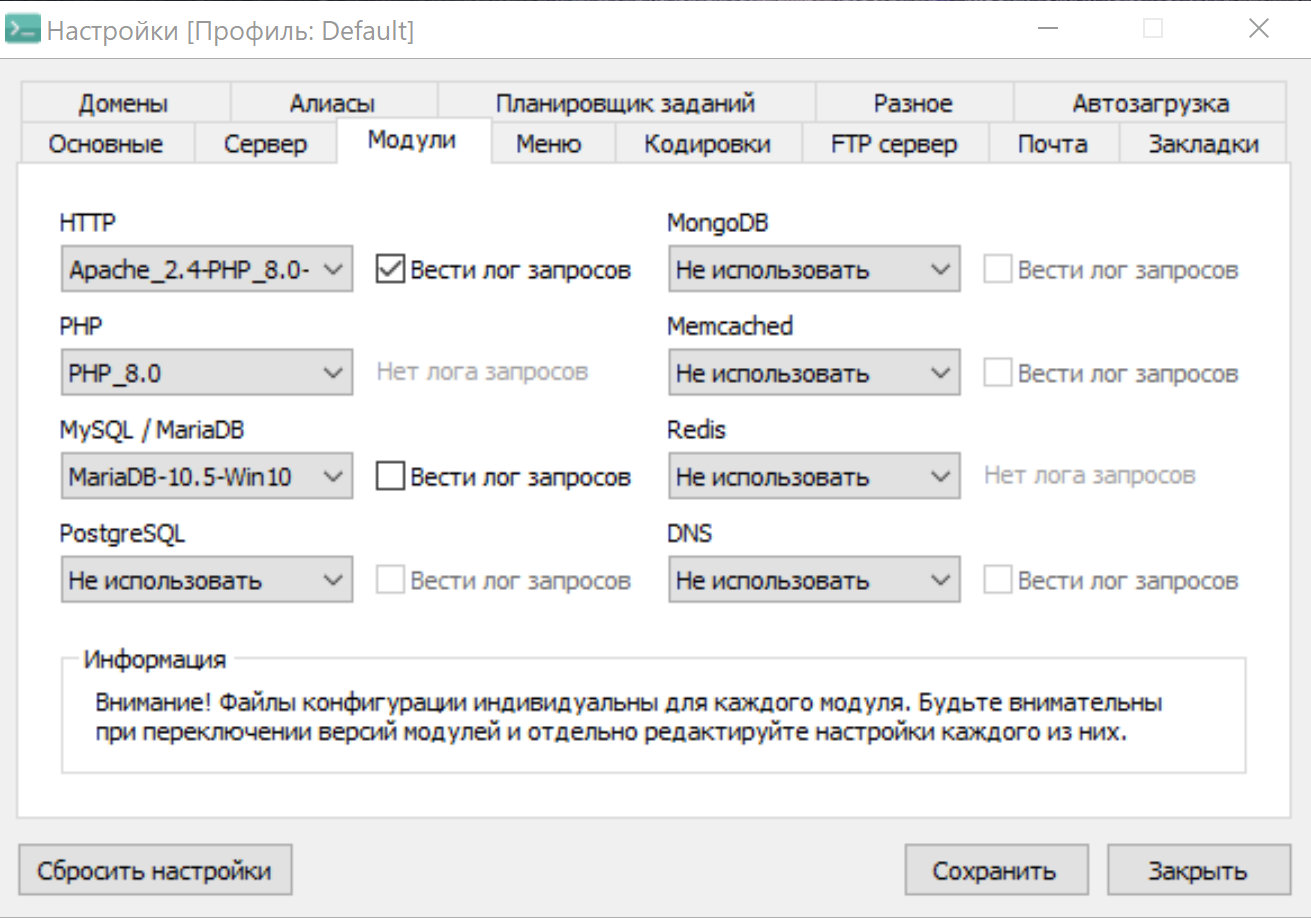


1. Процесс установки Open Server

Программа Open Server включает в себя множество модулей веб-серверов, серверов баз данных разных версий. Поэтому требуется настройка требуемых версий модулей для разработки веб-приложения «Advanced Schedule», а именно:

* веб-сервер Apache версии 2.4;
* интерпретатор языка PHP версии 8.0;
* сервер реляционной базы данных MariaDB версии 10.5.

На рисунке ? изображено окно настройки Open Server.

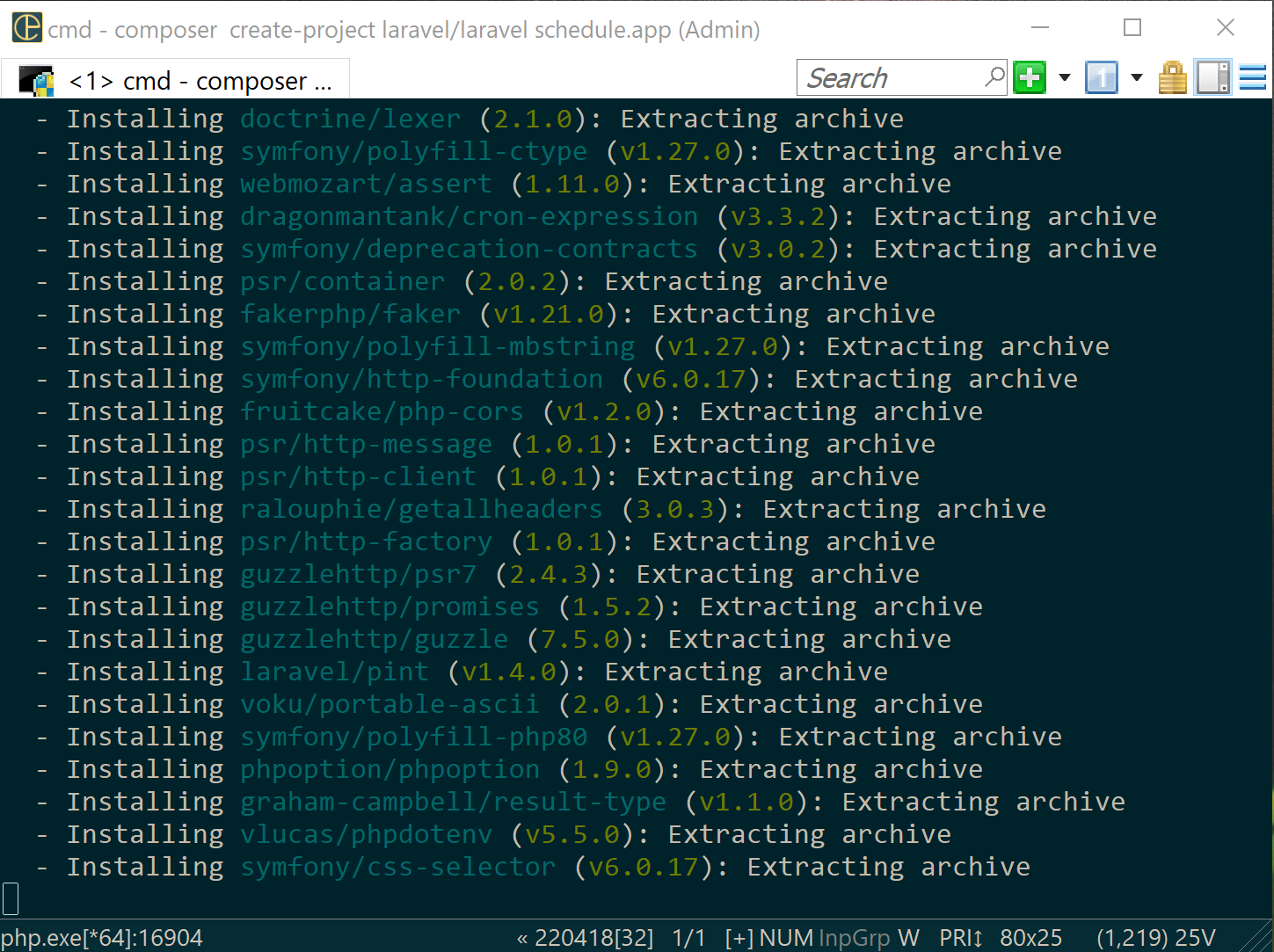


1. Окно настройки Open Server

После завершения настройки Open Server необходимо создать проект в директории «domains» данной программы с помощью пакетного менеджера Composer, выполнив следующие шаги:

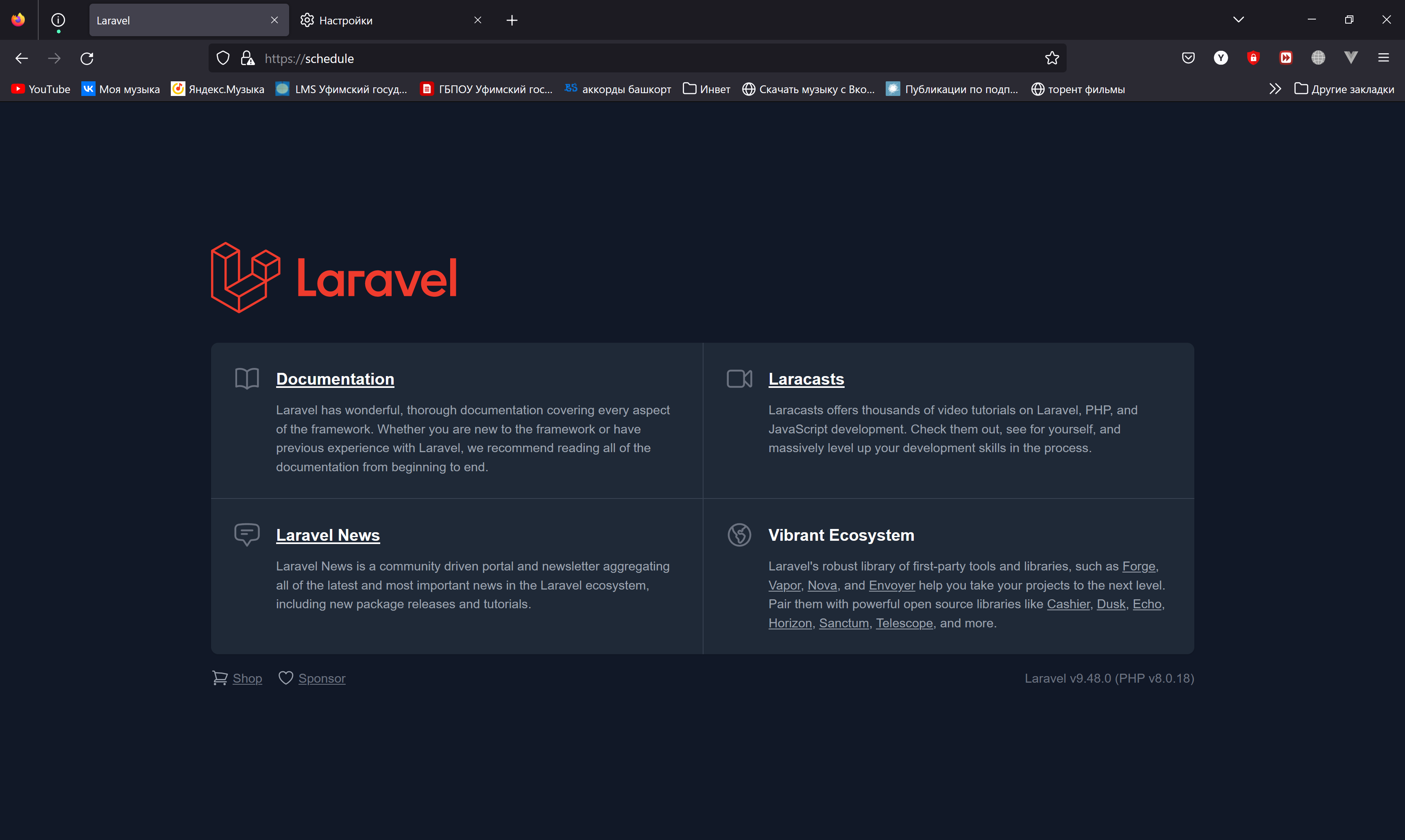
* запустить Open Server и открыть консоль поставляемую вместе с данной программой;
* в консоли перейти в директорию «domains»;
* ввести команду «composer create-project laravel/laravel schedule.app», где «laravel/laravel» обозначает создать пустой проект с фреймворком Laravel, а «schedule.app» название директории, в которую будет помещен созданный проект;

На рисунке ? показан процесс создания нового проекта.



1. Процесс создания нового проекта и загрузки зависимостей с помощью пакетного менеджера Composer

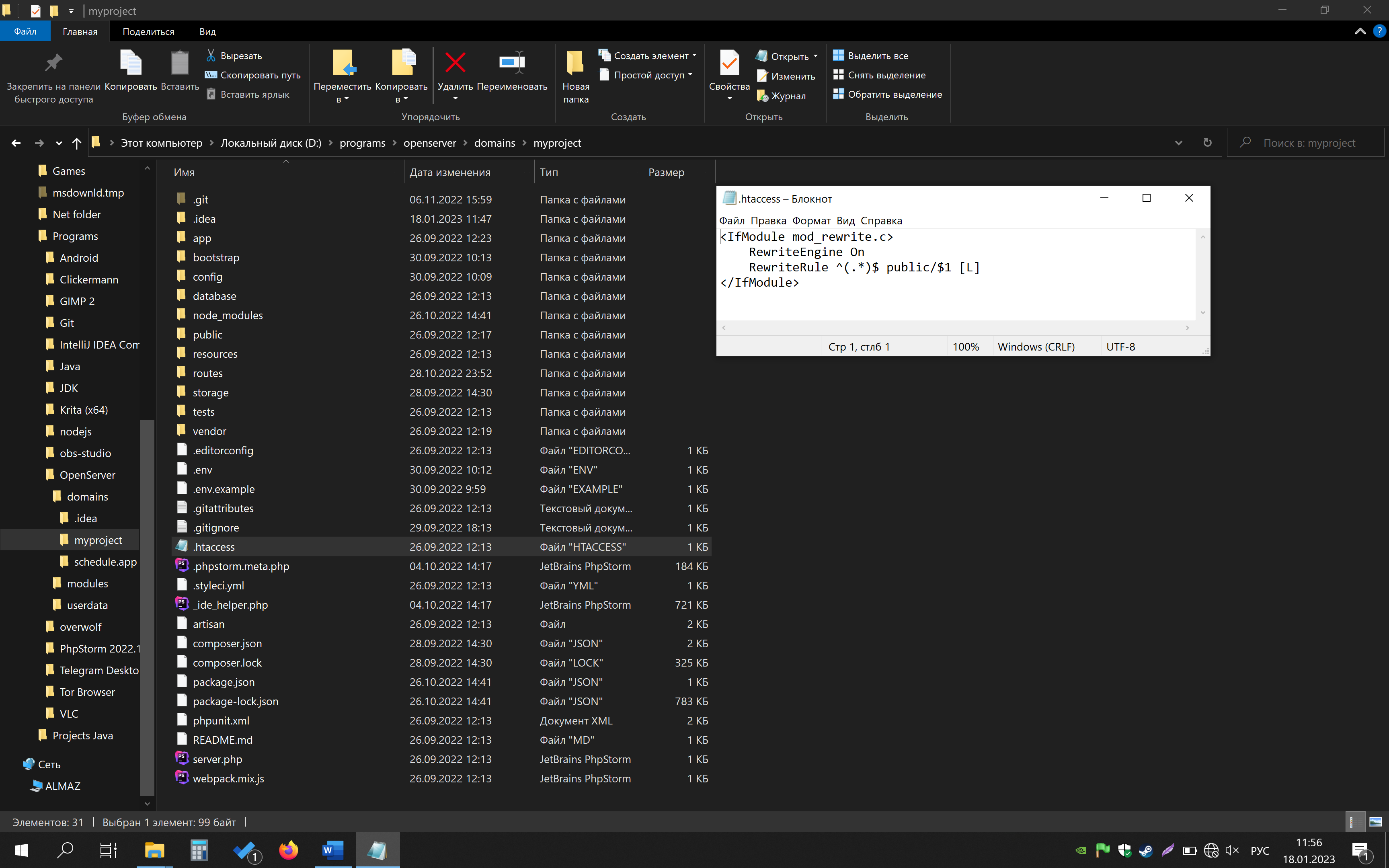
После выполнения установки можно открыть в браузере внешний вид нового проекта. Графический интерфейс нового проекта изображен на рисунке ?.



1. Внешний вид нового проекта

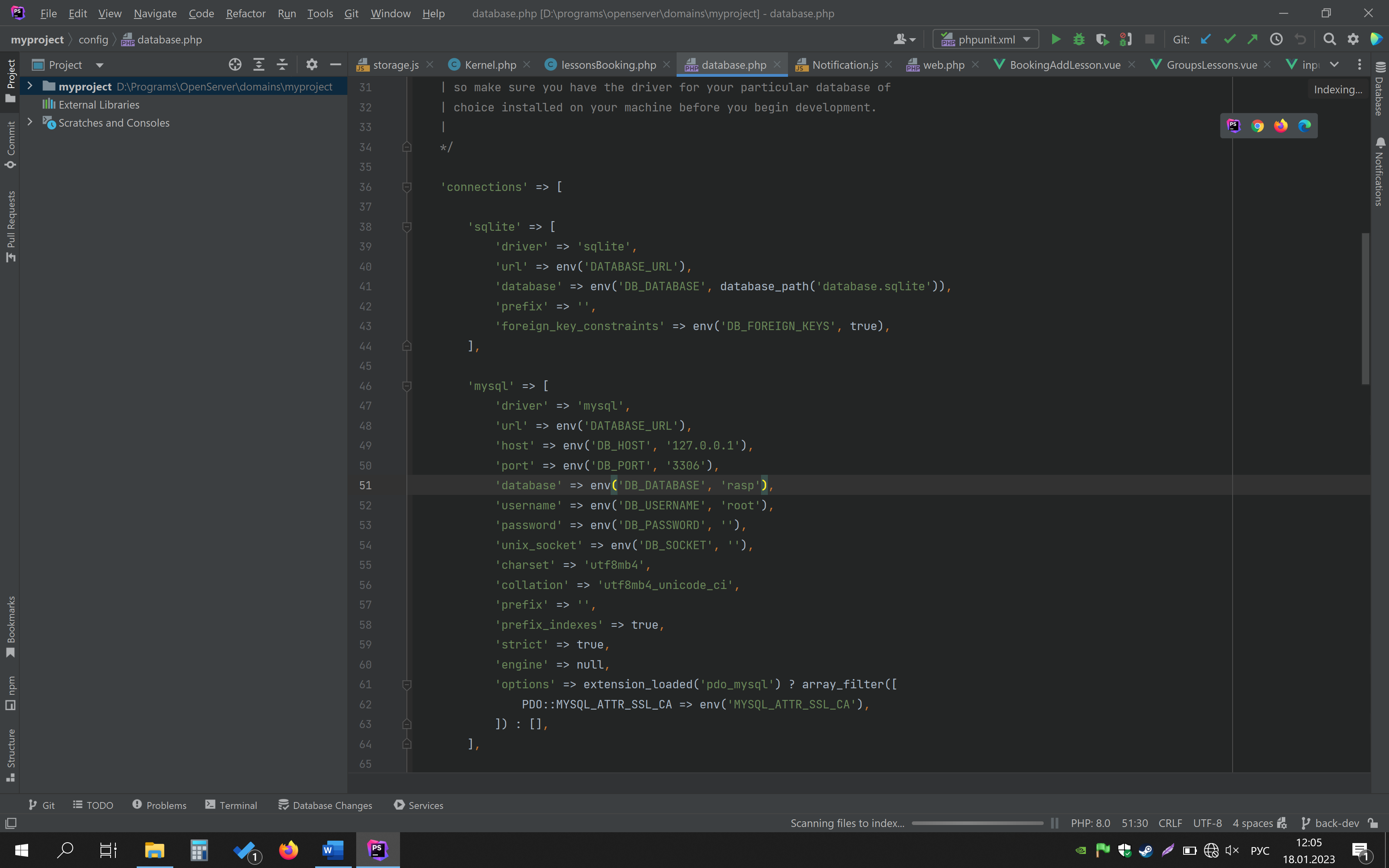
Далее необходимо настроить конфигурацию проекта для дальнейшей работы. Необходимо создать и\или отредактировать существующие файлы в проекте, перечисленные ниже:

* создать файл «.htaccess» в корне проекта, данный файл необходим для перенаправления запросов в необходимый файл, для того, чтобы фреймворк Laravel мог его принять, и ввести в нем необходимый код, который представлен на рисунке ?;



1. файл «.htaccess»

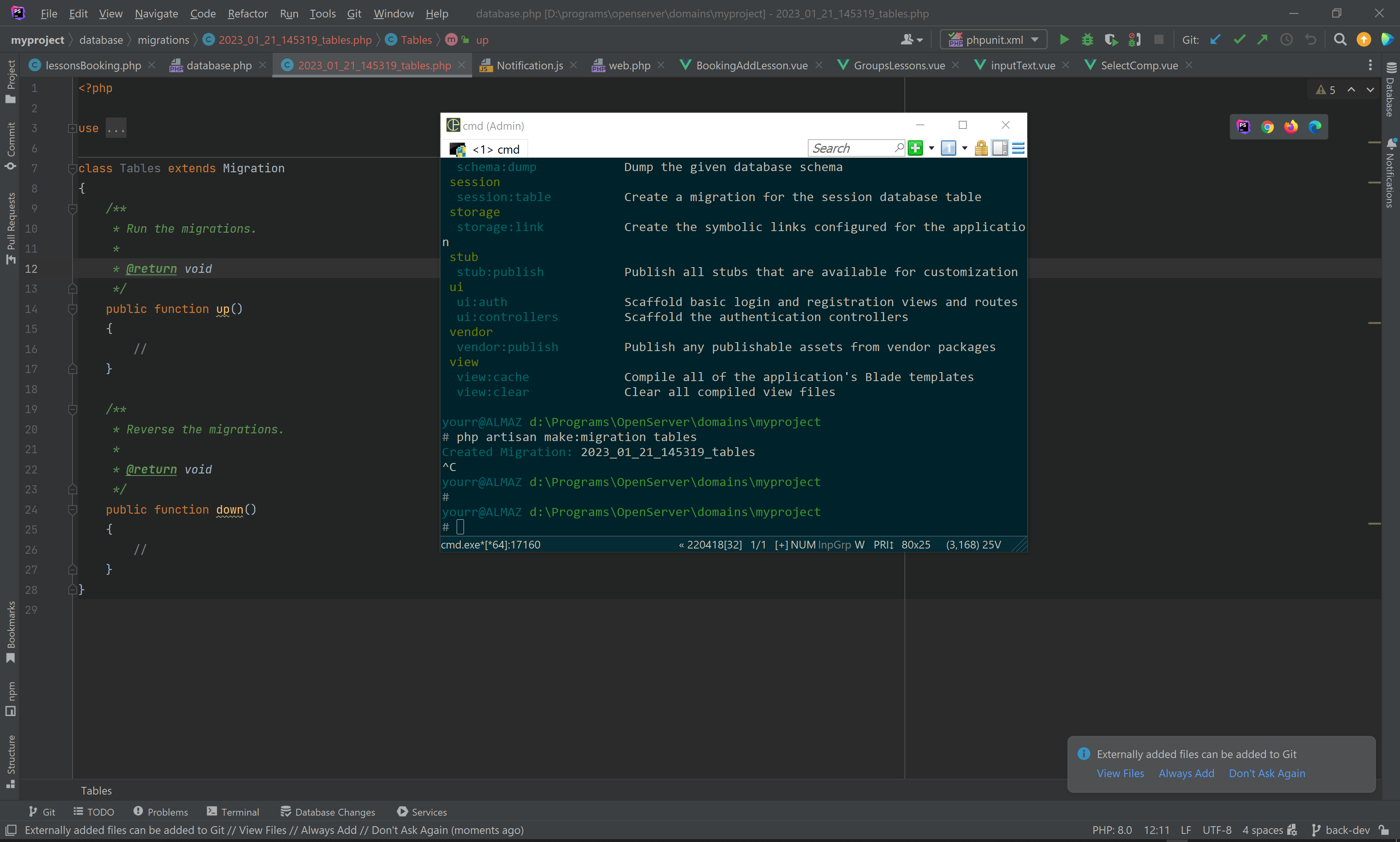
* отредактировать файл «database.php» в директории «config», данный файл определяет подключение к базе данных, в нем необходимо ввести информацию для доступа и авторизации в базе данных, файл представлен на рисунке ?;



1. файл «database.php»

Первоначальная настройка завершена. Фреймворк Laravel действует по шаблону проектирования MVC – Model View Controller. Это методология декомпозиции логики приложения на отдельные логические элементы, отвечающие за свою предметную область. Model отвечает за работу с данными - запись, удаление, вывод. Controller принимает запросы, проверяет входные данные и обращается к Model для какого-либо изменения информации в базе данных, либо для запроса данных из базы данных, затем передаёт данные во View. View отвечает за предоставление данных для пользователя, оно принимает чистую информацию и приводит в графический вид, который пользователь способен воспринимать. Также фреймворк разделяет маршрутизацию запросов. За это отвечает логический элемент «маршрутизатор», в нем задают правила направления запросов.

Первым этапом в разработке программного кода веб-приложения является составление структуры базы данных. Во фреймворке Laravel это осуществляется с помощью инструмента миграций. В файлах миграций задаётся структура таблиц и их связи. Для того, чтобы создать файл миграции, фреймворк Laravel предоставляет специальный помощник в терминале – «artisan»; необходимо обратиться к нему с соответствующей командой. На рисунке ? изображен процесс создания файла миграции и как выглядит данный файл.



1. Создание файла миграции

В файле миграции присутствуют две функции:

* up() – отвечает за действия, которые будут совершаться при выполнении миграции, это может быть создание таблиц;
* down() – отвечает за действия, которые будут совершаться при отмене миграций, это может быть удаление созданных таблиц при выполнении миграций.

На рисунке ? отображено каким образом заполняется файл миграции.



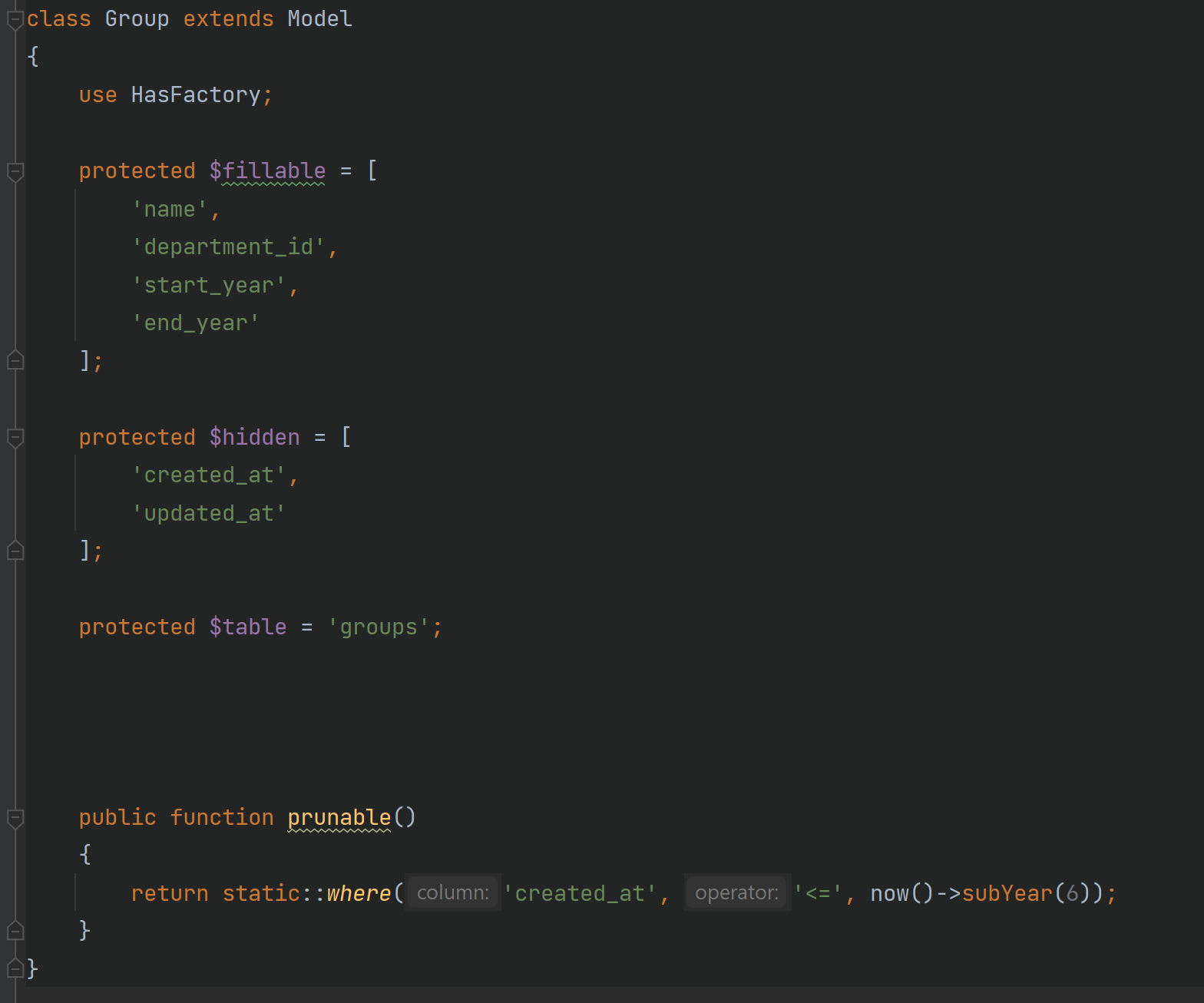
1. Процесс заполнения файла миграции

После того, как структура базы данных задана, необходимо описать модели для работы с каждой таблицей, определить в них необходимые функции, которые позволят взаимодействовать с данными.

Хотелось бы отметить возможности моделей:

* в случае, если какие-либо данные, которые используются системой, необходимо скрыть, то можно указать защищенное свойство «$hidden» и передать в него массив, состоящий из строк с названиями колонок таблицы, которые необходимо скрыть;
* для автоматического удаления устаревших данных имеется специальный метод «prunable()», если его определить и указать, при каком условии данные необходимо удалить, то система будет автоматически удалять по данным условиям записи связанные с данной моделью.

На рисунке ? изображен процесс создания и пример заполнения моделей.



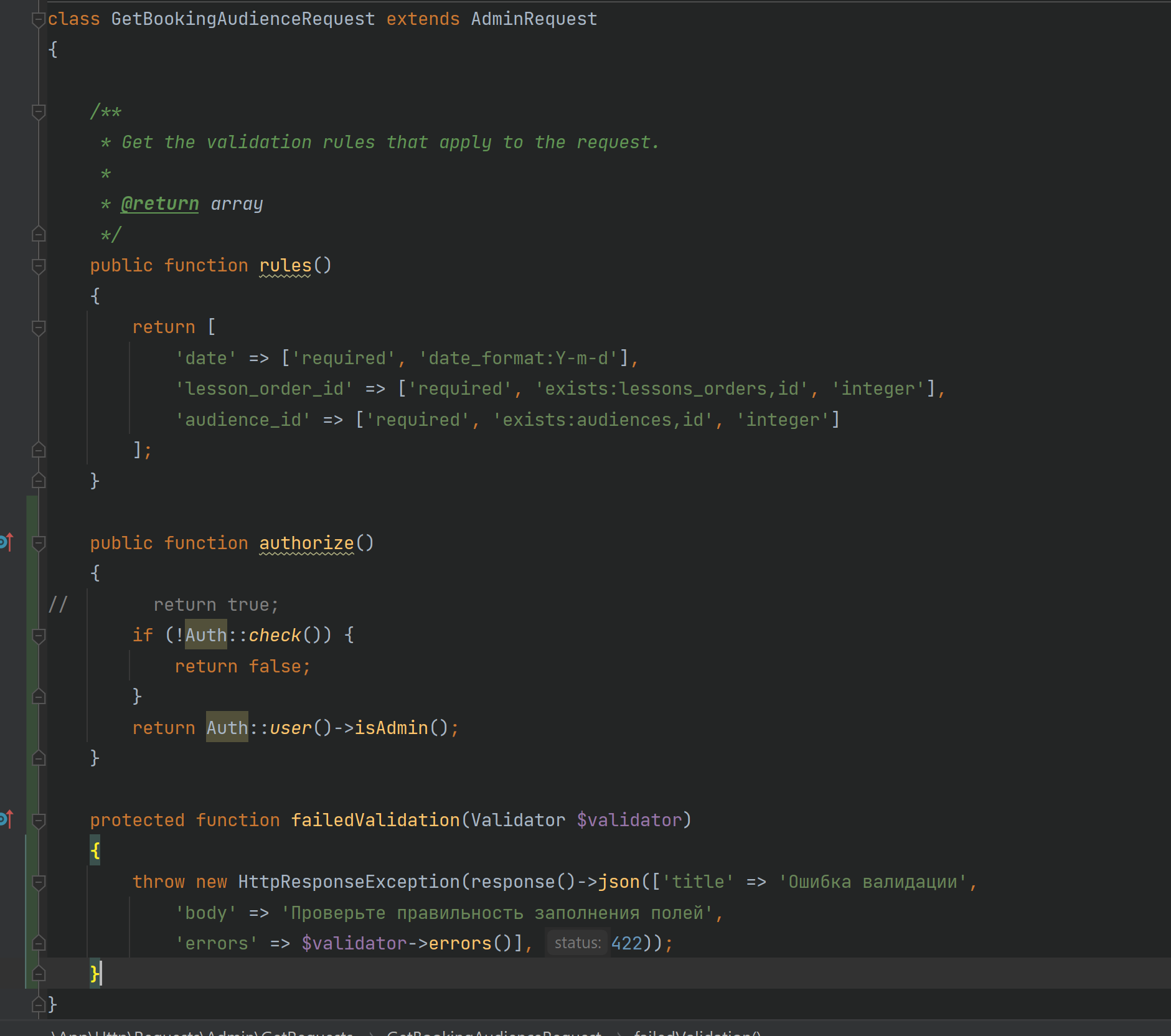
1. Заполнение модели

Модели для работы с базой данных определены. Для того, чтобы сервер принимал и отвечал на запросы необходимы контроллеры и классы-валидаторы, наследующиеся от класса «form request». Валидация возможна и в классах контроллерах, но это приводит к их увеличению, что понижает читабельность кода. Дабы соответствовать принципу единственной ответственности из набора принципов SOLID, необходимо декомпозировать логику обработки данных в классы, отвечающие за одну конкретную задачу.

Структура класса-валидатора:

* метод «rules()» отвечает за правила валидации, он возвращает набор с правилами валидации для каждого параметра в запросе;
* метод «authorize()» отвечает за допуск к выполнению данного запроса, в нем описывается проверка пользователя на право доступа к выполнению данного действия;
* метод «failedValidation()» позволяет указывать ответ, который будет возвращаться, если валидация не пройдена.

На рисунке ? изображен пример класса-валидатора.



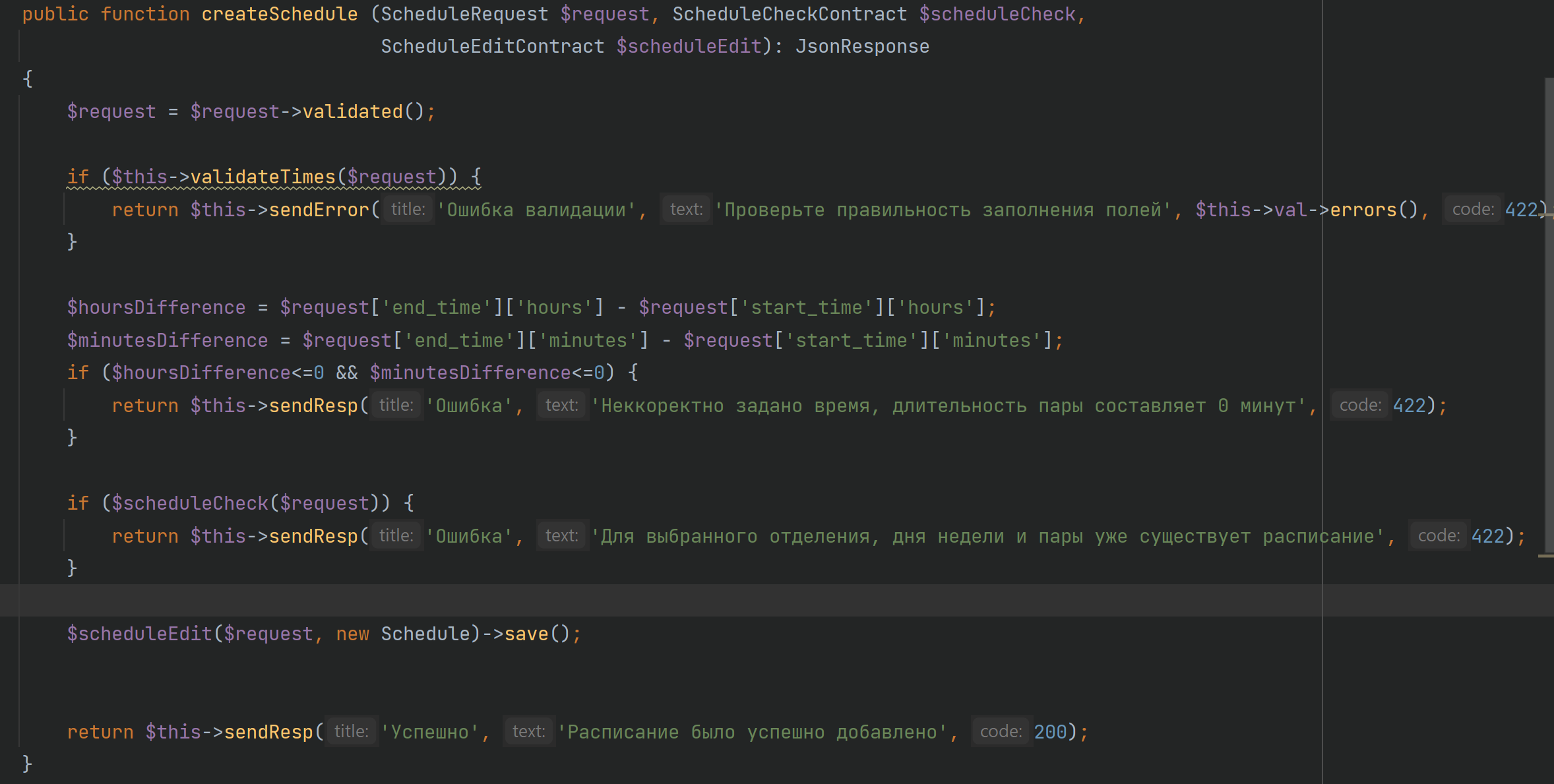
1. Класс-валидатор

Затем в качестве запроса, вместо обычного запроса, контроллер принимает данный класс и вызывает у него метод «validated()», в случае, если валидация прошла, то класс валидатор не выбрасывает исключение, соответственно программный код продолжает выполняться дальше в контроллере. Благодаря декомпозиции, программный код становится более читабельным, на рисунке ? изображен декомпозированный метод контроллера.



1. Декомпозированный метод контроллера

Также в контроллерах могут быть какие-либо сложный для чтения действия, которые следует декомпозировать в отдельные классы-действия или отдельные методы и назвать соответственно той задаче или действию, которые они выполняют. На рисунке ? изображен декомпозированный метод контроллера.

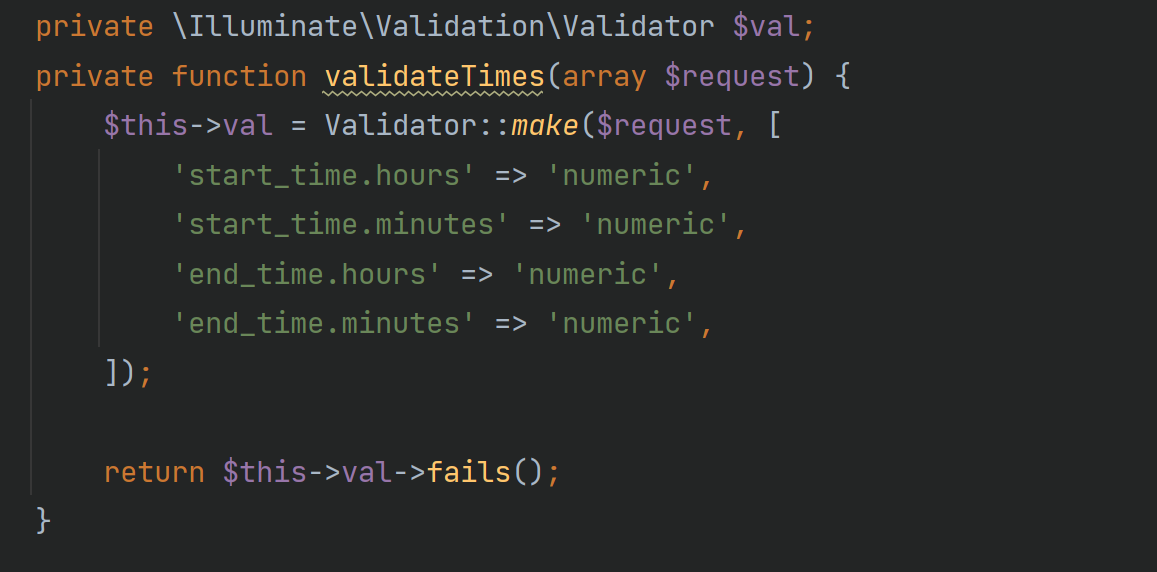


1. Декомпозированный метод контроллера

Были декомпозированы:

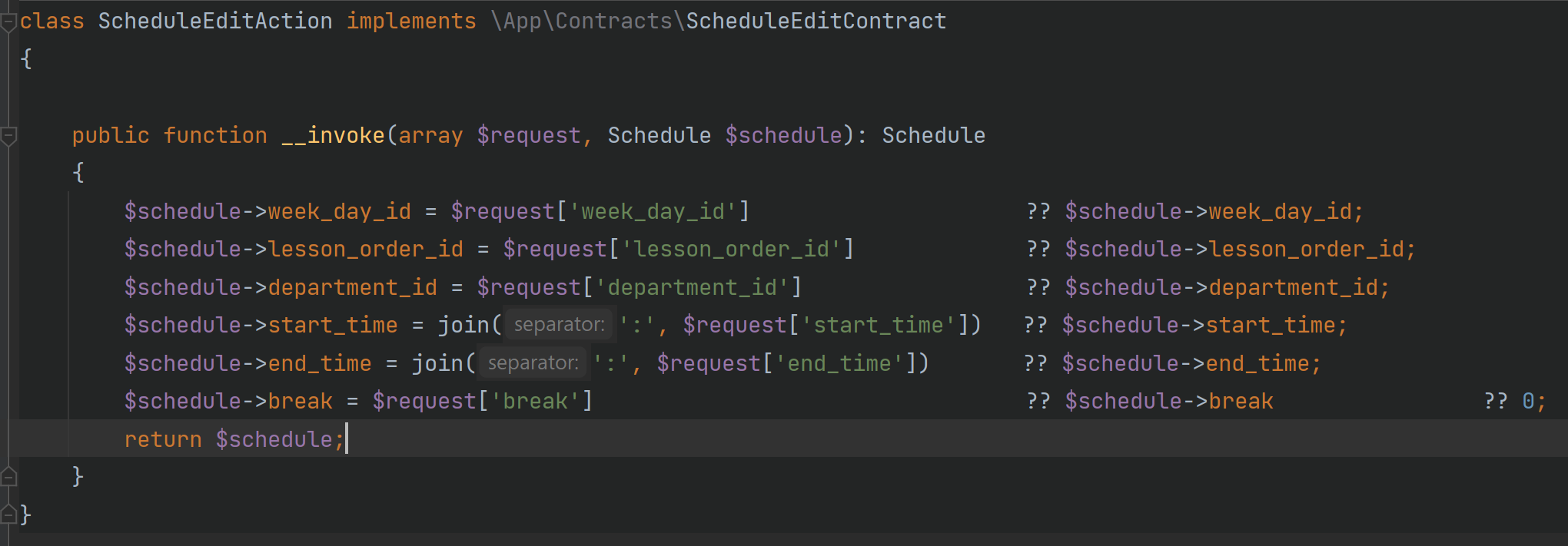
* валидация времени, вынесена в отдельный метод;
* проверка накладки, вынесена в отдельный класс-действие.

На рисунке ? изображен метод валидации времени.



1. Метод валидации времени

На рисунке ? изображен класс-действие, отвечающий за проверку накладок расписания.



1. Класс-действие проверки накладок расписания

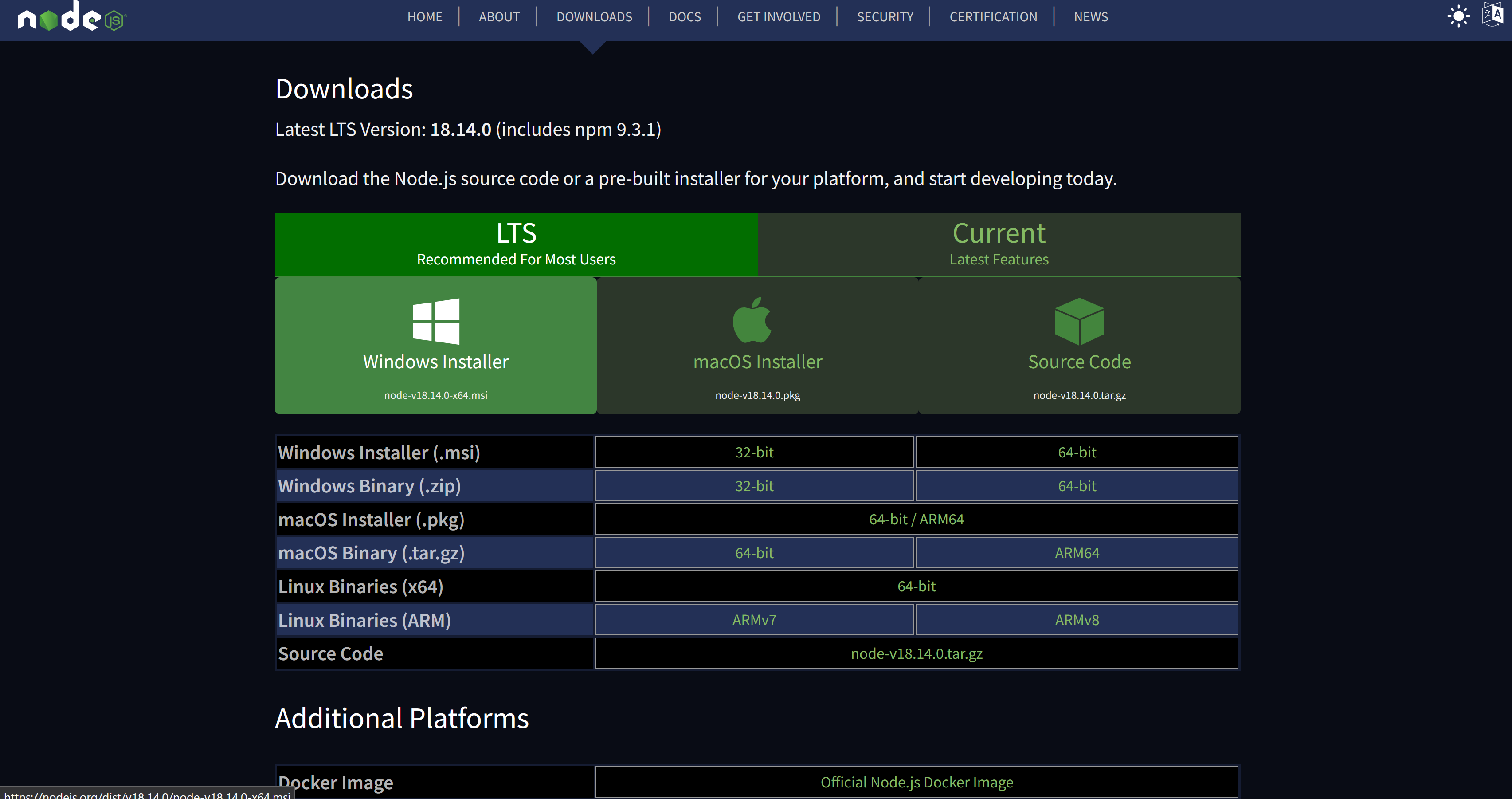
Для того, чтобы запросы проходили через программный код и попадали в нужные методы и в нужные контроллеры, необходимо определить маршруты запросов по программному коду. За маршруты отвечают специальные файлы для маршрутизации, с английского языка «router». Фреймворк Laravel предоставляет несколько маршрутизаторов для различных целей. В случае веб-приложения будет использован соответствующий роутер «web». На рисунке ? представлен маршрутизатор и заданные маршруты запросов.



1. Маршрутизатор запросов

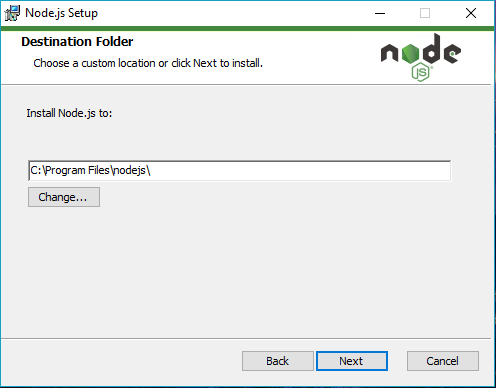
После окончания разработки серверной части веб-приложения была выполнена разработка клиентской части веб-приложения «Advanced Schedule», используя фреймворк «Vue», библиотеку CSS стилей «Bootstrap», библиотеку JS «jQuery».

Перед тем, как приступить к установке вышеперечисленных библиотек требуется установка программы «Node JS» и пакетного менеджера «npm». На рисунке ? представлен процесс скачивания установочного файла.



1. Скачивание установочного файла «Node JS»

Далее, запустив установочный файл, была выполнена настройка установки и запущен процесс установки программы «Node JS» и пакетного менеджера «npm», процесс изображен на рисунке ?.

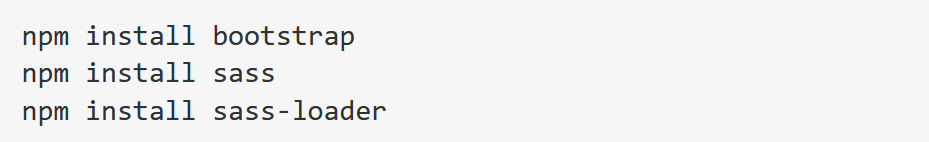


1. Процесс установки «Node JS» и «npm»

Для установки набора библиотек для разработки были введены команды изображенные на рисунках ?, ? и ?.



1. Команды для установки фреймворка «Vue»



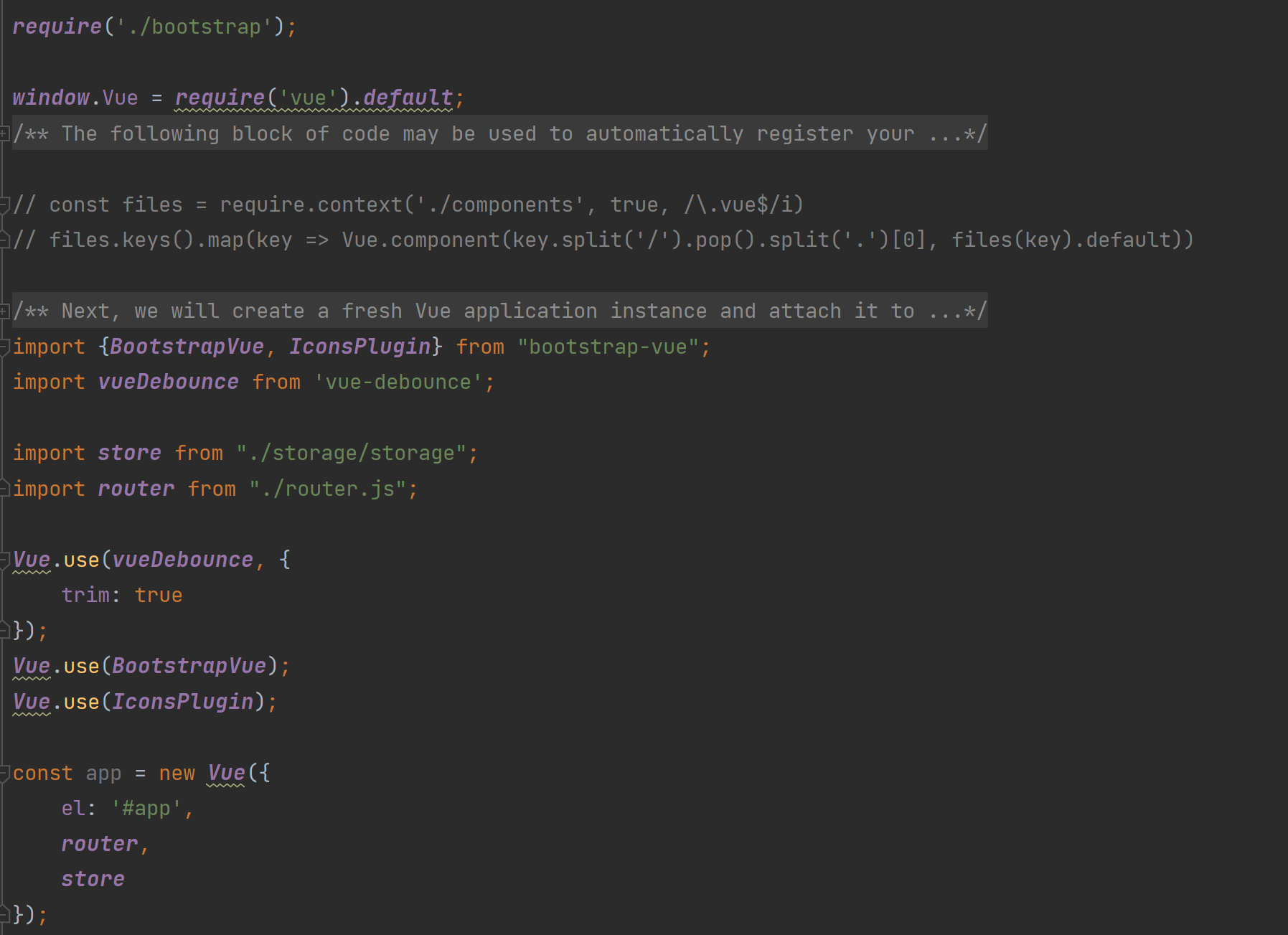
1. Команды для установки библиотеки стилей «Bootstrap»



1. Команда для установки библиотеки «jQuery»

После выполнения загрузки всего необходимого ПО было выполнено подключение в программном коде.

На рисунках ?, ? и ? изображено подключение всех библиотек для дальнейшей работы с ними.



1. Подключение фреймворка «Vue»



1. Подключение библиотек JS



1. Подключение стилей

Клиентская часть состоит из графического интерфейса, с которым можно взаимодействовать. Разработка клиентской части была выполнена в несколько этапов:

* Разметка HTML, вёрстка;
* Применение CSS стилей для улучшения графического интерфейса;
* Разработка взаимодействия пользователя и графического интерфейса;
* Обеспечение работы клиентской части с программным интерфейсом серверной части приложения.

Фреймворк Vue предоставляет компонентный подход к разработке клиентской части приложений. Каждый компонент состоит из HTML, CSS и JS. Данные компоненты возможно переиспользовать и внедрять в другие компоненты, благодаря этому разработка становится эффективней.

На рисунке ? изображен пример компонента.

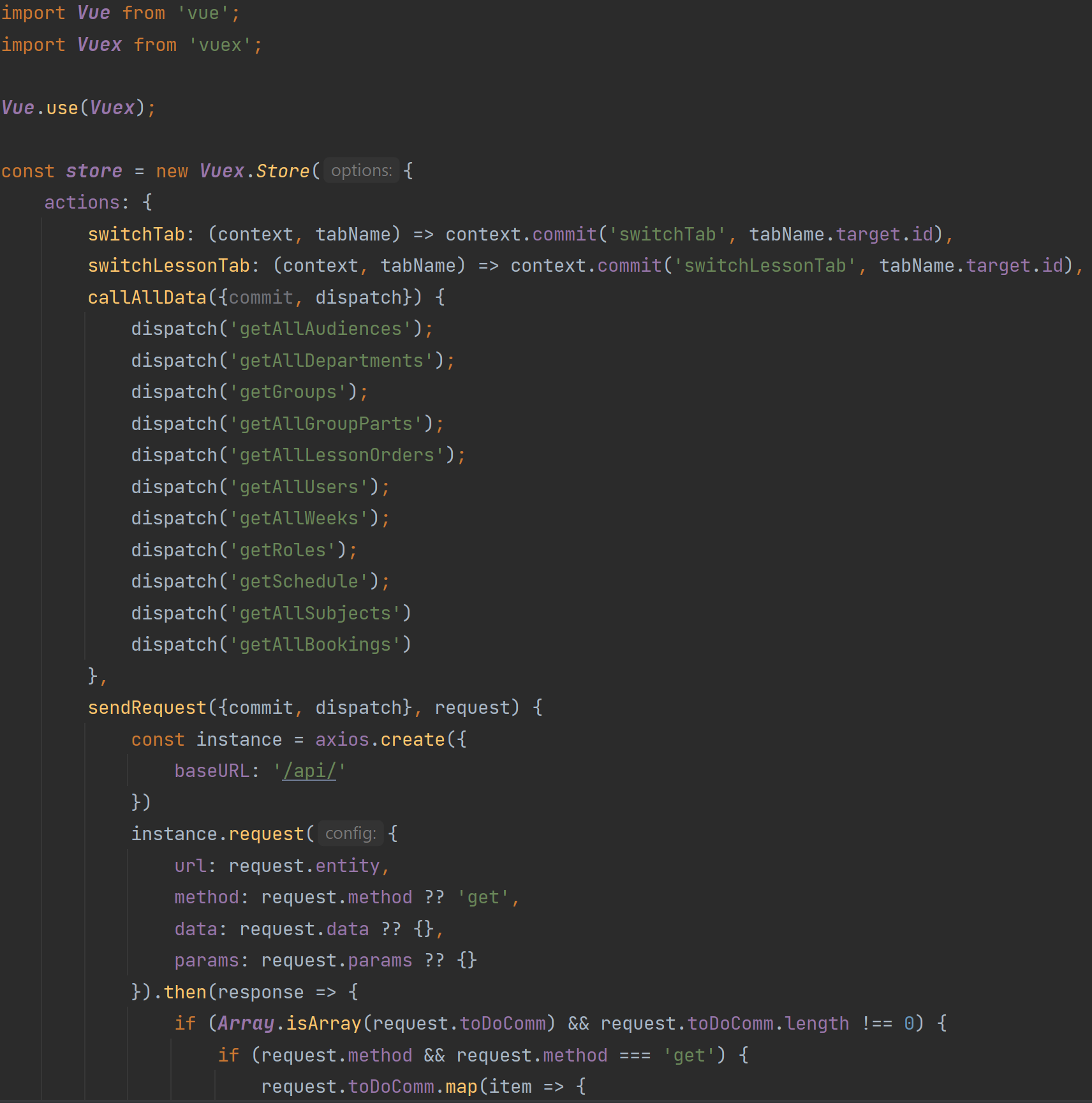


1. Компонент Vue

Данный компонент разделен на три составляющих, перечисленных выше, и находятся внутри тегов:

* «template» здесь помещается верстка, а также, кроме стандартных HTML тегов, возможно поместить другие компоненты в верстку, введя их название;
* «script» здесь помещается JS код, который обеспечивает взаимодействие пользователя с данным компонентом;
* «style» здесь помещаются стили.

Внутри компонентов выполняются первые три этапа разработки клиентской части приложения. Последний этап выполняет модуль фреймворка «Vue» под названием «Vuex». Данный модуль позволяет отделить данные в общий контейнер, который отвечает за оперирование данными и взаимодействием с программным интерфейсом серверной части приложения посредством сети «Internet». На рисунке ? изображен программный код модуля «Vuex».



1. Программный код модуля «Vuex»