

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 ТЕОРИТИЧЕСКАЯ ЧАСТЬ	7
1.1 Исследование предметной области.....	7
1.2 Анализ и выбор инструментальных средств.....	17
2 ПРОЕКТНАЯ ЧАСТЬ.....	23
2.1. Обоснование выбора средств разработки.....	23
2.2. Проектирование и разработка ПО.....	27
3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ	56
ЗАКЛЮЧЕНИЕ.....	61

					ФГБОУ ВО «УНИКИТ им. К.Г. Разумовского (ПКУ)» -09.02.05-090205-9о-17/1			
Изм	Лист	№ докум.	Подп.	Дата				
Разраб.		А.О. Овчинников			Разработка Learning Management System (на примере Университетского колледжа информационных технологий ФГБОУ ВО «Московский государственный университет технологий и управления им. К.Г. Разумовского (ПКУ)»)	Литера	Лист	Листов
Руководит		С.Ю. Кузьменко				КП	2	76
						090205-9о-17/1		

ВВЕДЕНИЕ

В современном мире, образование является важнейшим процессом в жизни любого человека. Начиная с самого детства, человек так или иначе сталкивается с образованием в своей жизни.

С каждым годом, средний мировой уровень образования бесперебойно растёт. Практически, все страны стремятся повысить уровень образования граждан для выращивания специалистов, способных развивать науку, культуру, экономику и общество в целом.

От качества образования зависит не только будущая профессиональная карьера студента, но и его социализация в обществе. Каждый человек, так или иначе, применяет навыки, приобретенные в учебных заведениях, в повседневной жизни, в том числе навыки коммуникации, получаемые при работе в коллективе.

Безусловное постоянное обеспечение актуальной информацией – сложнейшая проблема в сфере образования, вызываемая техническими ограничениями из-за использования старых методов передачи информации.

Во многих современных образовательных учреждениях информация хранится, записывается и передаётся в виде документов, возможно в их онлайн версиях, без возможности комплексно распространять информацию для студентов.

В частности, в ФГБОУ ВО «МГУТУ им. К.Г. Разумовского (ПКУ)», несмотря на собственный веб-сайт, распространяет информацию для студентов, в основном, с помощью документов и личных сообщений, что оставляет общее информирование на низком уровне.

Одним из лучших решений для данной проблемы и способов улучшения самого процесса образования, применяемых во множестве передовых образовательных учреждениях, являются системы управления обучением (Learning Management Systems), позволяющие студентам напрямую

взаимодействовать с их образовательными учреждениями в любое время и для любых целей без прямого участия преподавателей.

Подобные системы снижают административную нагрузку на состав образовательного учреждения, а также позволяет студентам в любое время получать актуальную информацию об их обучении, находить дополнительные источники знаний для повышения профессиональных навыков, а также оперативно решать некоторые вопросы с администрацией.

Системы управления обучением в корне меняют процесс образования, автоматизируя его, делая более простым как для студента, так и для преподавателей. Системы постоянно обрастают новым функционалом и постепенно расширяют своё влияние на процесс обучения, органично дополняя, позволяя установить лучший контакт между сторонами обучения.

Для систем управления обучением не существует особых правил и границ. Системы могут иметь вид как небольших приложений, например, информирующих об актуальном расписании, так и огромных систем, с которыми студенты обучаются на уровне обычных занятий с настоящими преподавателями, собирающих и анализирующих статистику.

На данный момент в России не существует единых систем управления обучением для высших или средне-специальных учебных заведений и это является одним из затормаживающих факторов, не позволяющим обеспечить высокий уровень образования.

Данная тема актуальна, так как проблема качества образования в России вызвана в том числе недостаточным уровнем технической поддержки и в совокупности с другими факторами вызывает негативное влияние на уровень образования в целом.

Объектом исследования является система управления обучением для образовательной организации.

Предметом исследования в проекте является разработка прототипа системы управления обучением ФГБОУ ВО «МГУТУ им. К.Г. Разумовского (ПКУ)».

Результат реализации проекта должен обеспечить возможность студенту пользоваться актуальной информацией о процессе обучения, а также предоставлять автоматизированный доступ к некоторым административным процессам связанных с обучением.

Целью исследования является проектирование и разработка прототипа системы управления обучением.

В соответствии с поставленной целью были определены следующие задачи исследования:

- изучение теоретической и методической информации о цели проекта,
- изучение аналогичного проекта, применяемого в реальных условиях,
- анализ и обработка данных, полученных при изучении,
- анализ и выбор инструментов разработки,
- разработка прототипа системы управления обучением,
- анализ результатов проделанных работ.

В проекте применялись следующие методы исследования:

- анализ,
- абстракция,
- конкретизация.

Теоретическая значимость проекта заключается в успешно разработанном прототипе системы управления обучением, который может применяться и расширяться образовательной организацией с целью повышения качества образования и развития уровня информирования внутри образовательного учреждения.

Практическая значимость проекта заключается в комплексном предоставлении информации о процессе обучения для студентов, снижении нагрузки на преподавательский состав.

Структура дипломного проекта обусловлена предметом, целью и задачами исследования. Проект состоит из введения, теоретической,

проектной и экономической частей, а также заключения и списка используемых источников.

В теоретической части рассмотрен аналогичный введенный в работу проект системы управления обучением, определен минимальный функционал, исследована предметная область и выбраны инструменты разработки.

В проектной части проведено обоснование выбранных средств разработки, описаны проектирование и разработка системы управления обучением.

В экономической части проведен расчёт стоимости проекта и спрогнозирована экономическая польза от создания и реализации проекта.

1 ТЕОРИТИЧЕСКАЯ ЧАСТЬ

1.1 Исследование предметной области

1.1.1 Анализ предметной области

Learning management system (система управления обучением) – приложения для администрирования учебных процессов в рамках взаимодействия между учебной организацией и обучающимися на расстоянии, также необходимые для планирования и управления учебными мероприятиями.

Для успешной разработки системы управления обучением необходимо определить её преимущества и недостатки на концептуальном уровне.

Преимущества:

- свобода доступа к информации,
- потенциальная гибкость обучения,
- оптимизация административных процессов,
- потенциальная возможность применения интерактивных инструментов

обучения,

- возможность анализировать обучение,
- снижение нагрузки на преподавательский состав.

Недостатки:

- внедрение требует хорошей технологической инфраструктуры,
- необходимость во внедрении и обслуживании системы.

Для разработки качественного проекта необходимо выделить типы систем управления обучением, для дальнейшего определения вида разрабатываемого прототипа.

Типы систем управления обучением:

- академические,
- корпоративные,
- отраслевые,

- площадки продаж,
- тренинговые площадки.

Разрабатываемый продукт соответствует академической системе управления обучением, так как предназначается для колледжа.

Существует всего 2 вида систем управления обучением:

- облачные,
- серверные.

Исходя из данных электронного ресурса vs.ru, можно сказать что серверная система управления обучением работает с пользователями через браузер, как обычный сайт, а облачные являются лишь арендуемой ресурсной базой для обучения [1].

После проведённого анализа было определено, что разрабатываемым проектом является серверная система управления обучением академического типа.

1.1.2 Анализ проекта LMS HSE

Одним из самых популярных примеров удачных систем управления обучением можно выделить проект Высшей школы экономики (рисунок 1) – LMS HSE, которая стабильно работает и развивается с 2014 года [2].

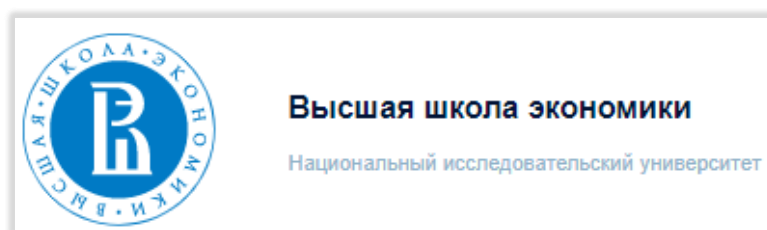


Рисунок 1 – Логотип Высшей школы экономики

Данный проект подходит для изучения и выделения сильных и слабых сторон.

На страницу системы управления обучением (рисунок 2) можно перейти по ссылке <https://lms.hse.ru> или с главной страницы сайта Высшей школы экономики.

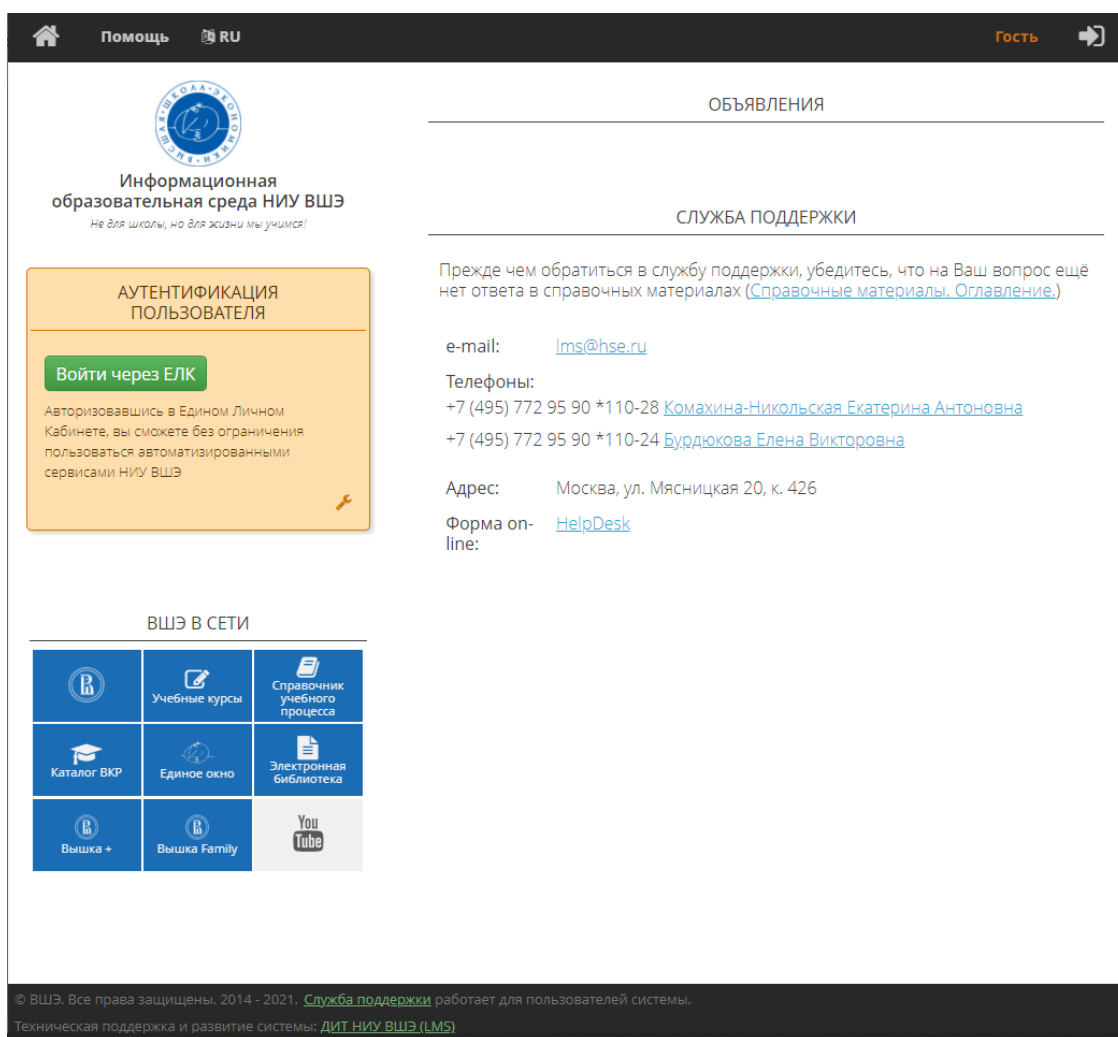


Рисунок 2 – Главная страница LMS HSE

Главная страница системы управления обучением Высшей школы экономики содержит ссылку на авторизацию, контактную информацию и ссылки на другие информационные ресурсы образовательной организации (рисунок 3).

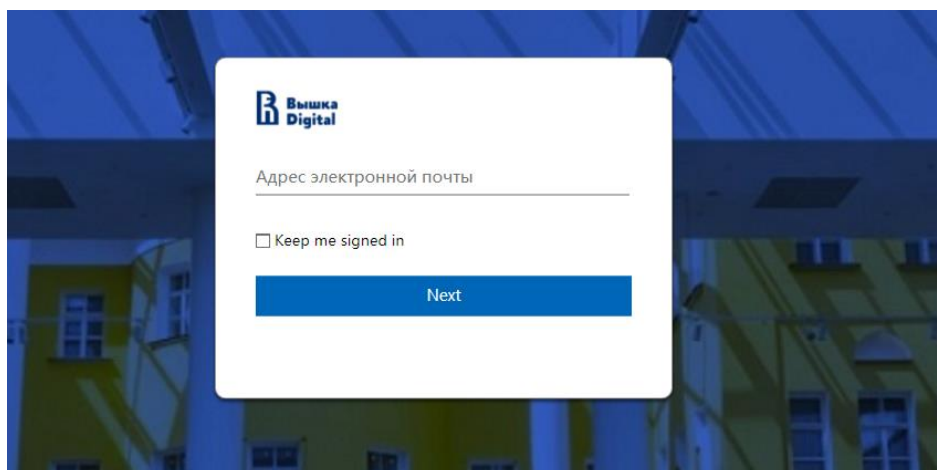


Рисунок 3 – Форма авторизации LMS HSE

После авторизации в системе управления обучением, в меню открываются дополнительные вкладки и появляются новые объявления, а также размещаются дополнительные вкладки «инструменты» и «дисциплины» в шапке сайта.

Самые важные элементы системы выделены отдельными цветами в меню и находятся в начале плитки.

Вид страницы для авторизованного пользователя можно увидеть на рисунке (4).

- практические занятия,
- праздничные события.

Каждая активность содержит информацию о типе занятия:

- способе проведения,
- времени,
- дате и дне недели,
- тэгах проводимого занятия,
- закрепленного преподавателя,
- информацию о способе участия в активности.

Пример активности расписания представлен на рисунке (5).

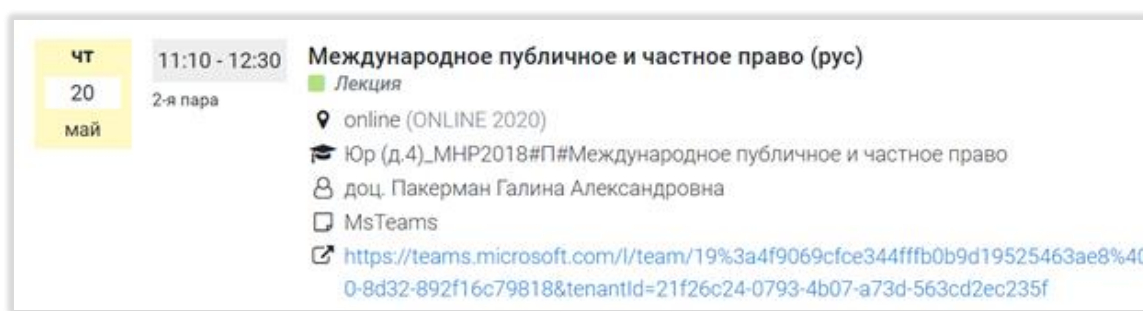


Рисунок 5 – Пример лекции в расписании LMS HSE

Страница расписания имеет отличающийся от основной страницы дизайн и расположена на отдельной странице, т.е. каждый раз для открытия актуального расписания, необходимо переходить на отдельную страницу.

В таблице расписания реализована возможность загрузки, фильтрации и сортировки расписания (рисунок 6).

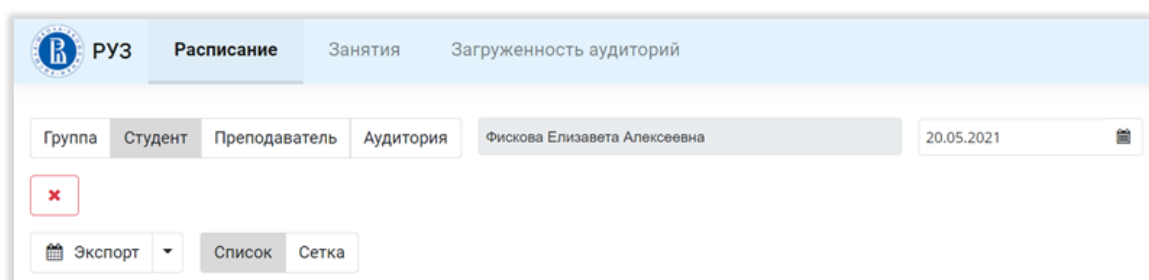


Рисунок 6 – Фильтры и сортировка расписания

В целом, страница расписания вызывает неудобства, длинный список с подробной информацией сложен для восприятия (рисунок 7).

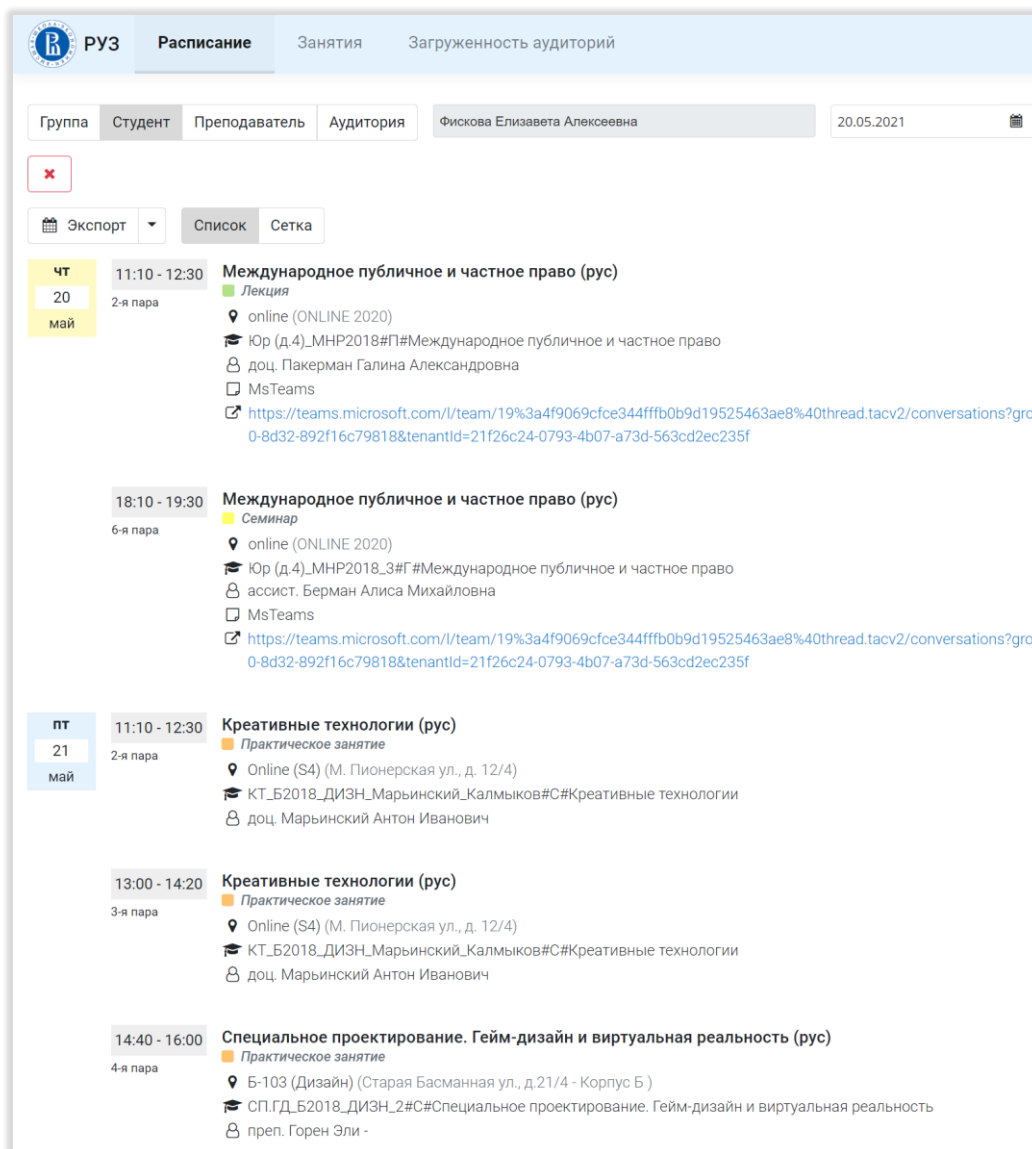


Рисунок 7 – Расписание занятий LMS HSE

Студент на любом этапе обучения может посмотреть все дисциплины, которые он должен освоить за время обучения (рисунок 8).

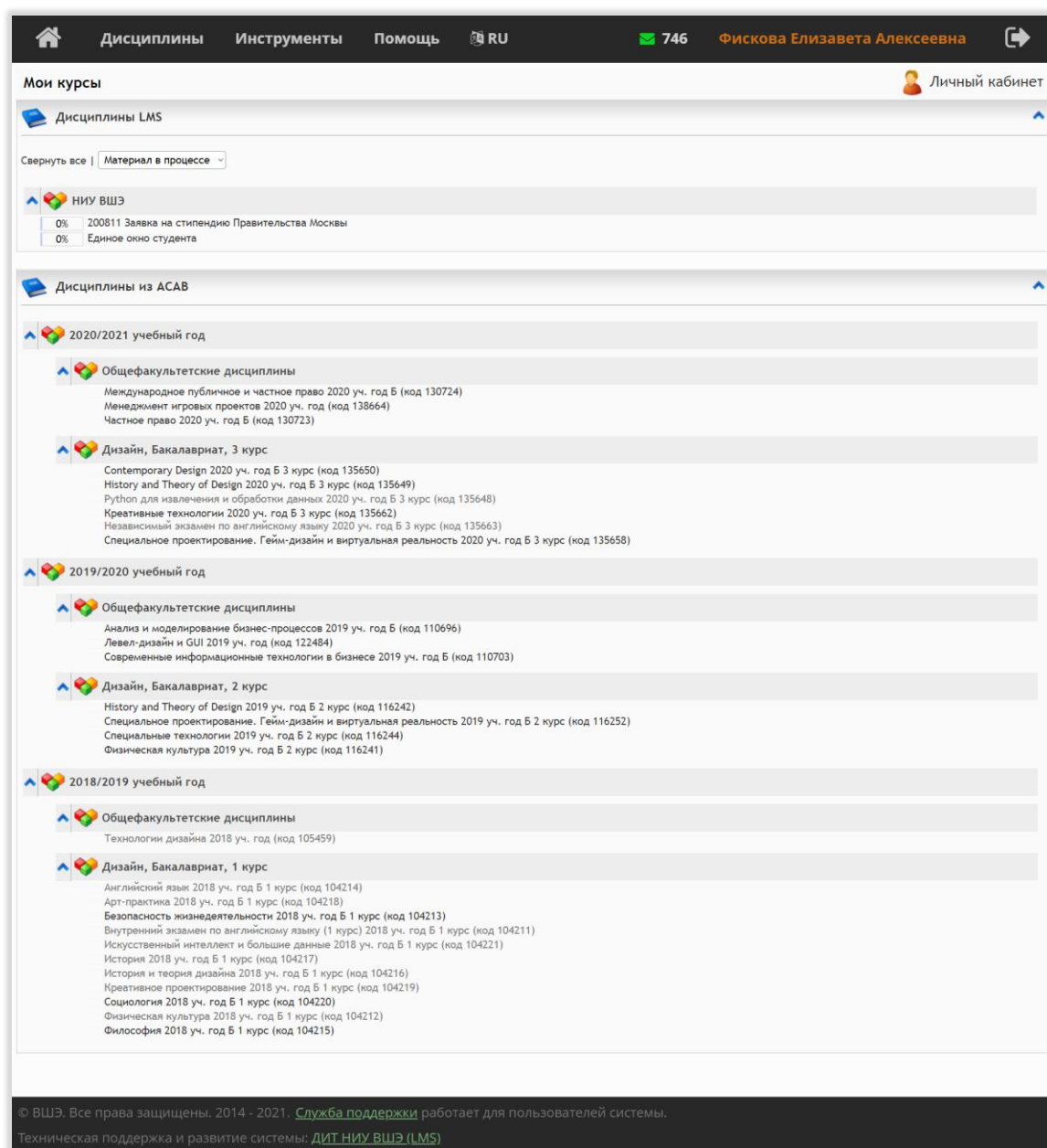


Рисунок 8 - Страница «Мои курсы» LMS HSE

В HSE, в рамках образовательного процесса предусмотрена проектная деятельность, в LMS HSE имеется список всех курсовых работ, с приложенными документами и файлами, для дальнейшего обращения к ним.

Список выполненных проектов помогает студентам обращаться к опыту своих прошлых работ, не заботясь об их хранении на личных носителях информации.

Данное решение находится на стадии внедрения и на рисунке (9) отображается его текущее состояние.

Календарный период	Тип работы	Тема	Тема на английском	Загружено	Презентация	Приложение	Процент заимствований	Статус проверки	Оценка	Отзыв	Рецензия
2020/2021 учебный год	Проект	3 курс		нет			-	-	-		
2019/2020 учебный год	Проект	Проект 2 курс		нет			-	-	9		

Рисунок 9 – Страница проектов студента

Помимо просмотра собственных проектов, у пользователя есть возможность просматривать работы других студентов (рисунок 10).

Выпускные квалификационные работы студентов НИУ ВШЭ

Всего по вашему запросу работ **55610**

Образование как инструмент мягкой силы России в Индонезии
 Санкт-Петербургская школа социальных наук и востоковедения
 Программа: Сравнительная политика Евразии (Магистратура)
 ФИО студента: - Пуutri Квартa Каприарти Суманто -
 Руководитель: Агеева Вера Дмитриевна
 Год защиты: 2021

Повышение логистического сервиса в электронной торговле в Индонезии за счет доставки через постаматы (на примере компании LAZADA)
 Высшая школа бизнеса
 Программа: Цифровая логистика и управление цепями поставок (Магистратура)
 ФИО студента: - Аршад Хартати Хи -
 Руководитель: Морозова Юлия Александровна
 Год защиты: 2021

Управление CEO эффективностью компаний малого и среднего бизнеса на основе денежных потоков
 Факультет экономических наук
 Программа: Стратегическое управление финансами фирмы (Магистратура)
 ФИО студента: - Ву Тхи Фьонг -
 Руководитель: Макеева Елена Юрьевна
 Год защиты: 2021

Внедрение технологий в сельском хозяйстве Эритреи: социально-экономическая перспектива

Фильтры:
 Кампус/факультет: Все
 ФИО студента:
 Руководитель:
 Год защиты: Все
 Название работы:
 Уровень образования: ☐ бакалавриат ☐ магистратура ☒ Не важно
 Оценка: Не имеет значения
 Программа: Все
 Доступен полный текст: Не имеет значения
 Язык: ☒ Все ☐ Русский ☐ Английский
 Поиск

Рисунок 10 – Работы студентов НИУ ВШЭ

Данное решение позволит набраться опыта и идей в проектах других студентов.

Все проекты добавляются в базу проверки антиплагиата, что исключает полное копирование работ.

Одной из самых сложных и удобных функций системы является система обмена сообщениями (рисунок 11), позволяющая рассылать оповещения студентам о мероприятиях, оценке их работ, проведении контрольных, а также домашние задания и рекомендации самостоятельному к обучению.

У студентов имеется возможность обмениваться сообщениями между собой, в том числе для проведения групповых собраний во время разработки проектов.

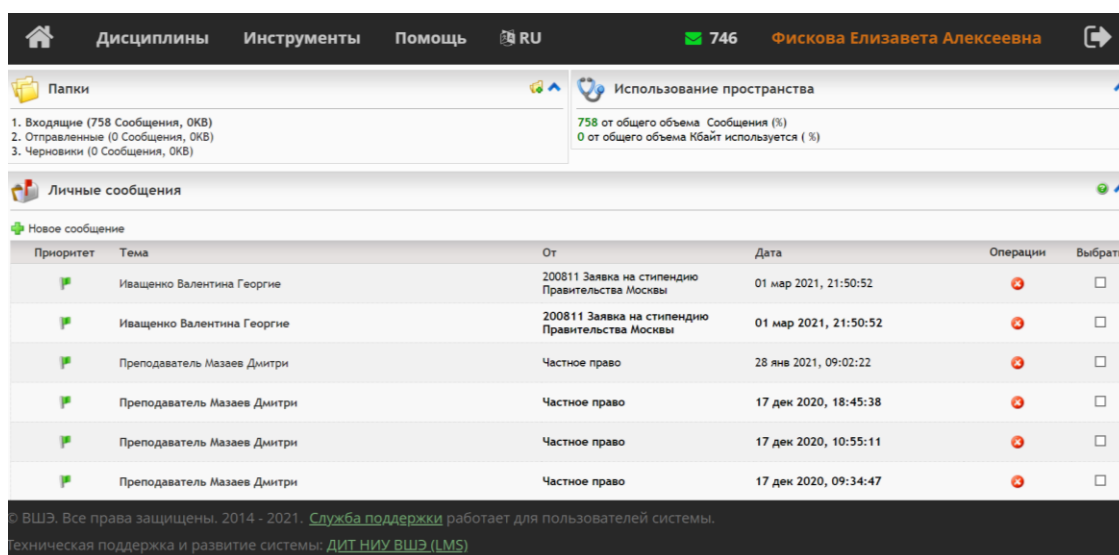


Рисунок 11 – Страница уведомлений LMS HSE

На момент анализа, система была протестирована на разных устройствах для исключения ошибок отображения и более объемного анализа.

Система управления обучением Высшей школы экономики имеет преимущества и недостатки.

Преимущества:

- всесторонняя поддержка процесса обучения,
- информативное простое для восприятия главное меню,
- возможность расширения функционала,
- возможность обращения к администрации внутри системы.

Недостатки:

- разная стилистика на страницах, нет единого дизайна для системы,
- сложное для восприятия и громоздкое расписание занятий.

1.1.3 Постановка задач для разработки

После проведения анализа предметной области и существующего проекта LMS, возникла необходимость определить функционал разрабатываемого проекта для тестирования и дальнейшего расширения.

Так как система должна быть максимально доступна и проста [3], было принято решение по разработке системы в виде сайта с доступом через авторизацию.

Для извлечения пользы, система управления обучением должна интегрироваться и подстраиваться под нужды организации.

В условиях отсутствия статистических данных выделен минимальный функционал:

- система авторизации,
- личная информация студента на странице,
- форма онлайн-заказа справки с места обучения,
- баннер с уведомлением для группы,
-
- страница поддержки.

1.2 Анализ и выбор инструментальных средств

1.2.1 HTML

HTML [4] – язык разметки страницы, который используют большинство веб-страниц в интернете, данный язык необходим для разработки и должен применяться для отображения контента.

Пример HTML кода представлен на рисунке (рисунок 12).

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <header>
    <a href="/main">главная</a>
  </header>
  <main>
    <h1>страница не найдена!</h1>
  </main>
  <footer>
    <a href="/vk.com/company">vk</a>
  </footer>
</body>
</html>

```

Рисунок 12 – HTML код страницы

В HTML5 реализовано множество новых синтаксических особенностей, таких как элементы `<video>` и `<audio>`, а также возможность использования SVG и математических формул [5]. Данные нововведения разработаны для упрощения создания и управления графическими и мультимедийными объектами в сети без необходимости использования плагинов.

1.2.2 CSS

CSS – язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам (например, HTML страницам) [6].

Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML [7].

Язык CSS стал стандартом в веб-разработке, потому что позволяет быстро изменить визуальное оформление сайта, не прибегая к использованию более сложных языков программирования.

Пример CSS кода представлен на рисунке (13).

```
* {
    margin: 0;
    padding: 0;
    border: none;
}

body {
    display: flex;
    flex-direction: column;
    justify-content: flex-start;
    align-items: center;
    background: #fff;
    min-height: 100vh;
}

header {
    margin-bottom: 10px;
    width: 100%;
    height: 64px;
    background: #fff;
    box-shadow: 0 0 3px -2px #000;
}
```

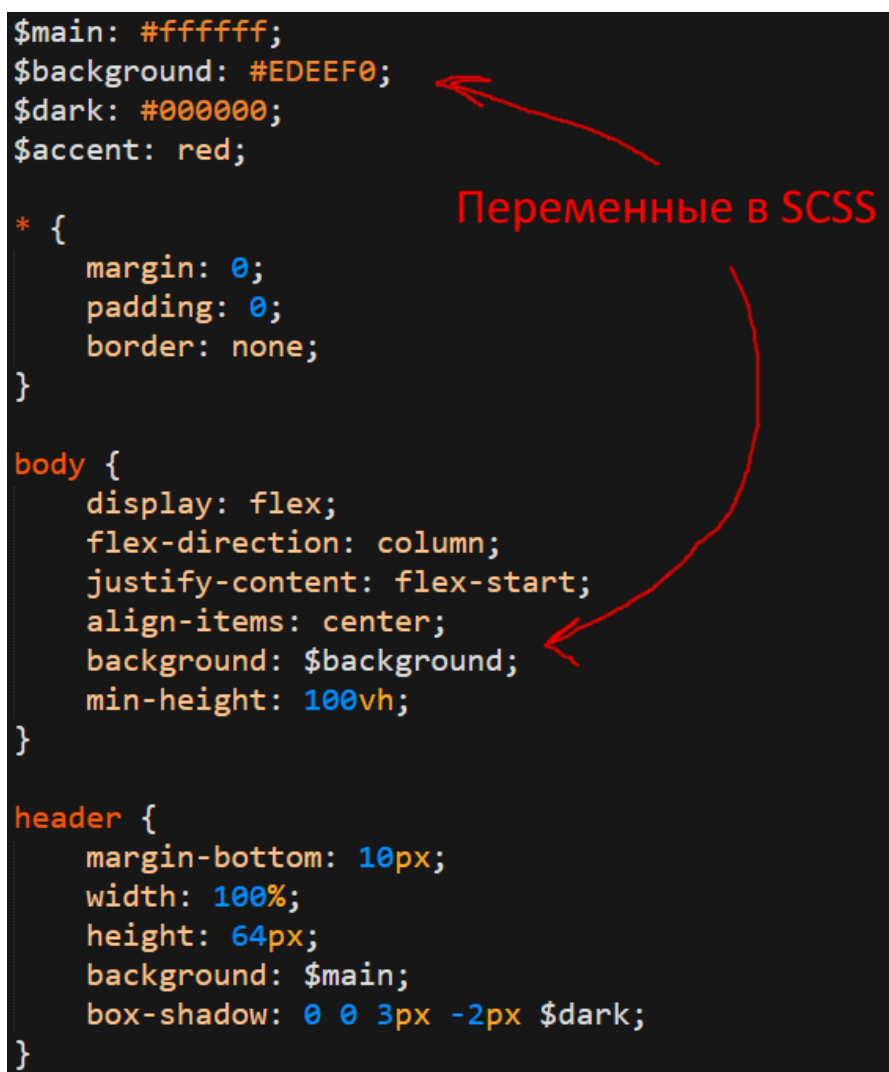
Рисунок 13 – Содержание css файла

В коммерческой разработке редко используется чистый CSS. В большинстве случаев основная масса таблиц стилей разрабатывается на препроцессорах SAAS, в частности на языке SCSS.

Для ускорения процесса существует библиотека Bootstrap, позволяющая решить проблемы с адаптацией контента под мобильные устройства.

SCSS частично предоставляет некоторые возможности языков программирования, таких как переменные или циклы. Также, синтаксис языка позволяет лучше воспринимать код, что сильно упрощает процесс поиска ошибок или изменения существующего кода [8].

Пример SCSS представлен на рисунке (14).



```
$main: #ffffff;
$background: #EDEEF0;
$dark: #000000;
$accent: red;

* {
  margin: 0;
  padding: 0;
  border: none;
}

body {
  display: flex;
  flex-direction: column;
  justify-content: flex-start;
  align-items: center;
  background: $background;
  min-height: 100vh;
}

header {
  margin-bottom: 10px;
  width: 100%;
  height: 64px;
  background: $main;
  box-shadow: 0 0 3px -2px $dark;
}
```

Перменные в SCSS

Рисунок 14 – Содержание SCSS файла

1.2.3 PHP

PHP (рекурсивный акроним словосочетания PHP: Hypertext Preprocessor) - это распространенный язык программирования общего назначения. PHP сконструирован специально для web-разработки и в его код может внедряться HTML [9].

Разработка на PHP в основном ведётся с помощью различных фреймворков. Самым популярным на данный момент фреймворком является Laravel.

Пример синтаксиса PHP представлен на рисунке (15).

```
//реализация синглтона
public static function instance() {
    if(self::$instance === null) {
        self::$instance = new self;
    }

    return self::$instance;
}

//проверка на выполнение запроса
public function execute($query) {
    $stmt = $this->pdo->prepare($query);
    return $stmt->execute();
}

//запрос с выводом данных
public function query($query) {
    $stmt = $this->pdo->prepare($query);
    $res = $stmt->execute();
    if($res !== false) {
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }

    return [];
}
```

Рисунок 15 – Синтаксис PHP

1.2.3 SQL

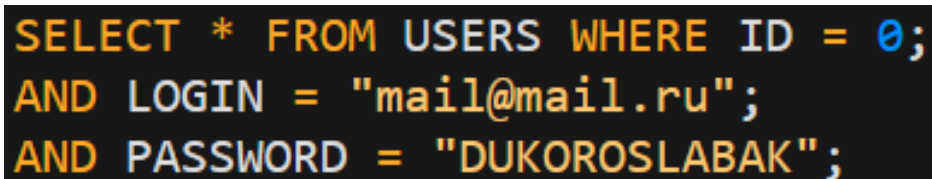
SQL – декларативный язык программирования, применяемый для создания и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных [10].

Язык SQL является языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. Данный язык не является тьюринг-полным.

В контексте баз данных, под данными понимают набор значений, который собирается в строки и столбцы, образуя таблицу.

Базы данных – это совокупность данных, представленных определенным образом, и набор инструментов для управления [11].

Пример синтаксиса SQL представлен на рисунке (16).



```
SELECT * FROM USERS WHERE ID = 0;  
AND LOGIN = "mail@mail.ru";  
AND PASSWORD = "DUKOROSLABAK";
```

Рисунок 16 – Синтаксис SQL

1.2.3 Выбор средств разработки

Для создания проекта были выбраны следующие инструменты и языки разработки [12]:

- PHP 8.0 как основной язык программирования,
- СУБД MariaDB 10.4,
- SCSS с применением препроцессора Koala [13],
- браузер Opera,
- текстовый редактор Sublime Text 3,
- локальный сервер Open Server.

2 ПРОЕКТНАЯ ЧАСТЬ

2.1. Обоснование выбора средств разработки

2.1.1 Список выбранных средств разработки

Для создания проекта были выбраны средства разработки:

- PHP 8.0 как основной язык программирования,
- СУБД MariaDB 10.4,
- SCSS с применением препроцессора Koala,
- браузер Opera,
- текстовый редактор Sublime Text 3,
- локальный сервер Open Server.

Выбор инструментов разработки производился на основе опыта разработчика, а также в ходе принятия решений на основе теоретического анализа.

2.1.2 PHP

Для разработки логики проекта был выбран язык PHP, так как он обладает высокой скоростью работы и общей производительностью ресурсов, а также в связи с выходом новой версии PHP 8.0.

Основным конкурентом языка в сфере веб-разработки является Python, в связи с чем был проведён анализ языков.

Язык PHP обладает некоторыми преимуществами над языком Python в сфере веб-разработки:

- эффективная работа на стороне сервера,
- PHP имеет большее количество фреймворков,
- огромная информационная база,
- широкая поддержка работы с базами данных.

Баннер на главном сайте `php.net` с кратким описанием обновления представлен на рисунке (17).

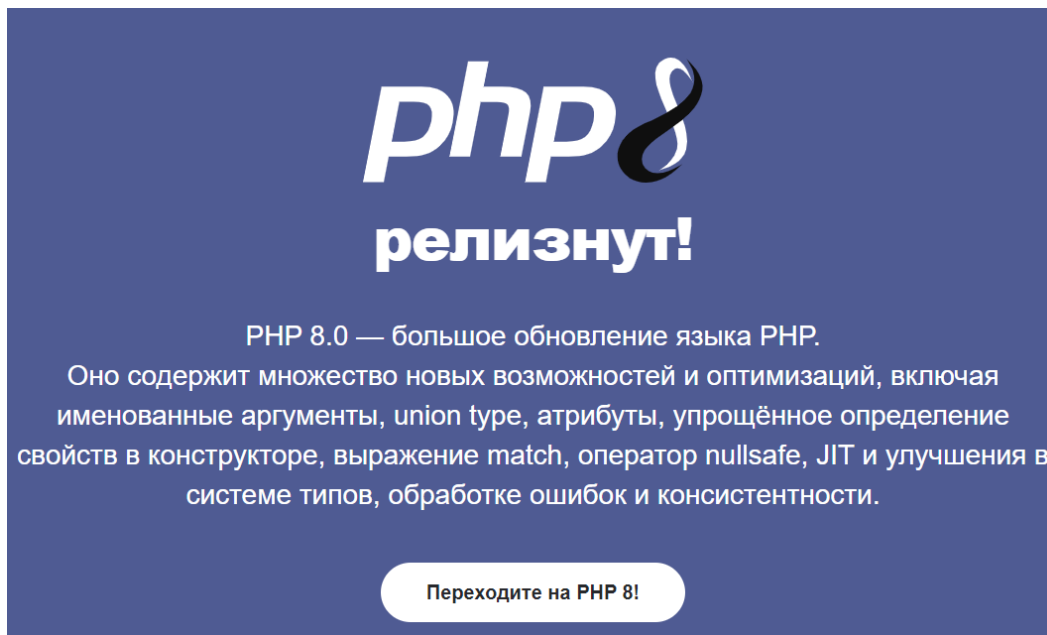


Рисунок 17 – Баннер обновления PHP

Одними из решающих факторов в выборе языка являлись опыт разработчика в сфере разработки на PHP, а также некоторые наработки с прошлых проектов.

В проекте, для разработки на PHP применяются рекомендации написания кода PSR:

- PSR-1 – стандарт стиля кодирования,
- PSR-4 – стандарт автозагрузки.

2.1.3 СУБД

Для разработки была выбрана СУБД MariaDB, так как она является первой СУБД, совместимой с версией языка PHP 8.0 и альтернатив на момент разработки не существовало.

Альтернативой MariaDB является одна из лучших СУБД – PostgreSQL, обладающая преимуществами в производительности и функционале над MariaDB.

В дальнейшем поддержании программы возможен полный переход на PostgreSQL.

При успешной интеграции системы допустима возможность расширения поддержки работы с базами данных при помощи резидентной системы управления базами данных (Redis).

2.1.4 Препроцессор CSS

Препроцессором для языка SCSS была выбрана программа Koala. Данная программа имеет ряд преимуществ перед другими препроцессорами:

- бесплатная open source лицензия,
- возможность обработки множества файлов одновременно,
- возможность сжатия конечных файлов,
- возможность создания каталога проекта,
- автоматическая компиляция файлов,
- возможность сжатия скриптов,
- возможность разделения области видимости.

Интерфейс Koala в рабочей среде с отображением преимуществ представлен на рисунке (18).

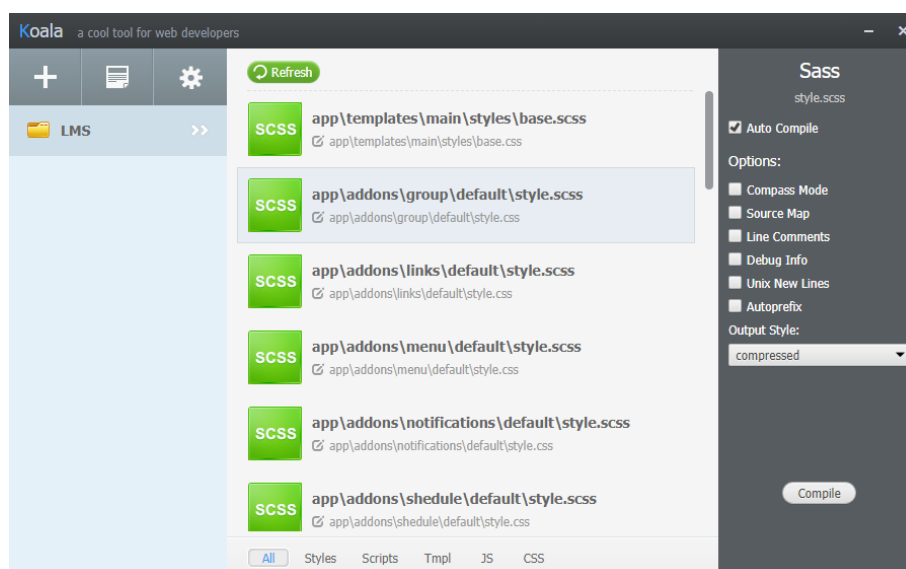


Рисунок 18 – Интерфейс Koala

Выбор препроцессора Koala был основан опытом разработчика в сфере фронтенд-разработки. Данный препроцессор показал себя одним из лучших со стороны удобства, надёжности, быстродействия и ресурсозатратности. Бесплатная лицензия позволила сократить средства на разработку.

2.1.5 Браузер

Для отладки выбран браузер Опера, так как на момент начала разработки вышло обновление с нововведениями, связанными с поддержкой отладки стилей и вёрстки.

Средства отладки Опера представлены на рисунке (19).



```
<html>
  <head>...</head>
  <body> flex
    <header>...</header>
    <main> grid
      <section class="shedule"> flex
        <table>...</table> flex
      </section>
      <section class="notifications">...</section> flex
      <section class="links">...</section> flex
      <section class="menu">
        ...
      </section> flex
      <section class="group">...</section> flex
    </main>
    <hr class="divider">
    <footer>
      ...
    </footer>
  </body>
</html>
```

Рисунок 19 – Консоль браузера Опера

2.1.6 Текстовый редактор

Основным текстовым редактором был выбран Sublime text 3 с плагинами SCSS и Emmet, ориентированными на быструю верстку, позволяя ускорить процесс разработки контента приложения, так как Sublime text 3 распространяется условно-бесплатно и имеет хорошую производительность.

2.1.4 Локальный сервер

Локальным сервером для разработки был выбран Open Server, так как на момент разработки он является самым надёжным, быстроедейственным и обладающим удобным интерфейсом.

Open Server, в отличие от Denwer, позволяет гибко настраивать использующие его языки, в том числе новую версию PHP 8.0, используя только интерфейс программы.

2.2. Проектирование и разработка ПО

2.1.1 Выбор архитектурного паттерна

Так как системы управления обучением являются долгосрочными проектами, предполагается их длительное поддержание и дальнейшее развитие. Для обеспечения качества разрабатываемого проекта, необходимо выбрать и использовать архитектурный паттерн системы, позволяющий реализовать весь потенциал с минимальным проявлением слабых сторон.

Самым популярным архитектурным паттерном в области разработки сайтов является MVC-паттерн (Model-View-Controller)[14]. MVC – простой способ построения структуры приложения, целью которого является отделение бизнес-логики от пользовательского интерфейса (рисунок 20).

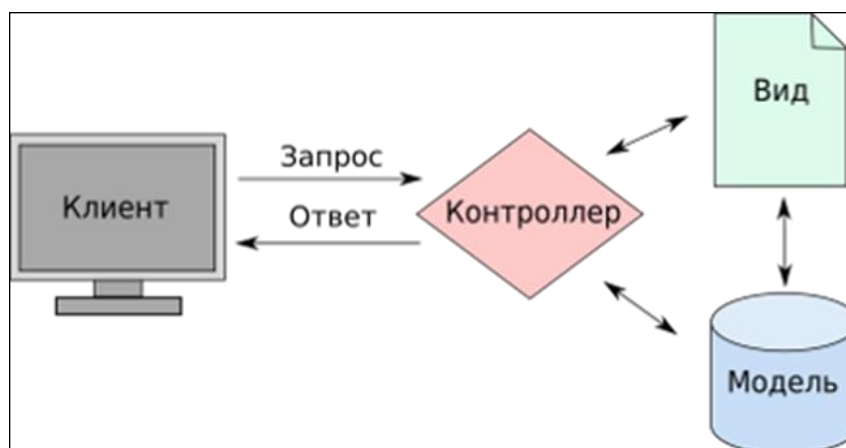


Рисунок 20 – Паттерн MVC

Многие системы и фреймворки основываются на базе компонентов MVC, в основном из-за простоты поддержания и развития проектов в долгосрочной перспективе.

Особенность паттерна возникает из-за разделения системы на составные части, которые слабо связаны между собой, что позволяет легко находить ошибки и разделять работу над проектом среди группы разработчиков.

Однако чистый концепт паттерна MVC имеет весомые недостатки:

- необходимость оперирования множеством моделей одновременно,
- отображение данных из множества моделей через отдельные виды,
- необходимость отображения шаблонов стилей через вид.

Проблема отображения стилей решается через шаблонизацию стилей. Шаблонизация является популярным способом реализации отображения одинаковых видов на странице.

Самым критическим недостатком паттерна являются неочевидные зависимости, появляющиеся при работе над несколькими моделями одновременно, когда возникает необходимость создавать контроллер, описывающий операции не только над моделью к которой он относится, но над моделями к которым отношения не имеет (рисунок 21).

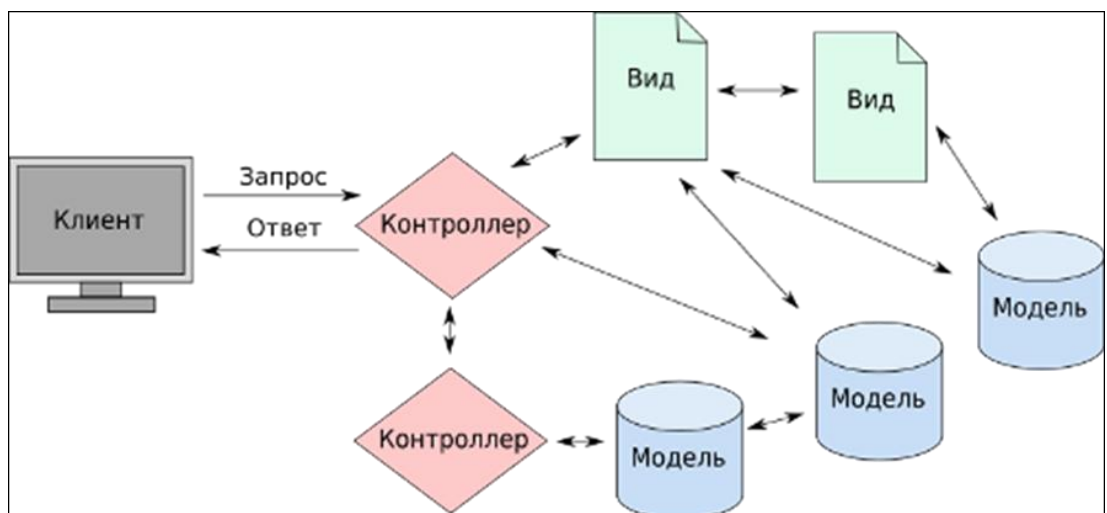


Рисунок 21 – Неочевидные зависимости MVC паттерна

Данные недостатки компенсирует архитектурный паттерн HMVC [15], который позволяет не смешивать множество моделей, видов и контроллеров в одном пространстве.

Паттерн HMVC представлен на рисунке (22).

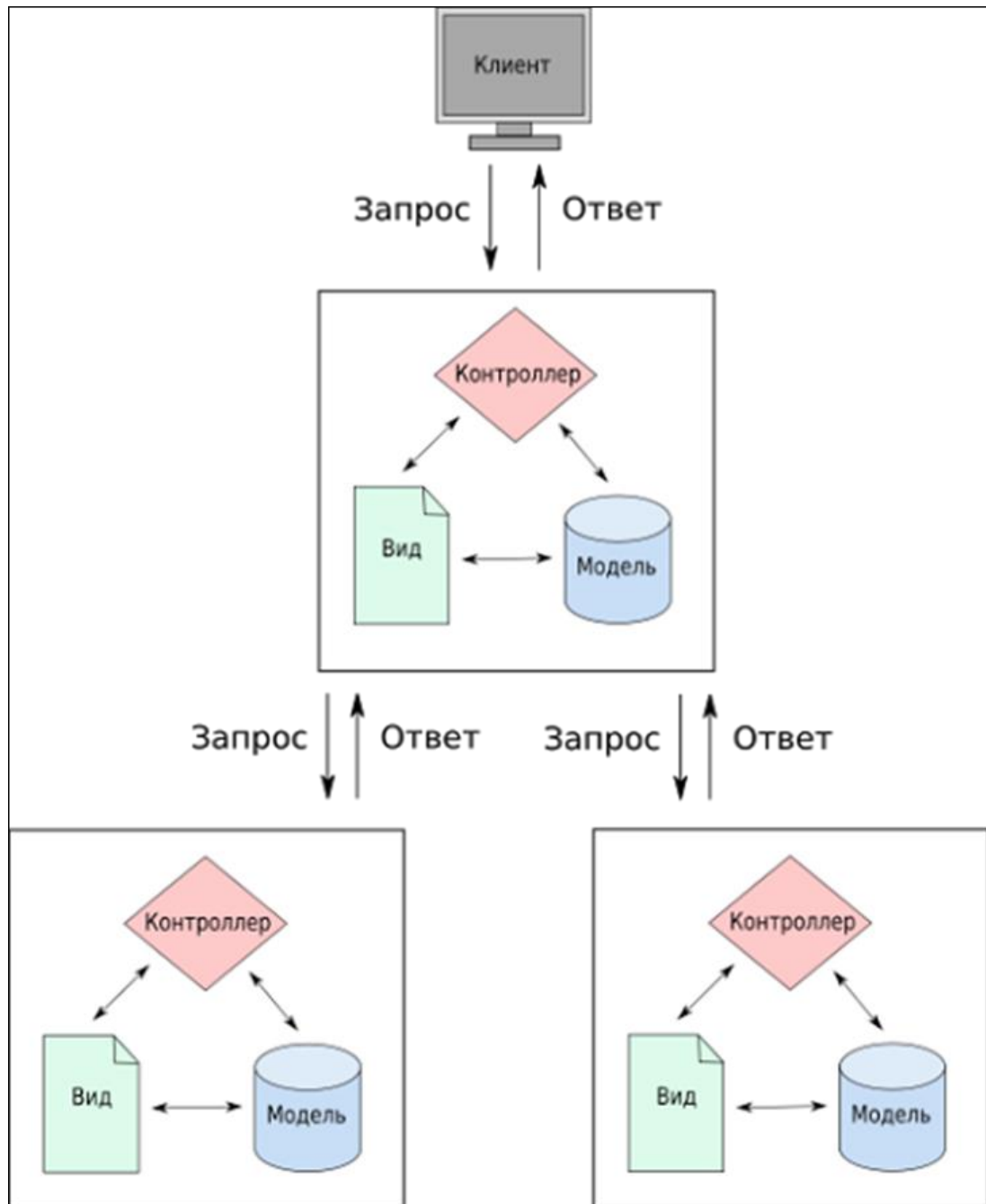


Рисунок 22 – Паттерн HMVC

Однако HMVC привязывает архитектуру к использованию только одного контроллера, модели и вида одновременно, что может сделать разрабатываемый проект менее гибким и расширяемым.

Решить проблему использования паттерна можно с помощью следующих модификаций MVC архитектуры:

- шаблонизация стилей,
- система компонентов, схожая с Битрикс-24.

Система компонентов обеспечивает разделение разрабатываемого функционала системы, при этом не обязательно привязывая логику, компонент может содержать только вёрстку и стиль, без связи с моделью.

После проведённого сравнения, MVC с доработками в сторону HMVC был принят как архитектурный паттерн разрабатываемой системы.

2.1.2 Принятие решений по разработке

После выбора архитектурного паттерна, следовало определить реализацию системы:

- организовать обращение к системе через паттерн фронт-контроллер (единая точка входа),
- разработать систему шаблонизации для повторного использования стилей,
- обеспечить безопасность и создать ЧПУ с помощью системы маршрутизации,
- реализовать загрузку страницы с автоматическим смещением стилей и JS файлов в тэги шаблона.

Для повышения общего качества архитектуры и возможности дальнейшей оптимизации и расширения выделены дополнительные задачи:

- для хранения настроек системы создать и обрабатывать файл конфигурации,
- реализовать абстракцию над паттернами регулярных выражений, применяемых в системе маршрутизации,
- подключение к базе данных организовать через PDO,

- методы ядра, использующиеся для разработки контента системы, должны быть обернуты в глобальные функции с упрощённым названием,
- во время разработки, придерживаться PSR-1, PSR-4 с упрощёнными пространствами имён и частично PSR-2,
- выделить константы в отдельный файл.

2.1.1 Система контроля версий

После определения основных критериев системы, необходимо приступить к разработке приложения. Для упрощения разработки и обеспечения безопасности хранения файлов проекта, было принято решение хранить приложение в системе контроля версий GIT на платформе GitHub [16].

Для проекта был создан репозиторий с кратким названием системы (рисунок 23).



Рисунок 23 – Полное название репозитория в системе GitHub

Для долгосрочного поддержания проекта необходимо произвести настройку репозитория. Для разработки создана ветка dev. На ветку main слиются все наработки, прошедшие тестирование. В дальнейшей разработке необходимо создавать новые ветки, наследованные от dev, применяемые в разработке отдельных компонентов системы.

2.1.1 Проектирование системы

Любая разработка программы начинается с проектирования. Для разработки качественного ПО необходимо составить схему работы ядра приложения. Диаграмма основных классов ядра приложения представлена на рисунке (24).

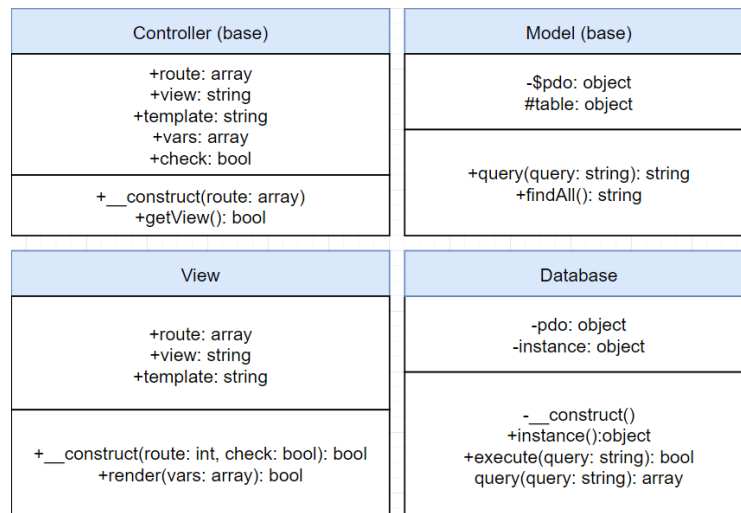


Рисунок 24 – Лиаграмма классов ядра

Принцип работы MVC спроектирован и представлен на рисунке (25).

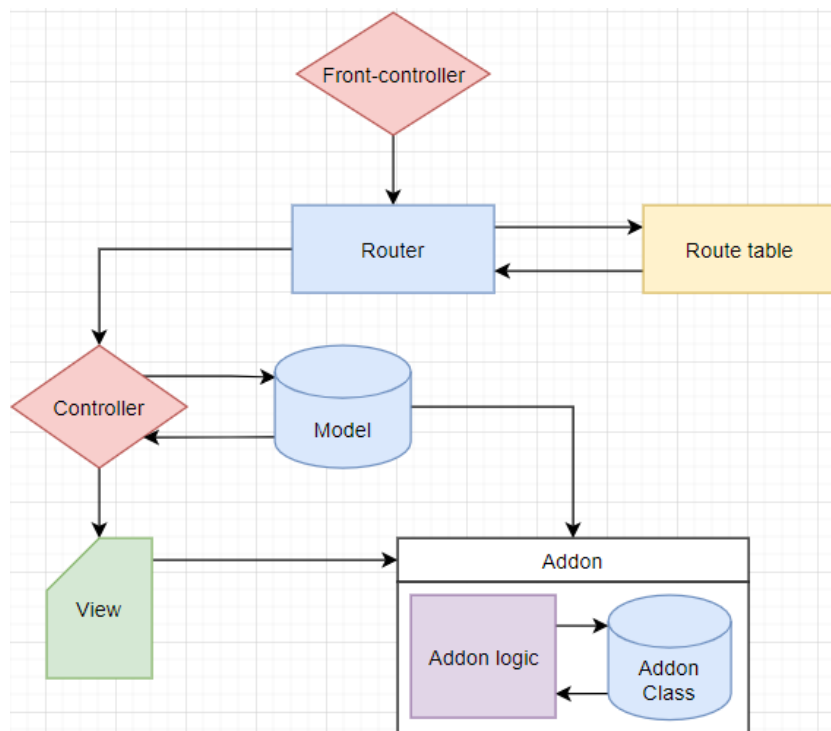


Рисунок 25 – Схема работы ядра системы

2.1.1 Структура файлов

Для лучшего обращения к файлам системы, применено разделение пространства ядра от пространства приложения, таким образом обеспечивая лучшее восприятие структуры файлов (рисунок 26).







	app	- каталог приложения
	core	- каталог ядра
	resources	- каталог прочих файлов
	.htaccess	- файл конфигурации Apache
	config	- конфиг
	index	- фронт-контроллер

Рисунок 26 – Структура корневого каталога

Каталог app предназначен для разработки контента и содержит в себе все MVC компоненты, аддоны, а также шаблоны и таблицу маршрутов для рутинга (рисунок 27).







	addons	- аддоны системы
	controllers	- контроллеры (MVC)
	models	- модели (MVC)
	templates	- шаблоны страниц
	views	- виды (MVC)
	routes	- файл маршрутов

Рисунок 27 – Каталог приложения

В приложении используется автозагрузка классов стандарта PSR-4 и в каталоги controllers, models и addons были добавлены собственные пространства имён /controller/, /model/ и /addon/ соответственно для упрощения написания кода с участием классов в этих каталогах.

Каталог views содержит конкретное содержание страниц и не имеет размещенных внутри классов, а лишь вызывает методы модели, ядра или аддонов.

Каталог core (рисунок 28) предназначен для основной логики приложения, не ориентированной на конкретные задачи бизнес-логики, а применяемой в системе в целом.

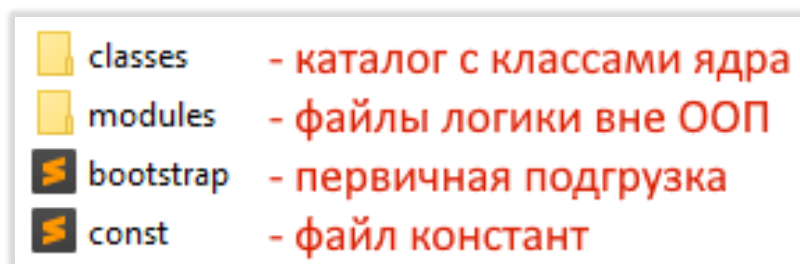


Рисунок 28 – Каталог ядра

Каталог `classes` подключен к системе автозагрузки классов стандарта PSR-4, имеет сокращенное пространство имен `/core/` и содержит все классы ядра, применяемые в приложении.

Каталог `modules` содержит остальную логику, сокращенные статические методы и не используется в автозагрузке классов, поэтому любое размещение классов рекомендуется производить в каталоге `classes`.

Папка `resources` содержит различные файлы, которые хранятся на сервере, наименование каталогов произвольное. Первичная структура каталога показана на рисунке (29).

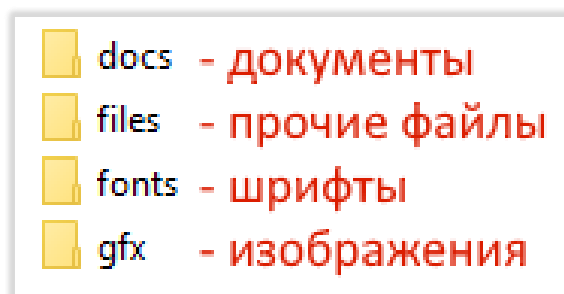


Рисунок 29 – Каталог `resources`

2.1.2 Единая точка входа

Паттерн единой точки входа – фронт-контроллера, широко применяется в приложениях самых разных видов сложности и назначений. Его концепция состоит в единой точке входа, где пользователь при переходе на любую ссылку попадает на главную страницу, которая, в зависимости от содержания ссылки, подключает файлы, формирующие запрошенную пользователем страницу.

Фронт-контроллер тесно взаимосвязан с системой маршрутизации и в данном проекте применяется вместе [17].

Данный паттерн организуется двумя файлами в корневом каталоге, которые предоставлены на рисунке (30).

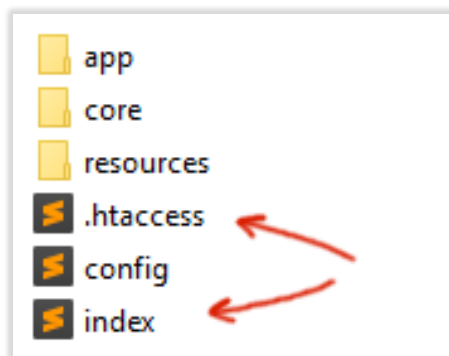


Рисунок 30 – Файлы фронт-контроллера

Файл .htaccess содержит перенаправление на файл index.php при попытке открытия любой ссылки с сохранением значения ссылки в глобальной переменной.

Реализация файла .htaccess представлена на рисунке (31)

```
RewriteEngine On

RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d

RewriteRule (.*?) index.php?$1 [L,QSA]
```

Рисунок 31 – Содержание .htaccess

Файл index.php подключает загрузочный файл и перенаправляет пользователя по запрошенному маршруту (рисунок 32).

```
<?php
require('core/bootstrap.php');

core\Router::dispatch($query);
```

Рисунок 32 – Содержание index.php

Для перенаправления пользователя используется система маршрутизации, которая вызывается в методе `dispatch` с входящим параметром `$query`, который является исходным запрашиваемым `url`.

2.1.3 Автозагрузка классов

Для улучшения качества кода, во всех приложениях обычно используется автозагрузка классов.

Автозагрузка классов – это стандарт, при котором описанные разработчиком классы хранятся в отдельном файле по одному, а при обращении к ним, метод автозагрузки ищет путь к файлу с помощью пространств имён (абстракции).

В языке PHP популярен стандарт автозагрузки PSR-4, который ищет файл напрямую из корневого каталога, например: при обращении к `‘/common/Load::file($file);’` будет произведен поиск класса `Load` в папке `common` начиная от корня приложения и запуск метода `file` с входящим параметром `$file`.

Для упрощения автозагрузки было принято решение сократить пространства имён для целевых каталогов убрав из них пространства ядра и приложения:

- ядра,
- контроллера,
- модели,
- аддонов.

Таким образом пространство имени `/core/classes/Router` сокращается до `/core/Router`, `/app/addons/header/Profile` до `addons/header/Profile` и т. д.

Автозагрузка является модулем ядра и размещена в каталоге `/core/modules/autoload.php` и вызывается загрузчиком. Код автозагрузчика представлен на рисунке (33).

```

spl_autoload_register(function($class){
    $file = str_replace('\\', '/', $class);
    if(is_file(ROOT.'/'.$file.'.php')) {
        require_once(ROOT.'/'.$file.'.php');
    } else {
        $eFile = explode('/', $file);
        $prefix = $eFile[0];
        unset($eFile[0]);

        $file = implode('/', $eFile);
        switch ($prefix) {
            case 'controller':
                $file = APP.'/controllers/'.$file.'.php';
                break;
            case 'model':
                $file = APP.'/models/'.$file.'.php';
                break;
            case 'addon':
                $file = APP.'/addons/'.$file.'.php';
                break;
            case 'core':
                $file = CORE.'/classes/'.$file.'.php';
                break;
        }
        if(file_exists($file)) {
            require_once $file;
        }
    }
}

```

Рисунок 33 – Код автозагрузки классов

2.1.4 Маршрутизация

Обращение к серверу происходит по паттерну фронт-контроллера, в связи с чем маршрутизация становится необходимой частью загрузки страницы.

Маршрутизация разбирает входящий url, обрабатывает его и выполняет определенный код по результату. В MVC, маршрутизация переносит пользователя на нужный контроллер, который, в свою очередь, вызывает связанную с запросом модель, обрабатывающую данные, а затем вид, таким образом формируя страницу.

Для обеспечения работы маршрутизации, необходимо составить таблицу маршрутов с базовыми правилами обработки, а также маршруты

перехода по страницам. Таблица маршрутов находится в пространстве приложения под наименованием routes.php (рисунок 34).

```
<?php

core\Router::add('^news/?<id:[0-9]+>?$', ['controller' => 'News', 'action' => 'view']);
core\Router::add('^auth$', ['controller' => 'Profile', 'action' => 'auth']);

core\Router::add('^error/?<id:[0-9]+>?$', ['controller' => 'Main', 'action' => 'error']);
//правила по умолчанию

//адрес главной страницы
core\Router::add('^$', ['controller' => 'Main']);
//обработка адресов
core\Router::add('^<controller:[a-z-]+>/?<action:[a-z-]+>?$');
```

Рисунок 34 – Файл routes

Метод `core\Router::add` добавляет паттерн регулярного выражения в таблицу маршрутов и описывает место направления в случае совпадения с url. Для лучшего восприятия, над регулярными выражениями была выстроена абстракция над паттернами регулярных выражений.

Подобная абстракция существует в фреймворке Yii2 и является одним из самых удобных решений среди MVC фреймворков для упрощения написания маршрутов.

Для реализации вывода содержимого из базы данных на отдельной странице, разработана возможность передачи параметров в маршрутах.

Метод расширения таблицы маршрутов представлен на рисунке (35).

```
class Router {
    //таблица маршрутов
    protected static $routes = [];
    //текущий маршрут
    protected static $route = [];

    //добавление маршрута - в resources/routes.php
    public static function add($regexp, $route = []) {
        //обработка регулярного выражения из сокращенной
        $regexp = str_replace('<', '(<P<', $regexp);
        $regexp = str_replace('>', ')', $regexp);
        $regexp = str_replace(':', '>', $regexp);
        self::$routes[$regexp] = $route;
        return true;
    }
}
```

Рисунок 35 – Метод добавления маршрутов и свойства хранения

Для примера, паттерн типа `^(?P<controller>[a-z-]+)$` заменяется на `^<controller:[a-z-]+>$` и для длинных url, данная абстракция значительно упростит конструирование маршрутов.

Метод `matchRoute` необходим для сопоставления url со всеми маршрутами в таблице маршрутов и, в случае успеха, выведении в свойство `route` соответствующего данному url контроллера и действия (рисунок 36).

Для упрощения разработки, в таблице маршрутов необходима возможность отсутствия заполнения действия для контроллера. В случае, когда в таблице маршрутов не указано действие, вызывается стандартный `indexAction`.

```
//поиск текущего запроса в таблице маршрутов
private static function matchRoute($url) {
    foreach(self::$routes as $pattern => $route) {
        if(preg_match("#$pattern#i", $url, $matches)) {
            foreach ($matches as $key => $value) {
                if(is_string($key)) {
                    $route[$key] = $value;
                }
            }
            $route['controller'] = self::format('controller', $route['controller']);
            if(isset($route['action'])) {
                $route['action'] = self::format('action', $route['action']);
            } else {
                $route['action'] = 'index';
            }
            self::$route = $route;
            return true;
        }
    }
    return false;
}
```

Рисунок 36 – Метод `matchRoute`

Метод `dispatch` вызывается из фронт-контроллера после выполнения всех компонентов загрузчика. Данный метод предназначен для перенаправления пользователя по маршрутам.

В первую очередь, вызывается статичный метод `matchRoute`, который проверяет, корректен ли url для перенаправления. В случае, где распознать url в таблице маршрутов не удастся, метод `dispatch` возвращает ошибку уровня 404 и вызывает метод `error`.

В случае где совпадение в таблице маршрутов было найдено, метод форматирует добавленные в свойство \$route названия контроллера и действия для соответствия стандартам PSR наименования методов и классов, а затем создает объект класса контроллера, с последующим вызовом метода действия.

Метод форматирования наименований можно увидеть на рисунке (37).

```
//форматирование controller и action
private static function format($type, $string) {
    switch ($type) {
        case 'controller':
            $string = str_replace('-', ' ', $string);
            $string = ucwords($string);
            $string = str_replace(' ', '', $string);
            return $string;
            break;
        case 'action':
            $string = str_replace('-', ' ', $string);
            $string = ucwords($string);
            $string = str_replace(' ', '', $string);
            $string = lcfirst($string);
            return $string;
            break;
        case 'query':
            if($string) {
                $params = explode('&', $string, 2);
                if(false === strpos($params[0], '=')) {
                    return rtrim($params[0], '/');
                } else {
                    return '';
                }
            } else {
                return '';
            }
            break;
    }
}
```

Рисунок 37 – Метод format

Если запрошенный класс не существует, метод вызовет ошибку уровня 404 с указателем на отсутствие класса и аналогично с отсутствием метода действия.

Метод dispatch создаёт экземпляр класса контроллера, определённого с помощью маршрутизации.

Реализация метода dispatch представлена на рисунке (38).

```
//переход по маршруту
public static function dispatch($url) {
    $url = self::format('query', $url);
    if(self::matchRoute($url)) {
        $controller = 'controller\\'.self::$route['controller'];
        if(class_exists($controller)) {
            $object = new $controller(self::$route);
            $action = self::$route['action'].'Action';
            if(method_exists($object, $action)) {
                $object->$action();
                $object->getView();
            } else {
                if(method_exists($object, 'indexAction')) {
                    $object->indexAction();
                    $object->getView();
                } else {
                    self::error($controller.'.action');
                }
            }
        } else {
            self::error($controller);
        }
    } else {
        self::error();
    }
}
```

Рисунок 38 – Метод dispatch

При любой из ошибок перехода по маршруту, выводится статус ошибки – 404 и создается объект класса Main с действием error, который содержит ссылку на страницу 404.

Если при этом в конфиге свойство debug_mode хранится значение true, то на отображаемой странице будет выводиться причина вызванной ошибки.

В переменную errorCode на странице, записывается значение 404 и принудительно вызывается запуск метода контроллера.

Данная система отображения ошибок необходима для точного отображения данных на странице 404. Альтернативой системе служит правило в таблице маршрутов, на которое всегда реагирует система маршрутизации при отсутствии совпадений с существующими маршрутами.

Реализация метода error отображена на рисунке (39).

```

public static function error($debug = 'default') {
    http_response_code(404);
    $object = new \controller\Main(['controller' => 'Main', 'action' => 'error']);
    if(config('debug_mode') == true) {
        if($debug) {
            $object->vars['debug'] = $debug;
        } else {
            $object->vars['debug'] = self::$route;
        }
    }
    $object->vars['errorCode'] = 404;
    $object->errorAction();
    $object->getView();
}

```

Рисунок 39 – Метод error

2.1.4 Реализация MVC

Система разрабатывается на архитектурном MVC паттерне с возможностью разделения функционала системы на отдельные компоненты с собственной логикой.

Основными компонентами системы являются:

- контроллеры,
- модели,
- виды,
- шаблонизация,
- аддоны [18].

Для их применения и реализации в целом, следовало создать базовые классы основных компонентов (рисунок 40).

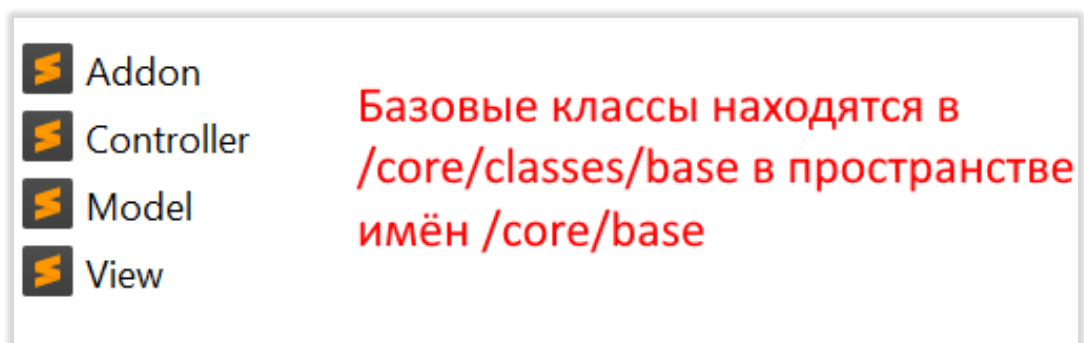


Рисунок 40 – Расположение базовых классов

Абстрактный базовый класс контроллера включает свойства и методы (рисунок 41):

- маршрута, по которому был открыт контроллер,
- вида, который должен быть вызван из данного маршрута,
- шаблона,
- переменных, передаваемых из контроллера.
- конструктор, заполняющий свойства собственного класса из передаваемого значения маршрута,
- метод getView, вызывающий вид,
- базовый метод с постфиксом Action, который необходим во всех контроллерах и открываться при отсутствии вызываемого маршрутом Action.

```
<?php

namespace core\base;

abstract class Controller {
    public $route;
    public $view;
    public $template;
    public $vars = [];

    public function __construct($route) {
        $this->route = $route;
        $this->view = $route['action'];
    }

    public function getView() {
        $object = new View($this->route, $this->template, $this->view);
        $object->render($this->vars);
    }

    public function indexAction() {

    }
}
```

Рисунок 41 – Код базового контроллера

Абстрактный базовый класс контроллера включает в себя размещение свойств: маршрута, вида, подключаемого шаблона, дополнительных переменных.

Метод indexAction необходим как маркер, вызывающий страницу с параметрами по умолчанию при отсутствии действия в таблице маршрутов.

Базовый класс View предназначен для запуска контента страницы и шаблона, передаваемого из контроллера.

Часть кода класса View представлена на рисунке (42).

```
<?php

namespace core\base;

class View {
    public $route;
    public $view;
    public $template;

    public function __construct($route, $template = '', $view = '') {
        $this->route = $route;
        $this->template = $template ?: config('default_template');
        if($view === false) {
            $this->view = false;
        } else {
            $this->view = $view;
        }
    }
}
```

Рисунок 42 – Часть кода core\base\View

Основной частью кода класса View является метод render. Данный статичный метод предназначен для отображения контента страницы вместе с шаблоном.

Подразумевается что система может включать себя несколько файлов стилей и скриптов, разбросанных внутри контента страницы, поэтому метод смещает стили и скрипты по маркерам в шаблоне (рисунок 43).

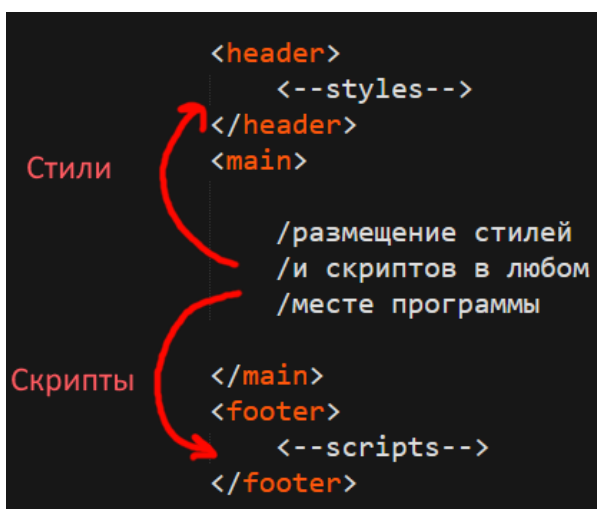


Рисунок 43 – Принцип смещения скриптов и стилей

Для правильного размещения стилей и скриптов следует пользоваться глобальными функциями `addStyle` и `addScript`.

Одной из самых важных задач метода является передача переменных из контроллера в глобальные переменные страницы, для дальнейшего обращения и использования в других частях кода. Обратиться к переданным из контроллера переменным можно с помощью массива `GLOBALS['vars']` (рисунок 44).

```
public function render($vars) {
    $view_path = VIEW.'/'.$this->route['controller'].'/'.$this->view.'.php';
    $template_path = TEMPLATE.'/'.$this->template;
    $GLOBALS['vars'] = $vars;
    ob_start();
    if(is_file($view_path)) {
        if(is_file($template_path.'/header.php') and is_file($template_path.'/footer.php')) {
            require $template_path.'/header.php';
            require $view_path;
            require $template_path.'/footer.php';
        } else {
            echo \core\Router::error('template');
        }
    } else {
        echo \core\Router::error('view', $view_path);
    }
    $content = ob_get_clean();

    if ($this->view !== false) {
        $content = \core\structure\Document::placeStyles($content);
        $content = \core\structure\Document::placeScripts($content);
        echo $content;
    }
}
```

Рисунок 44 – Метод `render` класса `\core\base\View`

Дополнительный функционал в системе реализуется через систему дополнений – `addons`. Каждый `addon` может иметь несколько шаблонов и моделей, которые могут комбинироваться в любом порядке.

В случае где дополнительные шаблоны `addon`'а не требуются, подключается шаблон с именем `default`.

Дополнения не являются полноценной реализацией MVC, так контроллеры и виды в них размещены в одном файле, что также позволяет загружать PHP код из любых источников без проблем с совместимостью.

Реализация подключения дополнений представлена на рисунке (45).

```

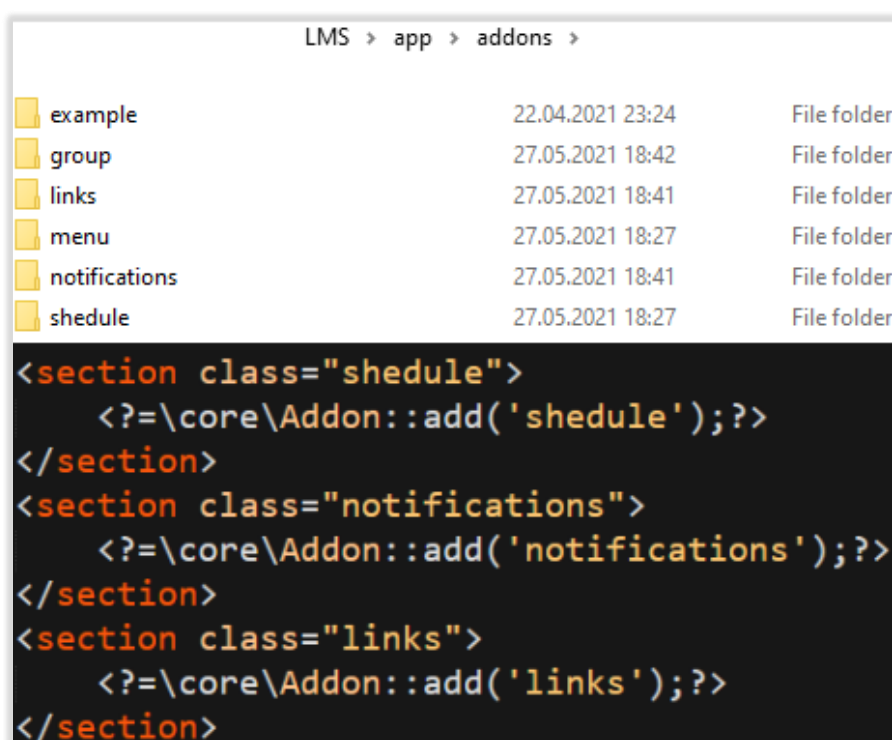
public static function add($name, $template = 'default') {
    if(is_file(ADDON.'/'.$name.'/'.$template.'/main.php')) {
        require(ADDON.'/'.$name.'/'.$template.'/main.php');
    } else {
        echo "Не удалось обнаружить аддон ".$name;
    }
}
}

```

Рисунок 45 – Реализация подключения дополнений

Было решено отказаться от хранения информации о подключенных аддонах, так как данный функционал не несёт пользы без полноценной системы информирования о разных компонентах системы. Однако, разработка функционала хранения информации об аддонах позволит производить настройку информации в них, подобно «Компонентам» в Битрикс-24.

Пример размещения компонентов и их подключения представлен на рисунке (46).



The image shows a file manager interface with a breadcrumb path 'LMS > app > addons >'. Below the path is a list of folders: 'example', 'group', 'links', 'menu', 'notifications', and 'shedule'. Each folder has a timestamp and a label 'File folder'. Below the list is a code snippet showing how to connect these folders using the 'add' function.

```

<section class="shedule">
    <?=\core\Addon::add('shedule');?>
</section>
<section class="notifications">
    <?=\core\Addon::add('notifications');?>
</section>
<section class="links">
    <?=\core\Addon::add('links');?>
</section>

```

Рисунок 46 – Подключение и размещение дополнений

Каждое дополнение может иметь собственные классы, с пространством имён '\addon\'название дополнения\'название класса'. Классы в дополнениях возможно размещать в шаблонах и корне.

2.1.4 Шаблоны

Во время разработки было принято решение ограничить количество шаблонов до 2 – основного и шаблона без шапки.

Структура файлов шаблона представлена на рисунке (47).

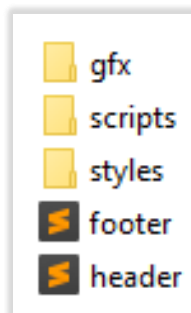


Рисунок 47 – Структура файлов шаблона

Реализация шаблонов не привязана к логике и может содержать любые каталоги и файлы. Единственным ограничением является размещение вёрстки в файлах header и footer (рисунок 48).

```
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Document</title>
7     <!--styles-->
8     <?php
9
10    addStyle(ROOT.'/resources/fonts/roboto/roboto.css');
11    addStyle(__DIR__.'/styles/base.css');
12
13    ?>
14 </head>
15 <body>
16     <header>
17         <div class="content">
18             <a href="/" class="logo"></a>
20         </div>
21     </header>
22     <main>
23
24     </main>
25     <hr class="divider">
26     <footer>
27
28     </footer>
29     <!--scripts-->
30 </body>
31 </html>
```

header.php

footer.php

Рисунок 48 – Основные файлы шаблона

2.1.4 База данных

Для подключения к базе данных используется прослойка PHP Data Objects (PDO) [19]. Классическим примером подключения базы данных является подключение через ООП паттерн Singleton (рисунок 49).

```
private static $instance;

//реализация синглтона
public static function instance() {
    if(self::$instance === null) {
        self::$instance = new self;
    }

    return self::$instance;
}
```

Рисунок 49 – Реализация Singleton

В классе базы данных приватный конструктор создаёт новый объект PDO с параметрами, передаваемыми из конфига (рисунок 50).

```
private function __construct() {
    $options = [
        \PDO::ATTR_ERRMODE => \PDO::ERRMODE_EXCEPTION,
        \PDO::ATTR_DEFAULT_FETCH_MODE => \PDO::FETCH_ASSOC,
    ];
    $this->pdo = new \PDO(
        config('dsn'),
        config('db_user'),
        config('db_pass'),
        $options
    );
}
```

Рисунок 50 – Конструктор класса Database

Для создания запросов в базу данных используется 2 публичных метода, выполняющие функции извлечения данных и проверки на выполнение запроса (рисунок 51).


```

//проверка на выполнение запроса
public function execute($query) {
    $stmt = $this->pdo->prepare($query);
    return $stmt->execute();
}

//запрос с выводом данных
public function query($query) {
    $stmt = $this->pdo->prepare($query);
    $res = $stmt->execute();
    if($res !== false) {
        return $stmt->fetchAll(\PDO::FETCH_ASSOC);
    }

    return [];
}

```

Рисунок 51 – Методы извлечения данных

2.1.4 Режим отладки

Для отладки была разработана система переключения режима вывода ошибок [20]. Данная возможность позволит быстрее реагировать на возникшие неполадки в системе и скрыть их отображение при работе сайта в действии.

Включить режим отладки можно с помощью файла config.php, где размещена строка “debug_mode” с принимаемым булевым значением true или false.

Если режим отладки включен, загрузка всех стилей на странице, добавленных через глобальную функцию addScript, осуществляется в режиме принудительного обновления.

В режиме отладки происходит расширенное отображение ошибок. Для добавления возможности отображения ошибки в коде необходима проверка на конфиг с последующим выводом ошибки.

Отображение стилей со включенным режимом отладки представлено на рисунке (52).

```
<link rel="stylesheet" href="/resources/fonts/roboto/roboto.css?1622505810">
<link rel="stylesheet" href="/app/templates/main/styles/base.css?1622505810">
<link rel="stylesheet" href="/app/templates/main/styles/menu.css?1622505810">
<link rel="stylesheet" href="/app/addons/shedule/default/style.css?1622505810">
<link rel="stylesheet" href="/app/addons/notifications/default/style.css?1622505810">
<link rel="stylesheet" href="/app/addons/links/default/style.css?1622505810">
<link rel="stylesheet" href="/app/addons/menu/default/style.css?1622505810">
<link rel="stylesheet" href="/app/addons/group/default/style.css?1622505810">
```

Рисунок 52 - Отображение ссылок с режимом отладки

Данное решение позволяет во время разработки не сбрасывать кэш страниц для корректного отображения множество раз измененных стилей на странице.

В случае если режим отладки выключен, включается кеширование стилей, которое позволяет быстрее загружать страницу во время повторной загрузки стилей (рисунок 53).

```
<link rel="stylesheet" href="/resources/fonts/roboto/roboto.css">
<link rel="stylesheet" href="/app/templates/main/styles/base.css">
<link rel="stylesheet" href="/app/templates/main/styles/menu.css">
<link rel="stylesheet" href="/app/addons/shedule/default/style.css">
<link rel="stylesheet" href="/app/addons/notifications/default/style.css">
<link rel="stylesheet" href="/app/addons/links/default/style.css">
<link rel="stylesheet" href="/app/addons/menu/default/style.css">
<link rel="stylesheet" href="/app/addons/group/default/style.css">
```

Рисунок 53 – Отображение ссылок без режима отладки

2.1.4 Страница авторизации

Главной страницей системы (домен/), является страница авторизации. Система авторизации разрабатывается в пространстве модели приложения и не является частью ядра, что позволяет в любой момент модифицировать алгоритм входа в систему.

Страница входа доступна только неавторизованным пользователям. В случае когда пользователь авторизован в системе и пытается открыть главную страницу, происходит перенаправление в главное меню.

Страница авторизации выполнена в простом стиле и имеет ссылки на страницу FaQ и техподдержки (рисунок 54).

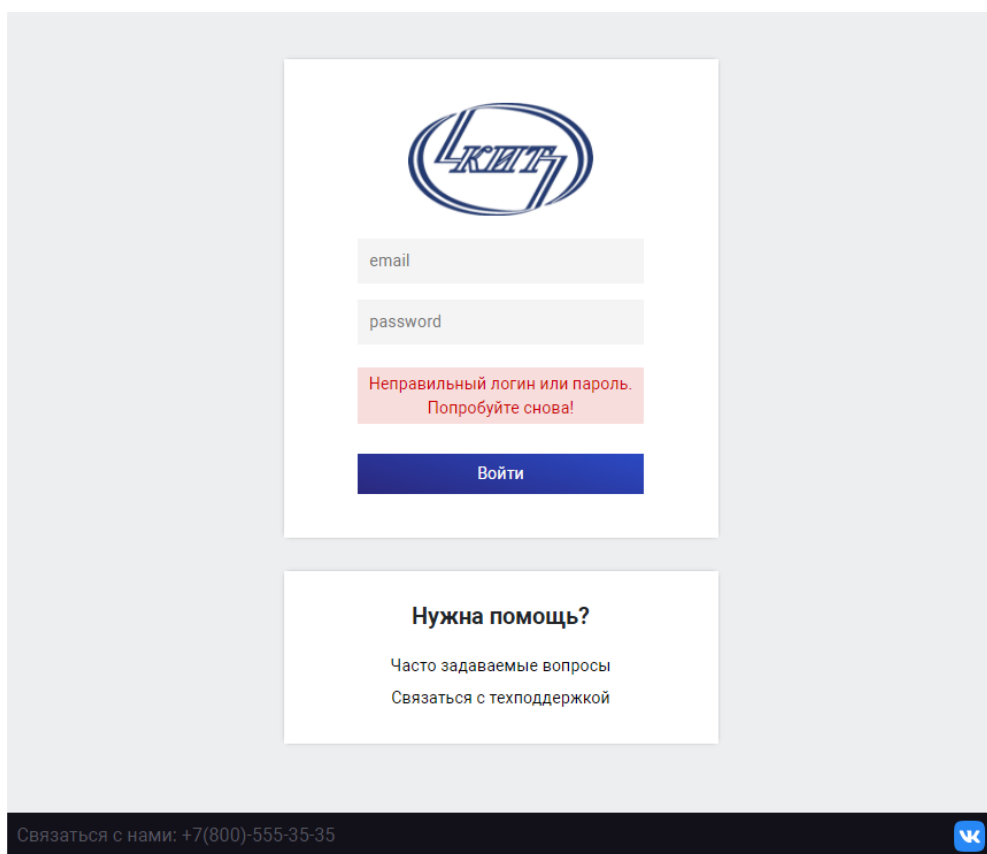


Рисунок 54 – Страница авторизации

Для страницы авторизации задействован шаблон с именем light, в котором нет шапки страницы.

При введении неправильного логина или пароля в окне авторизации выводится красный блок с ошибкой.

2.1.4 Аккаунты пользователей

Каждый авторизованный пользователь в системе имеет своё значение доступа. Права доступа записаны в базе данных пользователей и содержат

значение доступа от 1 до 100, где 31 – права доступа студента, а 100 – главного администратора (рисунок 55).

Admin	81-100
Teacher / Content Manager	51-80
Student	31-50
Guest	1-30

Рисунок 55 – права пользователя в системе

Для дальнейшего расширения системы, принято решение оставить свободным пространство доступа 1-30 для возможного использования пользователями, не относящимися к учебному процессу, например, для авторизации поступающим студентам (для прохождения курсов или ознакомления).

Система оценки доступа размерностью в 100 единиц позволяет создавать вспомогательные уровни доступа в любой момент без переписывания системы авторизации и смещения уровня доступа других групп.

2.1.4 Страница 404

Во время ошибки перехода по маршрутам, вызывается страница 404 (рисунок 56). Главная задача страницы – уведомлять об ошибке и перенаправлять пользователя на главную страницу.

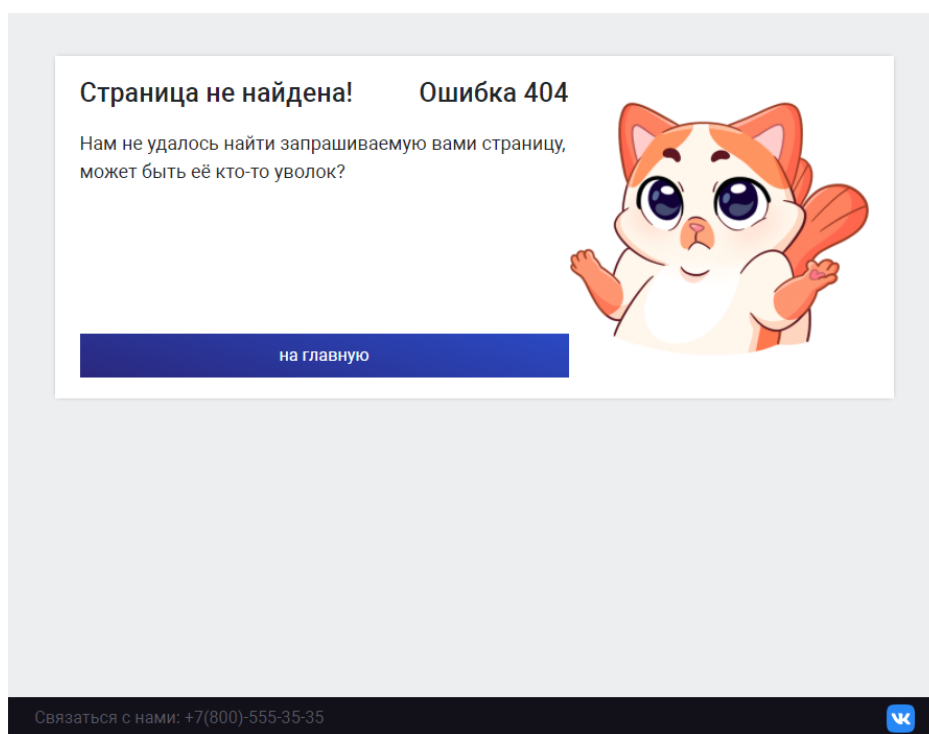


Рисунок 56 – Страница 404

Страница 404 является важнейшей частью системы и оформление напрямую влияет на восприятие сайта в целом. При нажатии на кнопку возвращения происходит проверка на авторизацию, которая определяет маршрут перехода.

2.1.4 Главное меню

Авторизованный пользователь перенаправляется в главное меню системы (маршрут /menu).

Меню содержит 2 основных раздела:

- виджеты, выводящие информацию,
- меню, содержащее ссылки на основной функционал системы.

В шапке сайта отображается полное ФИО авторизованного пользователя и страница выхода из система. Система дополнений позволяет в полной мере расширить функционал внутренних страниц, например для добавление цепочки навигации.

Главное меню системы представлено на рисунке (57).

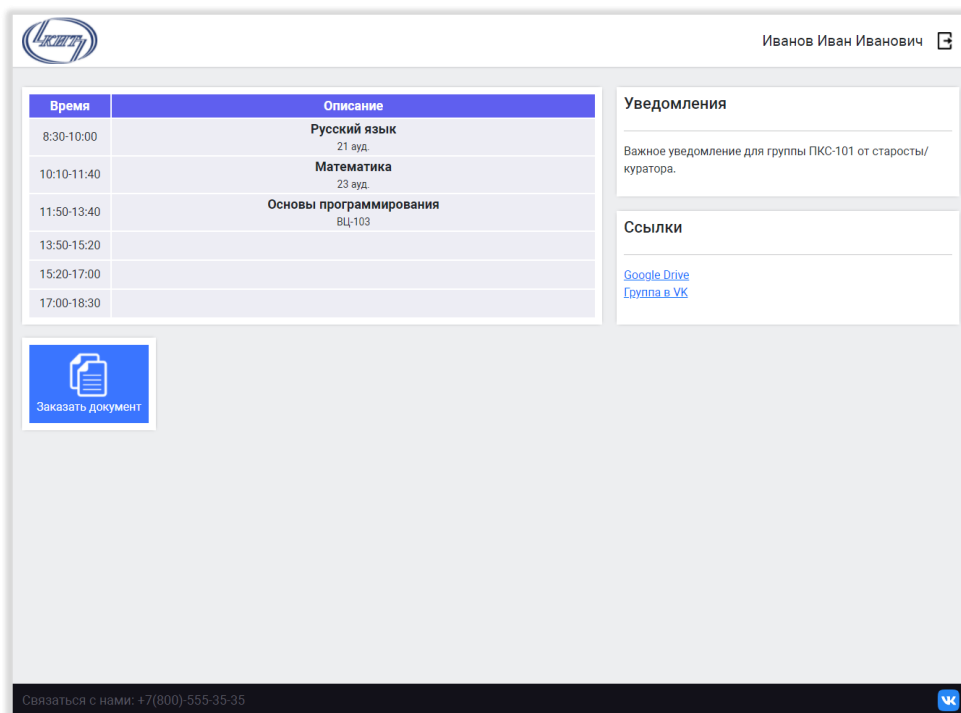


Рисунок 57 –Страница /menu

Для реализации минимального функционала были разработаны виджеты:

- расписание,
- уведомления группы,
- полезные ссылки группы,
- меню с ссылками на функции системы.

Виджеты и пункты привязаны к уровням доступа.

Пункты меню реализованы с помощью вспомогательного файла `map.php`, расположенного в корневом каталоге аддона (рисунок 58).

```
return [  
  [  
    'link' => 'menu',  
    'text' => 'Заказать документ',  
    'color' => '#454796',  
    'img' => 'doc',  
    'access' => 1  
  ]  
];
```

Рисунок 58 – Содержание `map.php`

2.1.4 Тестирование и слияние на main-branch

После завершения разработки прототипа системы управления обучением необходимо перенести изменения на главную ветку репозитория – main. Перед слиянием, было выполнено тестирование на совместимость с мобильными устройствами (рисунок 59).

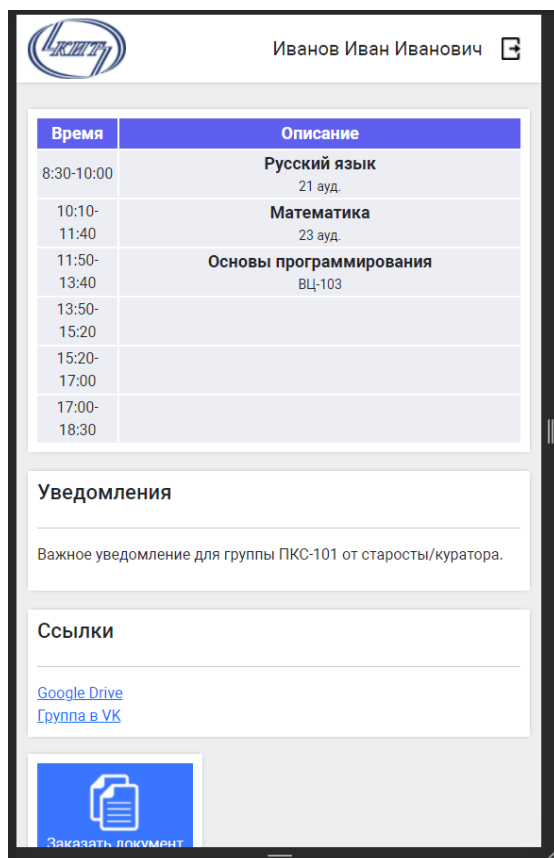


Рисунок 59 – Отображение мобильной версии

Мобильная версия системы работает стабильно без графических ошибок.

За время разработки проекта совершено 20 коммитов (рисунок 60).

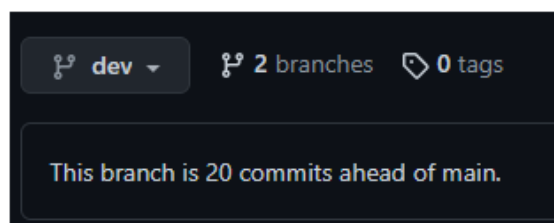


Рисунок 60 – Статистка в GitHub

3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

3.1 Затраты проекта

Команда проекта состоит из 1 фуллстак-разработчика, работающего над визуальной и логической частями проекта. Для расчета затрат используется почасовая оплата труда. Средняя ставка с учетом почасовой оплаты фуллстак-разработчика, по данным hh.ru, составляют 1600 руб/час [21].

Для определения затрат реализации проекта, был проведён расчет стоимости работ по разработке системы. Временные затраты на разработку представлены в таблице (1).

Таблица 1 – Расчет оценки времени разработки

Название	Длительность, ч.
Разработка ядра системы	36
Разработка бизнес-логики приложения	12
Разработка дизайна	7
Вёрстка шаблонов и страниц	5
Тестирование	3
Всего	63

Итоговое количество времени, затраченного на разработку – 63 ч. При нормированном 8-ми часовом рабочем дне, длительность разработки составит 8 дней.

Для уточнения стоимости каждого вида работ, проведён расчет стоимости каждой работы. Стоимость каждой из проведённых работ, при средней ставке фуллстак-разработчика, представлена в таблице (2).

Таблица 2 – Расчет стоимости проведённых работ

Название	Стоимость, руб.	Затраты, руб.
Разработка ядра системы	57 600	65 088
Разработка бизнес-логики приложения	19 200	21 696
Разработка дизайна	11 200	1 456
Вёрстка шаблонов и страниц	8 000	9 040
Тестирование	4 800	5 424
Итог	100 800	113 904

Подсчёт стоимости работ произведён до вычета НДС и отражает полные расходы на разработку.

Для более точных подсчётов затрат необходимо учесть амортизацию за использование оборудования во время разработки системы, затраты на расходные материалы и затраты на электроэнергию (таблица 3, 4, 5).

Таблица 3 – Амортизационные отчисления за время разработки

Наименование	Количество, ед.	Срок службы, лет.	Стоимость единицы, руб.	Амортизация за время разработки, руб.
Ноутбук	1	4	140 000	767
Принтер	1	8	6 000	16
Итого				783

Таблица 4 – Затраты на расходные материалы

Наименование	Единица измерения	Количество, шт.	Стоимость, руб.	Общая стоимость, руб.
Бумага А4	Упаковка х500	1	345	345
Картридж для принтера	Шт.	1	899	899
Итого				1244

Таблица 5 – Затраты на электроэнергию во время разработки

Наименование	Мощность, кВт.	Количество, ед.	Потребление в день, кВт.
Ноутбук	0,096	1	0,768
Принтер	0,084	1	0,672
Итого			1,440

Учитывая средний тариф энергии от компании Мосэнергосбыт 5,77 руб/кВт/ч, стоимость электроэнергии в день составляет 8,3 руб. Затраты на электроэнергию во время разработки составляют 66,4 руб.

Для извлечения прибыли, необходимо выполнить доработку и интеграцию системы в учебно-образовательную организацию. Учитывая опыт разработки прототипа, длительность работ по доработке и интеграции системы составит не менее 48 часов. Оценка стоимости работ по доработке и

интеграции производится по средней ставке фуллстак разработчика и составляет не мене 76 800 руб.

Интеграция и доработка системы – необходимые действие для использования системы управления обучением в образовательном процессе. На этапе внедрения выполняется анализ и разработка функционала в зависимости от статистических данных, собранных за время тестового периода эксплуатации системы. Общий перечень работ по интеграции представлен в таблице (6).

Таблица 6 – Список работ для внедрения с оценочной стоимостью и продолжительность

Наименование	Длительность, дней	Стоимость, руб.
Интеграция в образовательную организацию	4	22 000
Публичный тест системы и сбор статистики	28	8 000
Доработка системы в соответствии с собранной статистикой	18	64 000
Тестирование и устранение ошибок	4	12 000
Подготовка к выпуску и релиз системы	2	6 000
Заполнение данных и информирование студентов	7	18 000
Поиск и устранение ошибок после релиза	60	30 000
Итого	123	120 000

Затраты на этап внедрения системы оценивается в 975 руб/день. Поиск и устранение ошибок после релиза является дополнительной мерой обеспечения качества системы, проводимой после запуска системы в эксплуатацию.

Ожидаемый выход системы управления обучением приносит прибыль с момента релиза и заполнения данных в систему – спустя 63 дня после начала

работ по внедрению. Суммарные затраты на разработку и внедрение системы представлены в таблице (7).

Таблица 7 – Смета на разработку и внедрение системы

Наименование	Стоимость
Разработка прототипа	113 904
Амортизация	783
Расходные материалы	1 244
Электроэнергия	66
Внедрение	120 000
Итого	235 997

Для эксплуатации системы необходимо постоянное сопроводительное обслуживание. Список годовых расходов на обслуживание представлен в таблице (8).

Таблица 8 – Стоимость годового обслуживания системы

Наименование	Стоимость, руб.
Обслуживание домена	1 400
Хостинг сервера	7 000
Администрирование	164 000
Итого	172 400

3.2 Прогнозирование прибыли

Для прогнозирования прибыли необходимо собрать информацию об общем количестве студентов, поступающих в образовательную организацию. По статистике набора 2020-2021, в колледж прибыло 298 новых студентов. На момент разработки проекта, стоимость обучения в образовательной организации составляет 119 000 руб.

Студенты, поступившие в 2020 году, приносят прибыль организации в размере 35 462 000 руб.

Стоимость обучения в колледжах и университетов с системами управления обучением выше в среднем на 35 000 руб. Данные по стоимости обучения собраны с сайта msk.postupi.online [22].

При более слабом повышении стоимости обучения на 25 000 руб, стоимость обучения составит 144 000 руб/чел. При расчёте на 298 прибывших в 2020 году студентов, прибыль организации в следующих учебных годах возрастёт на 7 450 000 руб на каждый курс вплоть до 4-го.

Учитывая стоимость обслуживания системы, прибыль организации от эксплуатации системы в первый год составит 7 277 600 руб.

Стоимость разработки, доработки и интеграции составляет 235 997 руб, что полностью окупает первый поток поступающих студентов. Прибыль, с учетом стоимости разработки и интеграции системы, в первый год эксплуатации системы составляет 7 041 603 руб.

ЗАКЛЮЧЕНИЕ

В современном мире системы управления обучением являются эффективным решением проблемы информирования студентов.

В теоретической части проведён анализ аналога LMS, выделены сильные и слабые стороны анализируемой системы. Для реализации проекта выбраны инструменты разработки и поставлены задачи для разработки прототипа.

В проектной части произведена разработка прототипа системы обучением с помощью выбранных инструментов разработки.

Основной объём работ составила разработка ядра системы. На момент окончания разработки, архитектура системы полностью сформирована, создана модель обработки информации из баз данных и реализован функционал доступа в систему. В тестовом режиме разработан функционал системы:

- вывод расписания из базы данных,
- вывод уведомлений и ссылок группы,
- система онлайн-заказа документов.

Цель проекта – разработка прототипа и для полноценного использования системы управления обучением, прототи нуждается в доработке функционала и полноценной интеграции в образовательную организацию.

В экономической части составлен список затрат на разработку, прогноз расходов на внедрение и реализацию системы. Для анализа потенциальной прибыли от применения системы управления обучением, проведён прогноз повышения доходов. Система управления обучением окупит затраты на разработку и обслуживание в первый год эксплуатации.

Разработанный прототип обладает перспективами:

- расширение существующего функционала,

- повышение уровня социального взаимодействия в системе,
- сбор статистики пользователей системы,
- расширение круга пользователей (вновь поступающие студенты),
- интеграция на главный сайт образовательной организации.

:

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. vc.ru [Электронный ресурс]. URL: <https://vc.ru> (дата обращения: 18.05.2021)
2. LMS HSE [Электронный ресурс]. URL: <https://lms.hse.ru> (дата обращения: 19.05.2021)
3. Habr.ru [Электронный ресурс]. URL: <https://habr.com/>(дата обращения: 20.05.2021)
4. Html5book [Электронный ресурс]. URL: <https://html5book.ru> (дата обращения: 21.05.2021)
5. HTML5CSS.ru [Электронный ресурс]. URL: <https://html5css.ru> (дата обращения: 21.05.2021)
6. Сойер М.Д. Новая большая книга CSS / М.Д. Сойер – СПб.: Изд-во Питер, 2018. –512с.
7. sass-scss.ru [Электронный ресурс]. URL: <https://sass-scss.ru> (дата обращения: 21.05.2021)
8. Webref.ru [Электронный ресурс]. URL: <https://webref.ru> (дата обращения: 21.05.2021)
9. php.net [Электронный ресурс]. URL: <https://www.php.net> (дата обращения: 22.05.2021)
10. sql.ru [Электронный ресурс]. URL: <https://www.sql.ru> (дата обращения: 24.05.2021)
11. С. Макконел Совершенный код. Изд-во: Русская редакция 2019 – 896с.
12. StackOverflow [Электронный ресурс]. URL: <https://stackoverflow.com> (дата обращения: 26.05.2021)
13. Дакетт Дж. HTML и CSS Разработка и дизайн веб-сайта / Дакетт Джон – М. Эксмо, 2019. –384с.
14. Refactoring.guru [Электронный ресурс]. URL: <https://refactoring.guru> (дата обращения: 28.05.2021)

15. github.com [Электронный ресурс]. URL: <https://github.com> (дата обращения: 29.05.2021)
16. web-creator.ru [Электронный ресурс]. URL: <https://web-creator.ru> (дата обращения: 29.05.2021)
17. Wikipedia [Электронный ресурс]. URL: <https://ru.wikipedia.org/> (дата обращения: 30.05.2021)
18. Ruseller [Электронный ресурс]. URL: <https://ruseller.com/> (дата обращения: 30.05.2021)
19. docs.microsoft.com [Электронный ресурс]. URL: <https://docs.microsoft.com> (дата обращения: 31.05.2021)
20. Tproger.ru [Электронный ресурс]. URL: <https://tproger.ru> (дата обращения: 31.05.2021)
21. hh.ru [Электронный ресурс]. URL: <https://hh.ru> (дата обращения: 31.05.2021)
22. msk.postupi.online [Электронный ресурс]. URL: <https://msk.postupi.online> (дата обращения: 31.05.2021)

ПРИЛОЖЕНИЕ А

Templates/main/style/base.scss

```
* {  
  
    font-family: Roboto;  
  
}  
  
body {  
  
    display: flex;  
  
    flex-direction: column;  
  
    align-items: center;  
  
    min-height: 100vh;  
  
    background-color: #EDEEF0;  
  
}  
  
header {  
  
    width: 100%;  
  
    height: 72px;  
  
    background: #ffffff;  
  
    box-shadow: 0 0 4px -2px #000000;  
  
    section.container {  
  
        display: flex;  
  
        align-items: center;  
  
        justify-content: space-between;  
  
        height: 100%;  
  
    }  
  
}  
  
.profile {  
  
    display: flex;  
  
    align-items: center;
```

```
height: 100%;

p {

    margin: 0;

    margin-right: 10px;

    font-size: 20px;

}

a {

    padding: 20px 10px;

    img {

        height: 24px;

    }

}

}

a.logo {

    img {

        height: 64px;

    }

}

main {

    display: flex;

    flex-direction: column;

    align-items: flex-start;

    padding-top: 28px;

}
```

```
hr.divider {  
  
    flex-grow: 1;  
  
    border: none;  
  
}  
  
footer {  
  
    padding: 8px 0;  
  
    width: 100%;  
  
    background: #121119;  
  
    div {  
  
        display: flex;  
  
        flex-direction: row;  
  
        justify-content: space-between;  
  
        align-items: center;  
  
    }  
  
    p {  
  
        margin: 0;  
  
        font-size: 18px;  
  
        color: #585760;  
  
    }  
  
    a {  
  
        img {  
  
            height: 32px;  
  
        }  
  
    }  
  
}
```

Templates/main/header.php

```
<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Document</title>

    <link                href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css"
rel="stylesheet"                integrity="sha384-
+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x"
crossorigin="anonymous">

    <!--styles-->

    <?php addStyle(__DIR__.'./styles/base.css');?>

    <?php addStyle(ROOT.'./resources/fonts/roboto/roboto.css');?>

</head>

<body>

    <header>

        <section class="container">

            <a href="/" class="logo">

            </a>

            <div class="profile">

                <p>Иванов Иван Иванович</p>

                <a href="/"></a>

            </div>

        </section>

    </header>

    <main class="container">
```

Templates/main/header.php

```
</main>

<hr class="divider">

<footer>

    <div class="container">

        <p>Связаться с нами: +7(800)-555-35-35</p>

        <a href="https://vk.com/unikitpage"><svg width="30" viewBox="0 0 30 30"
fill="none" xmlns="http://www.w3.org/2000/svg"><path d="M0 14.375C0 7.599 0 4.21 2.105 2.105 4.21 0
7.6 0 14.375 0h1.25c6.776 0 10.165 0 12.27 2.105C30 4.21 30 7.6 30 14.375v1.25c0 6.776 0 10.165-2.105
12.27C25.79 30 22.4 30 15.625 30h-1.25c-6.776 0-10.165 0-12.27-2.105C0 25.79 0 22.4 0 15.625v-1.25z"
fill="#2787F5"></path><path fill-rule="evenodd" clip-rule="evenodd" d="M8.125 9.375H5.938c-.625 0-
.75-.294-.75-.619 0-.579-.741 3.453 3.453 7.253 1.807 2.596 4.354 4.003 6.671 4.003 1.391 0 1.563-.313 1.563-
.85v-1.962c0-.625-.132-.75-.572-.75-.325 0-.88-.162 2.179 1.413 1.483 1.484 1.727 2.149 2.561 2.149h2.188c.625
0-.938-.313-.757-.93-.197-.614-.905-1.506-1.845-2.563-.51-.602-1.274-1.251-1.506-1.576-.325-.417-.232-
.602 0-.973 0 0 2.665-3.754 2.943-5.029.14-.463 0-.804-.662-.804h-2.187c-.556 0-.813-.294-.952-.619 0 0-1.112
2.711-2.688 4.472-.51-.51-.742-.673-1.02-.673-.139 0-.34-.163-.34-.626v-4.334c0-.556-.161-.804-.625-.804h-
3.438c-.347 0-.556-.258-.556-.503 0-.527-.788-.649-.869 2.132v3.221c0-.707-.127-.835-.406-.835-.741 0-2.545-
2.724-3.615-5.84-.21-.606-.42-.851-.979-.851z" fill="fff"></path></svg></a>

    </div>

</footer>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js"
integrity="sha384-gtEjrD/SeCtmISkJKNUaaKMoLD0//EIJ19smozuHV6z3Iehds+3Ulb9Bn9Plx0x4"
crossorigin="anonymous"></script>

<!--scripts-->

</body>

</html>
```

ПРИЛОЖЕНИЕ Б

Router.php

```
namespace core;

class Router {

    //таблица маршрутов

    protected static $routes = [];
```

```

//текущий маршрут

protected static $route = [];


//добавление маршрута - в resources/routes.php

public static function add($regex, $route = []) {

    //обработка регулярного выражения из сокращенной записи

    $regex = str_replace('<', '?P<', $regex);

    $regex = str_replace('>', ')', $regex);

    $regex = str_replace(':', '>', $regex);

    self::$routes[$regex] = $route;

    return true;

}

//получение текущего маршрута

public static function get() {

    return self::$route;

}

//получение таблицы маршрутов

public static function getAll() {

    return self::$routes;

}

//поиск текущего запроса в таблице маршрутов

private static function matchRoute($url) {

    foreach(self::$routes as $pattern => $route) {

        if(preg_match("#$pattern#i", $url, $matches)) {

            foreach ($matches as $key => $value) {

```

```

        if(is_string($key)) {

            $route[$key] = $value;

        }

    }

    $route['controller'] = self::format('controller', $route['controller']);

    if(isset($route['action'])) {

        $route['action'] = self::format('action', $route['action']);

    } else {

        $route['action'] = 'index';

    }

    self::$route = $route;

    return true;

}

}

return false;

}

//переход по маршруту

public static function dispatch($url) {

    $url = self::format('query', $url);

    if(self::matchRoute($url)) {

        $controller = 'controller\\'.self::$route['controller'];

        if(class_exists($controller)) {

            $object = new $controller(self::$route);

            $action = self::$route['action'].'.Action';

            if(method_exists($object, $action)) {

```

```

        $object->$action();

        $object->getView();

    } else {

        if(method_exists($object, 'indexAction')) {

            $object->indexAction();

            $object->getView();

        } else {

            self::error($controller.'.action');

        }

    }

} else {

    self::error($controller);

}

} else {

    self::error();

}

}

public static function error($debug = 'default') {

    http_response_code(404);

    $object = new \controller\Main(['controller' => 'Main', 'action' => 'error']);

    if(config('debug_mode') == true) {

        if($debug) {

            $object->vars['debug'] = $debug;

        } else {

            $object->vars['debug'] = self::$route;

        }

    }

}

```



```

        }

    }

    $object->vars['errorCode'] = 404;

    $object->errorAction();

    $object->getView();

}

//форматирование controller и action

private static function format($type, $string) {

    switch ($type) {

        case 'controller':

            $string = str_replace('-', ' ', $string);

            $string = ucwords($string);

            $string = str_replace(' ', '', $string);

            return $string;

            break;

        case 'action':

            $string = str_replace('-', ' ', $string);

            $string = ucwords($string);

            $string = str_replace(' ', '', $string);

            $string = lcfirst($string);

            return $string;

            break;

        case 'query':

            if($string) {

                $params = explode('&', $string, 2);

```

```

        if(false === strpos($params[0], '=')) {

            return rtrim($params[0], '/');

        } else {

            return "";

        }

    } else {

        return "";

    }

    break;

default:

    echo "Не определен тип форматирования или строка форматирования
пуста";

    break;

}

}

}

```

Database.php

```

<?php

namespace core;

class Database {

    private $pdo;

    private static $instance;

    private function __construct() {

        $options = [

            \PDO::ATTR_ERRMODE => \PDO::ERRMODE_EXCEPTION,

```

```

        \PDO::ATTR_DEFAULT_FETCH_MODE => \PDO::FETCH_ASSOC,

    ];

    $this->pdo = new \PDO(

        config('dsn'),

        config('db_user'),

        config('db_pass'),

        $options

    );

}

//реализация синглтона

public static function instance() {

    if(self::$instance === null) {

        self::$instance = new self;

    }

    return self::$instance;

}

//проверка на выполнение запроса

public function execute($query) {

    $stmt = $this->pdo->prepare($query);

    return $stmt->execute();

}

//запрос с выводом данных

public function query($query) {

    $stmt = $this->pdo->prepare($query);

    $res = $stmt->execute();

```

```
        if($res !== false) {  
            return $stmt->fetchAll(\PDO::FETCH_ASSOC);  
        }  
        return [];  
    }  
}
```