Компьютерная Академия ШАГ

Разработка ПО

Курсовой проект WEB-программирование

Тема проекта:"Электронный дневник для учета оценок"

Куратор Паначев Александр

Ученик Дудиков Иван

Алматы январь 2024

21 января 2024 г.

1 Вступление

Современные технологии прочно вошли в нашу жизнь. Любая сфера жизни и деятельности человека не обходится без каких-либо современных технических решений, приспособлений, инструментов, источников данных и информации. Одним из таких инструментов является глобальная сеть Интернет. Благодаря сети Интернет мы не только можем получать доступ к любой информации, но и плотно социализироваться с любым человеком или организацией, так или иначе представленным на просторах сети. Мы можем общаться посредством различных мессенджеров, обмениваться контентом в социальных сетях, получать информацию о товарах и услугах, покупать и продавать необходимые нам товары, заказывать продукты и услуги, не выходя из дома, осуществлять платежи и пользоваться различными сервисами, которые можно перечислять достаточно долго. Сложно оценить удобство и ценность этих технологий в нашей повседневной жизни.

Одним из сегментов сетевых технологий является сфера автоматизации бизнес-процессов, среди которых можно отметить электронный документооборот и видео конференции. Сейчас для того, чтобы обсудить с коллегами текущие вопросы и проблемы, нет необходимости собираться в вместе или лететь в другие города, достаточно провести видеоконференцию, не выходя из дома и даже не вставая с кресла. Электронный документооборот существует достаточно давно. В первую очередь автоматизации подверглись бизнес- процессы, требующие большого объёма текстовой информации, представленной на бумажных носителях. Такие объёмы бумаги требуют больших затрат на хранение, каталогизацию и поиск нужных данных, а также не способствуют охране окружающей среды, так как производство бумаги губит не только деревья, но и происходит загрязнение окружающей среды продуктами и отходами бумажного производства.

Для удобства хранения информации были созданы базы данных, а также системы их управления и использования. Бумажные документы были преобразованы во множество текстовых и графических файлов, которые разместились внутри базы данных и стали доступны на экране компьютера, а не на столе или в шкафу. Время поиска нужной информации стала зависеть только от производительности серверов и скорости передачи данных по локальной сети. Со временем такие системы вышли за пределы зданий и офисов, и стали работать в глобальном формате. Теперь базы данных стали использовать в любых сферах жизни, где требуется хранение о обработка больших объемов информации. Из деловой и финансовой сферы автоматизация вошла во все сферы нашей жизни. Социальные сети, телевидение и радио, интернет-магазины, поликлиники, транспорт, производство, торговля, сфера обслуживания, и много другое. Однако учебным процессам, по моему мнению, не уделено достаточно внимания. Именно поэтому целью моего проекта является автоматизация учебного процесса.

1.1 Цель

Каждый из нас учился в школе. Каждый из нас помнит, сколько приходилось носить учебников и тетрадей. Вести дневник, записывать домашние задания, писать в тетради. В высших учебных заведения и колледжах вести объёмные конспекты, курсовые и лабораторные работы. Не говоря уже о работе учителей по контролю и проверкам выполнения домашних заданий.

Цель данного проекта – автоматизировать учебный процесс любого уровня. Для этого требуется создать сервис на базе web приложения и базы данных, которое позволяло бы вести электронный дневник, выдавать учебные материалы обучаемым, преподавателям проверять и контролировать учебный процесс, проверять выполнение домашних заданий, а также выставлять оценки за выполнения заданий и прохождение этапов обучения. Вести систему рейтингов и отчетов. Размещение базы данных предполагается на выделенном серверном пространстве с достаточным и расширяемым дисковым массивом для хранения большого и регулярно пополняемого объёма информации. Резервное копирование, поддержание работоспособности и защиту от несанкционированного доступа возложить на провайдера интернет -услуг, на площадке которого будет размещена база данных. Взаимодействие с пользователями осуществляется через web-интерфейс зарегистрированного доменного имени в зоне .kz или .edu.

1.2 Целевая аудитория

Необходимость в таком программном обеспечении требуется достаточно давно. Люди привыкли к общению в социальных сетях, мессенджерах, но при этом взаимодействие на уровне ученикреподаватель остается недостаточно технологичным. Учебные материалы, тетради и учебники – все приходится носить с собой на занятия. Поэтому данный проект возможно использовать в различных образовательных учреждениях. Предполагаемая целевая аудитория: Школы общеобразовательные, специализированные. Лицеи всех специальностей. Колледжи и специальные учреждения среднего образования. ВУЗы и институты. Учреждения профильного образования, академии длительного срока обучения.

Расширение функциональных возможностей проекта позволит объединить в себе множество образовательных учреждений по принципу социальной сети, где в последствии возможно предоставлять услуге по приобретению учебных принадлежностей, образовательных материалов, рекламы учебных заведений, курсов и других услуг, включая оплату за обучение. Это позволит еще больше расширить охват аудитории, заинтересованной в получении образования и повышения своего уровня знаний. На любом этапе внедрения и работы возможно будет расширять функциональные возможности, менять дизайн и удобство использования, в дальнейшем возможна разработка мобильного приложения, где будет дублирован функционал web версии. При достаточном уровне охвата возможно добавление функции социальной сети с возможностью хранения личных данных учеников, обмен сообщениями и группировка по интересам и увлечениям и дальнейшем расширении целевой аудитории.

2 Использованные языки

2.1 PHP

PHP - это скриптовый язык программирования, изначально разработанный для создания динамических веб-страниц. Вот несколько ключевых аспектов и функций PHP:

2.1.1 Веб-разработка

PHP широко используется для создания динамических веб-страниц и веб-приложений. Сценарии PHP выполняются на стороне сервера, что позволяет генерировать HTML и отправлять его клиенту.

2.1.2 Синтаксис

Синтаксис PHP похож на синтаксис C, Java и Perl, что облегчает изучение для разработчиков, знакомых с другими языками программирования.

2.1.3 Интеграция с HTML

PHP код встраивается в HTML-страницы, что обеспечивает совместную работу с разметкой и динамическое формирование содержимого страницы.

2.1.4 Поддержка различных баз данных

PHP обладает встроенной поддержкой многих систем управления базами данных (СУБД), таких как MySQL, PostgreSQL, SQLite и других, что делает его мощным инструментом для работы с данными.

2.1.5 Обработка форм

PHP часто используется для обработки данных из форм на веб-страницах. Он может получать данные из форм, выполнять проверку, отправлять электронные письма и обрабатывать загруженные файлы.

2.1.6 Создание сессий и куки

PHP предоставляет средства для работы с сессиями, что позволяет сохранять состояние пользователя между запросами, и управление куками для хранения информации на стороне клиента.

2.1.7 Множество встроенных функций

PHP поставляется с обширным набором встроенных функций, облегчающих работу с строками, массивами, файлами, работой с датами и временем, шифрованием и многими другими.

2.1.8 Frameworks

PHP часто используется в сочетании с фреймворками, такими как Laravel, Symfony и CodeIgniter, чтобы упростить и ускорить процесс разработки веб-приложений. Мы же используем Laravel.

2.1.9 Общее

Все эти особенности делают PHP важным инструментом веб-разработки, особенно при создании динамических и интерактивных веб-приложений.

2.2 Laravel

Laravel - это современный фреймворк для веб-разработки, построенный на языке программирования PHP. Он предоставляет разработчикам инструменты и абстракции, упрощающие создание сложных и масштабируемых веб-приложений. Вот несколько ключевых особенностей Laravel:

2.2.1 Eloquent ORM (Object-Relational Mapping)

Laravel предоставляет Eloquent ORM, который позволяет вам взаимодействовать с базой данных, используя объектно-ориентированный подход. Это упрощает выполнение запросов и работу с данными, так как они представлены в виде моделей.

2.2.2 Маршрутизация и Контроллеры

Laravel обеспечивает четкую и удобную маршрутизацию, которая позволяет определять, какие действия выполнять для каждого URL-адреса. Контроллеры Laravel облегчают организацию и структурирование кода для обработки запросов.

2.2.3 Библиотека Blade для шаблонов

Blade - это инструмент в Laravel, который помогает создавать веб-страницы. Простыми словами:

1)Красивые страницы

Blade делает создание веб-страниц более простым и понятным. С его помощью можно создавать красивый и удобный для восприятия код.

2)Повторное использование кода:

Blade позволяет создавать "шаблоны"для часто используемых элементов веб-страниц, таких как заголовки или формы. Это экономит время и уменьшает количество повторяющегося кода.

3)Простые условия и циклы:

Вы можете использовать простые конструкции в Blade для создания условий и циклов, что упрощает работу с данными и их отображением на странице.

4)Интеграция с PHP:

Blade легко интегрируется с PHP, поэтому если у вас уже есть опыт в программировании на PHP, вы сможете легко использовать его в Blade.

2.2.4 Artisan

Artisan - это инструмент командной строки, который предоставляет множество удобных функций. Вот что он делает:

1)Автоматизация задач:

Artisan помогает автоматизировать рутинные задачи разработки, такие как создание файлов, миграции базы данных, управление зависимостями и многое другое.

2)Управление базой данных:

Вы можете использовать Artisan для создания и управления миграциями базы данных. Это удобно, когда вы добавляете новые функции или изменяете структуру данных.

3)Создание своих команд:

Artisan позволяет вам создавать свои собственные команды, что может быть полезно при выполнении специфических задач, уникальных для вашего проекта.

4)Генерация кода:

Artisan может создавать стандартный код для вас. Например, команда php artisan make:controller создаст основу для вашего контроллера.

5)Тестирование:

Artisan помогает вам проводить тестирование вашего кода, чтобы убедиться, что он работает правильно. Эти инструменты делают разработку на Laravel более эффективной и удобной, позволяя сосредотачиваться на создании функциональности, а не на рутинных задачах.

2.2.5 Модульность и Пакеты

В Laravel поддерживается концепция модульности и использование пакетов. Простыми словами:

1)Модульность:

Модульность в Laravel означает способность организовывать ваш код в небольшие, независимые блоки, называемые модулями. Это улучшает структуру и обслуживание вашего кода, делая его более читаемым и легким для изменений.

2)Пакеты:

Пакеты - это готовые к использованию наборы кода, созданные сообществом или вами самими, которые можно легко внедрить в ваш проект. В Laravel использование пакетов позволяет избегать написания кода с нуля, ускоряя процесс разработки.

3)Composer:

Laravel интегрируется с инструментом Composer, который является пакетным менеджером для PHP. Composer позволяет управлять зависимостями (в том числе пакетами Laravel) и подключать сторонние пакеты к вашему проекту.

4)Простая установка:

Установка пакетов в Laravel - это простая задача благодаря Composer. Вы можете указать необходимый пакет в файле зависимостей (composer.json) и обновить свой проект. Расширение функциональности:

Использование пакетов позволяет легко расширять функциональность вашего приложения без необходимости внесения значительных изменений в код. Вы можете добавить пакет для работы с изображениями, обработки платежей, авторизации и многое другое.

5)Сообщество и Поддержка:

Laravel имеет активное сообщество разработчиков, которые создают и обновляют множество полезных пакетов. Это обеспечивает стабильность и поддержку ваших зависимостей.

2.2.6 Аутентификация и Авторизация

Laravel предоставляет встроенные средства для реализации системы аутентификации и авторизации. Это включает в себя готовые компоненты, такие как контроллеры, маршруты и представления для управления пользователями.

2.2.7 Middlewares

Middlewares - это промежуточные обработчики в Laravel, которые могут выполняться до или после обработки HTTP-запроса. В нескольких словах:

1)Обработка запросов: Middlewares могут выполнять код до того, как запрос попадет к контроллеру или действию, что полезно для выполнения определенных действий перед обработкой запроса.

2)Модификация запросов:

Они предоставляют возможность изменять данные запроса, добавлять заголовки, модифицировать параметры и многое другое.

3)Авторизация:

Middlewares часто используются для проверки прав доступа пользователя к определенным ресурсам. Например, проверка, имеет ли пользователь право на просмотр определенной страницы.

4)Логирование:

Можно использовать Middlewares для регистрации информации о запросах, например, для создания журнала действий пользователя или отслеживания ошибок.

5)Защита от атак:

Middlewares могут предоставлять защиту от определенных видов атак, фильтруя и проверяя входящие данные.

Middlewares обеспечивают гибкость и контроль над процессом обработки запросов в Laravel, что позволяет разработчикам встраивать дополнительную логику на уровне HTTP-запросов.

2.2.8 Тестирование

Тестирование в Laravel важно для обеспечения стабильности и уверенности в работоспособности кода.

1)Обеспечение надежности:

Тестирование в Laravel - это процесс создания специального кода, который проверяет, работает ли ваше приложение так, как задумано. Это гарантирует, что приложение остается стабильным и надежным.

2)PHPUnit:

Laravel использует PHPUnit для написания и запуска тестов. PHPUnit предоставляет методы для проверки функций вашего кода, а также обработку исключений и ошибок.

3)Unit-тесты:

Проверяют отдельные компоненты вашего кода (например, методы классов) для обеспечения их корректной работы в изоляции. Функциональные тесты:

Проверяют взаимодействие различных компонентов вашего приложения. Например, они могут проверять, что ваши маршруты и контроллеры взаимодействуют корректно.

4)Тестирование API:

Laravel обеспечивает средства для тестирования API, чтобы удостовериться, что ваши вебсервисы предоставляют ожидаемые результаты.

2.2.9 Общее

Laravel облегчает многие аспекты веб-разработки, предоставляя элегантный синтаксис и множество готовых компонентов. Фреймворк активно поддерживается сообществом разработчиков, что обеспечивает его постоянное обновление и развитие.

2.3 SQL

SQL (Structured Query Language):

SQL - язык программирования, используемый для работы с реляционными базами данных. В контексте Laravel и веб-разработки, SQL играет важную роль во взаимодействии с базой данных. Вот ключевые аспекты использования SQL в Laravel:

2.3.1 Операторы SQL

Операторы SQL:

SQL (Structured Query Language) предоставляет разнообразные операторы для выполнения операций над данными в реляционных базах данных. Вот основные операторы SQL:

SELECT:
SELECT column1, column2 FROM table WHERE condition;
Используется для извлечения данных из одной или нескольких таблиц.

INSERT:
INSERT INTO table (column1, column2) VALUES (value1, value2);
Добавляет новые строки данных в таблицу.

UPDATE:
UPDATE table SET column1 = value1 WHERE condition;
Модифицирует существующие данные в таблице.

DELETE:
DELETE FROM table WHERE condition;
Удаляет данные из таблицы.

WHERE:
SELECT * FROM table WHERE condition;
Используется для фильтрации данных, определяя условия, которым должны соответствовать строки.

AND, OR, NOT:
SELECT * FROM table WHERE condition1 AND condition2;
Логические операторы, используемые для комбинирования условий.

ORDER BY:
SELECT * FROM table ORDER BY column ASC;
Сортирует результаты запроса по указанным колонкам в порядке возрастания или убывания.

GROUP BY:
SELECT column, COUNT(*) FROM table GROUP BY column;
Группирует строки с одинаковыми значениями в определенной колонке.

JOIN:
SELECT * FROM table1 INNER JOIN table2 ON table1.column = table2.column;

Объединяет данные из двух или более таблиц на основе условия.

LIKE:
SELECT * FROM table WHERE column LIKE 'pattern';
Используется для поиска строк, содержащих определенные символы.

IN:
SELECT * FROM table WHERE column IN (value1, value2, value3);
Позволяет фильтровать данные, основываясь на списке значений.

2.4 HTML (Hypertext Markup Language)

HTML - это язык разметки, используемый для создания структуры и представления содержания веб-страниц. Он состоит из набора тегов, каждый из которых определяет различные элементы на странице.

HTML является основным строительным блоком веб-страниц и веб-приложений. Он обеспечивает структурирование контента, определение элементов, таких как заголовки, абзацы, ссылки, изображения, формы и многие другие. HTML также обеспечивает основу для визуального представления данных в браузере.

2.4.1 Структура Документа

HTML определяет базовую структуру веб-документа с использованием тегов <html>, <head>, и <body>. Это формирует начало и конец контента страницы.

2.4.2 Теги и Элементы

HTML использует теги для разметки различных элементов на странице. Тег <p> представляет абзац, <a> - ссылку, <img> - изображение, и так далее.

2.4.3 Формы и Ввод данных

Элементы форм, такие как <form>, <input>, и <button>, позволяют пользователям взаимодействовать с веб-страницей, отправляя данные на сервер.

2.4.4 Списки и Маркировка

HTML поддерживает упорядоченные (<ol>) и неупорядоченные (<ul>) списки, а также элементы списка <li> для каждого пункта.

2.4.5 Структурирование и Семантика

Использование семантических тегов, таких как <header>, <nav>, <section>, добавляет структуру и смысл контенту.

2.4.6 Интеграция с CSS и JavaScript

HTML интегрируется с CSS для стилизации и JavaScript для добавления динамического поведения, обеспечивая лучший пользовательский опыт.

2.4.7 Гиперссылки и Навигация

Использование тега <a> для создания гиперссылок обеспечивает навигацию между страницами и ресурсами.

2.4.8 Значение

HTML является фундаментальным языком веб-разметки, обеспечивая создание структурированных и информативных веб-страниц. В сотрудничестве с PHP и Laravel, HTML играет ключевую роль в разработке динамичных и интерактивных веб-приложений.

2.5 JavaScript

JavaScript - это высокоуровневый, интерпретируемый язык программирования, который обеспечивает взаимодействие с элементами веб-страницы. Он добавляет динамичность и интерактивность веб-приложениям.

JavaScript позволяет создавать динамические веб-страницы, реагирующие на действия пользователя, обеспечивает валидацию форм, анимацию и взаимодействие с сервером без перезагрузки страницы.

JavaScript играет ключевую роль в создании интерактивных и динамичных веб-приложений, взаимодействуя с HTML и CSS. В контексте Laravel, JavaScript также может быть интегрирован для обогащения пользовательского опыта.

2.6 CSS (Cascading Style Sheets)

CSS - это язык стилей, используемый для определения визуального представления веб-страниц. Он управляет стилями, цветами, шрифтами и расположением элементов на странице.

CSS позволяет разработчикам задавать внешний вид веб-страниц, делая их более привлекательными и пользовательски дружелюбными. Он обеспечивает разделение структуры и стиля веб-документа.

2.6.1 Селекторы

Использование селекторов для выбора HTML-элементов, к которым будут применяться стили.

2.6.2 Свойства и Значения

Определение свойств (например, color, font-size) и их значений для задания конкретных стилей.

2.6.3 Каскадность и Приоритет

CSS работает по принципу каскадности, где стили могут наследоваться и переопределяться с учетом приоритета.

2.6.4 Box Model

Понимание концепции "Box Model которая описывает как каждый элемент представлен как прямоугольный блок с контентом, отступами, границей и полем.

2.6.5 Позиционирование и Размещение

Использование свойств position и display для контроля расположения элементов на странице.

2.6.6 Анимации и Переходы

Применение анимаций и переходов для создания плавных изменений визуального представления при взаимодействии с элементами.

2.6.7 Медиазапросы

Применение медиазапросов для создания адаптивных стилей, которые изменяются в зависимости от размера экрана устройства.

2.6.8 Шрифты и Изображения

Управление стилями шрифтов (font-family, font-weight) и работа с изображениями для создания богатого визуального контента.

2.6.9 Общее

CSS позволяет разработчикам творчески подходить к визуализации веб-приложений, обеспечивая пользовательский опыт с точки зрения дизайна и внешнего вида. В сотрудничестве с HTML и JavaScript, CSS формирует полноценное веб-приложение.

3 Показ проекта

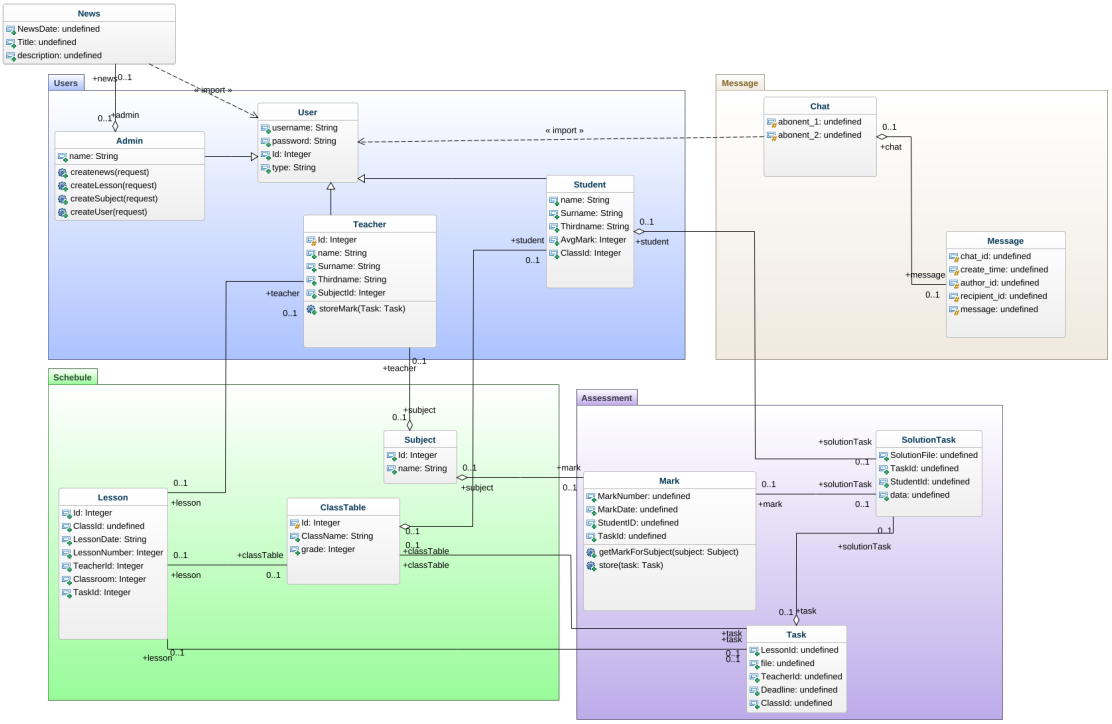

Рис. 1: UML проекта

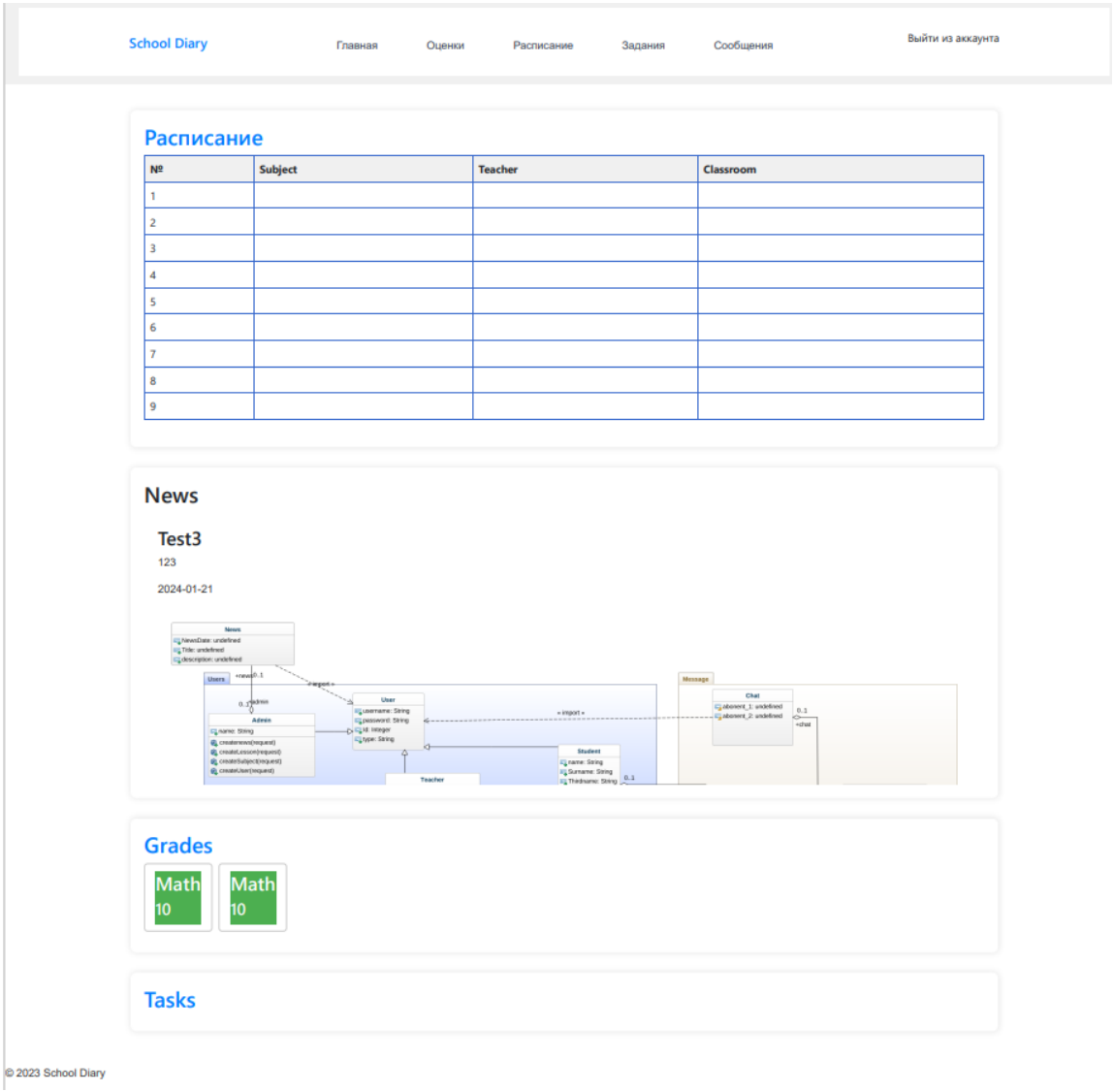

Рис. 2: Главная страница

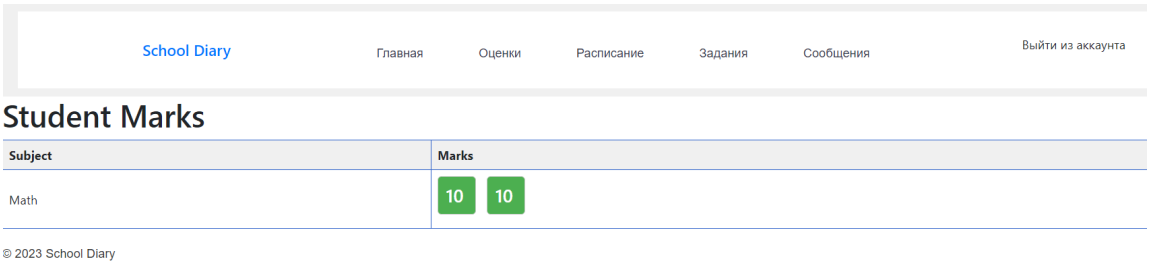

Рис. 3: Оценки

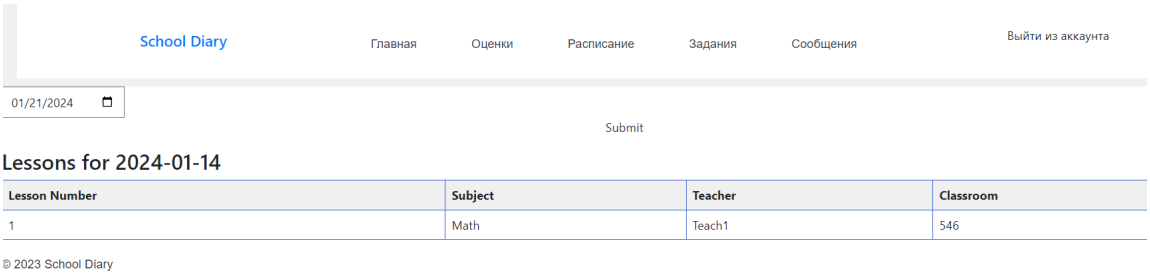

Рис. 4: Расписание

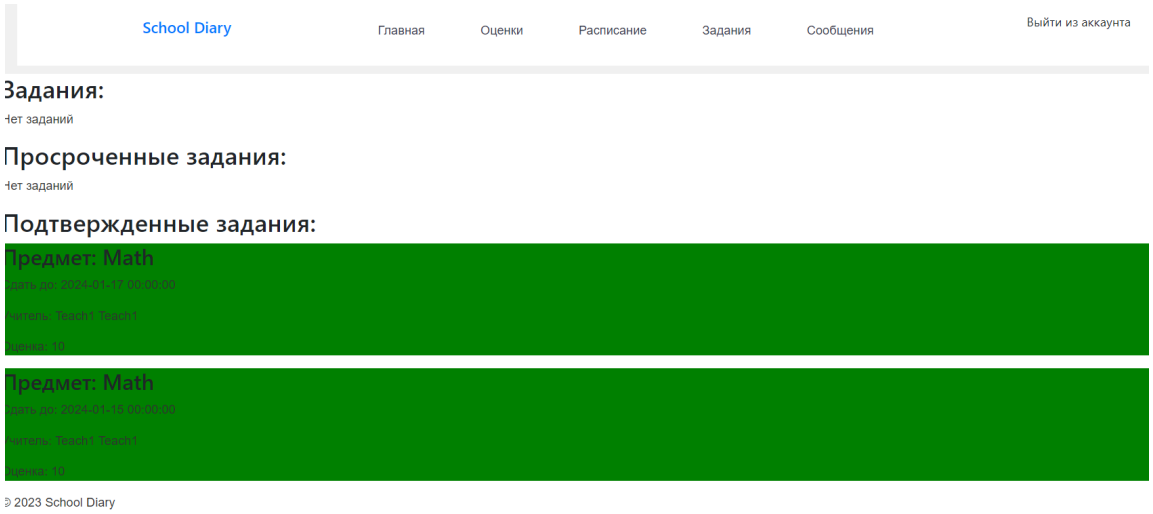

Рис. 5: Задание

Рис. 6: Чат с учителем

4 Системные требования

4.1 Системные Требования для OpenServer

OpenServer - это локальный сервер, который предоставляет среду для разработки веб-приложений. Вот общие системные требования для его установки и использования:

Операционная Система: OpenServer доступен для установки на операционные системы Windows и macOS.

Процессор: Рекомендуется использование многоядерного процессора для оптимальной производительности.

Оперативная Память (RAM): Рекомендуемый объем оперативной памяти - не менее 2 ГБ.

Свободное Пространство на Жестком Диске: Для установки OpenServer и необходимых компонентов требуется свободное место на жестком диске, обычно не менее 1 ГБ.

Версия PHP: OpenServer поддерживает различные версии PHP. При необходимости выбора конкретной версии для проекта, уточните совместимость.

4.2 Системные Требования для Visual Studio Code

Visual Studio Code - легкий, многофункциональный редактор кода от Microsoft. Вот минимальные системные требования для его работы:

Операционная Система: Поддерживается на Windows, macOS и Linux.

Процессор: 1.6 ГГц или более мощный процессор.

Оперативная Память (RAM): Рекомендуемый объем оперативной памяти - не менее 4 ГБ для комфортной работы.

Свободное Пространство на Жестком Диске: Для установки и хранения расширений рекомендуется не менее 200 МБ свободного места.

Разрешение Экрана: Рекомендуется разрешение экрана не менее 1280x768 пикселей.

5 Заключение

В процессе работы над проектом мы достигли комплексного решения, разработанного для улучшения процесса ведения учета и обеспечения взаимодействия между учениками и педагогами. В ходе разработки были реализованы следующие ключевые функциональности:

5.1 Электронный дневник

Внедрена система электронного дневника, обеспечивающая удобный доступ к актуальной информации об успеваемости и домашних заданиях для учеников.

5.2 График занятий

Разработан удобный график занятий, обеспечивающий структурированный и визуально понятный обзор расписания для всех участников образовательного процесса. А также ео создание и изменение в зависимости от нужд школы.

5.3 Оценивание и отчетность

Введена система онлайн-оценивания и генерации отчетов, облегчающая процесс оценки и предоставления обратной связи учениками.

6 Листинг программы

```
public function createUser(Request $request)
{
    //

    $userType = $request->input('userType');

    //

    switch ($userType) {
        case 'student':
            return $this->createNewUserStudent($request);
            break;
        case 'teacher':
            return $this->createNewUserTeacher($request);
            break;
        case 'admin':
            return $this->createNewUserAdmin($request);
            break;
    }
}

//
public function createNewUserStudent(Request $request)
{
    //
    $request->validate([
        'username' => 'unique:user',
        'password' => 'min:6',
        'subjectId' => 'exists:subject,id',
    ]);
```

```
29
30      //
31      $data = [
32          'name' => $request->input('teacherName'),
33          'Surname' => $request->input('teacherSurname'),
34          'Thirdname' => $request->input('teacherThirdname'),
35          'SubjectID' => $request->input('subjectId')
36      ];
37
38      $newTeacher = Teacher::createTeacher($data);
39
40      //
41      $user = new User();
42      $user->username = $request->input('username');
43      $user->password = Hash::make($request->input('password'));
44      $user->UserType = 'teacher';
45      $user->UserId = $newTeacher->id;
46      $user->save();
47
48      return redirect()->back();
49  }
50
51  //
52  public function createNewUserTeacher(Request $request)
53  {
54      //
55      $validator = Validator::make($request->all(), [
56          'username' => 'unique:user',
57          'password' => 'min:6',
58          'subjectId' => 'exists:subject,id',
59      ]);
60
61      //                                                  ,


62      if ($validator->fails()) {
63          return redirect()->back()->withErrors($validator)->withInput()->
        withErrors(['common-error' => 'There are errors in the form.']);
64      }
65
66      //
67      $data = [
68          'name' => $request->input('teacherName'),
69          'Surname' => $request->input('teacherSurname'),
70          'Thirdname' => $request->input('teacherThirdname'),
71          'SubjectID' => $request->input('subjectId')
72      ];
73
74      $newTeacher = Teacher::createTeacher($data);
75
```

```
76      //
77      $user = new User();
78      $user->username = $request->input('username');
79      $user->password = Hash::make($request->input('password'));
80      $user->UserType = 'teacher';
81      $user->UserId = $newTeacher->id;
82      $user->save();
83
84      return redirect()->back();
85  }
86
87  //
88  public function createNewUserAdmin(Request $request)
89  {
90      //
91      $user = new User();
92      $user->username = $request->input('username');
93      $user->password = Hash::make($request->input('password'));
94      $user->UserType = 'admin';
95
96      //
97      $data = [
98          'name' => $request->input('adminName')
99      ];
100
101     $newAdmin = AdminModel::createAdmin($data);
102
103     //
104     $user->UserId = $newAdmin->id;
105     $user->save();
106
107     return redirect()->back();
108 }
```

Функционал

createUser(Request *request*) :

- Метод принимает запрос *request* и определяет тип пользователя на основе переданного параметра 'userType'.
- В зависимости от типа пользователя вызывается соответствующий метод для создания пользователя.

createNewUserStudent(Request *request*) :

- Создает нового студента.
- Проверяет валидность данных формы (уникальный *username*, минимальная длина *password*, существование *subjectId*).

- Создает учителя и связывает его с пользователем.

createNewUserTeacher(Request *request*) :

- Создает нового учителя.
- Проверяет валидность данных формы (уникальный *username*, минимальная длина *password*, существование *subjectId*).
- Создает учителя и связывает его с пользователем.

createNewUserAdmin(Request *request*) :

- Создает нового админа.
- Создает пользователя с указанным *username* и хэшированным *password*.
- Создает админа и связывает его с пользователем.

```
public function createLessons(Request $request)
{
    //
    $request->validate([
        'lesson_date' => 'required|date',
    ]);

    //
    $lessonDate = $request->input('lesson_date');
    $errors = [
        'common' => [],
    ];

    //                                          9
    for ($i = 1; $i <= 9; $i++) {
        //
        $validator = \Validator::make($request->all(), $this->
    makeValidationRules($i, $request));

        //
        $fullName = $request->input("teacher_name_$i");
        $classFull = $request->input("class_name_$i");

        //
                                explode
        if (!empty($fullName) && strpos($fullName, ' ') !== false) {
            list($firstName, $lastName) = explode(' ', $fullName, 2);

            //

            $teacher = Teacher::where('name', $firstName)
```

```
                            ->where('Surname', $lastName)
                            ->first();

            if (!$teacher) {
                $errors['common'][] = "Teacher not found for row $i";
            }
        } elseif (!empty($fullName)) {
            $errors['common'][] = "Invalid format for teacher name in row $i";
        }

        //                              explode
        if (!empty($classFull) && strpos($classFull, ' ') !== false) {
            list($classGrade, $className) = explode(' ', $classFull, 2);

            //
            $class = ClassTable::where('grade', $classGrade)
                            ->where('ClassName', $className)
                            ->first();

            if (!$class) {
                $errors['common'][] = "Class not found for row $i";
            }
        } elseif (!empty($classFull)) {
            $errors['common'][] = "Invalid format for class name in row $i";
        }
    }

    //                                      ,
    if (empty($errors['common'])) {
        for ($i = 1; $i <= 9; $i++) {
            //
            $fullName = $request->input("teacher_name_$i");
            if (!empty($fullName)) {
                //
                $lesson = Lesson::updateOrCreate(
                    [
                        'LessonDate' => $lessonDate,
                        'LessonNumber' => $i,
                    ],
                    [
                        'TeacherId' => $teacher->id,
                        'classId' => $class->id,
                        'classroom' => $request->input("classroom_$i"),
                    ]
                );

```

```
                //                                                  4
                for ($week = 0; $week < 4; $week++) {
                    $lesson->replicate()->update([
                        'LessonDate' => \Carbon\Carbon::parse($lessonDate)
    ->addWeeks($week + 1),
                        'LessonNumber' => $i,
                    ]);
                }
            }
        }

        return redirect()->back();
    }

    //                              ,

    return redirect()->back()->withErrors($errors)->withInput();
}
```

createLessons(Request $request):

- **Описание:** Метод создания уроков.
- **Параметры:** $request - HTTP-запрос.
- **Действия:**
  1. Валидирует дату уроков.
  2. Инициализирует переменные, включая массив ошибок.
  3. Проверяет валидность данных для каждого из 9 уроков, используя метод makeValidationRules.
  4. Проверяет наличие преподавателя и класса для каждого урока.
  5. Если обнаружены ошибки, возвращает страницу с ошибками.
  6. Если ошибок нет, создает уроки и их расписание на 4 недели.
  7. В случае успеха, перенаправляет на предыдущую страницу.

```
public function getLessonsByDate(Request $request)
{
    $currentUser = Auth::user();
    $lessons = null;

    if($request)
    {
        $date = $request->input('lesson_date');
    }
    else
    {
```

```
12          $date = now()->toDateString();
13      }
14
15      if ($currentUser->UserType === 'student') {
16          $student = Student::where('Id', $currentUser->UserId)->first();
17          if ($student) {
18              $lessons = Lesson::where('classId', $student->ClassId)
19                  ->where('LessonDate', $date)
20                  ->orderBy('LessonNumber')
21                  ->get();
22                  return view('studentLessons', ['lessons' => $lessons,  '
    today' => now()->toDateString(),'selectedDate' => $date, 'userType' =>
    $currentUser->UserType]);
23          }
24      } elseif ($currentUser->UserType === 'teacher') {
25          $teacher = Teacher::where('Id', $currentUser->UserId)->first();
26
27          if ($teacher) {
28              $lessons = Lesson::where('TeacherId', $currentUser->UserId)
29                  ->where('LessonDate', $date)
30                  ->orderBy('LessonNumber')
31                  ->with('task')
32                  ->get();
33
34              return view('teacherLessons', ['lessons' => $lessons, 'today'
    => now()->toDateString(), 'selectedDate' => $date, 'userType' =>
    $currentUser->UserType]);
35          }
36      }
37      elseif ($currentUser->UserType === 'admin') {
38          $lessons = Lesson::where('LessonDate', $date)
39              ->orderBy('classId')
40              ->orderBy('LessonNumber')
41              ->with('task')
42              ->get();
43
44          return view('adminLessons', ['lessons' => $lessons, 'today' => now
    ()->toDateString(), 'selectedDate' => $date, 'userType' => $currentUser
    ->UserType]);
45      }
46      return redirect()->back();
47  }
```

getLessonsByDate(Request $request):

- **Описание:** Метод получения уроков по указанной дате.
- **Параметры:** $request - HTTP-запрос.
- **Действия:**

1. Получает текущего пользователя из системы аутентификации.
2. Инициализирует переменные, включая дату уроков.
3. Если передан параметр 'lesson$_date'$, , .(, ,) :
4. 5. Если пользователь - студент:
   - Получает информацию о студенте из базы данных.
   - Получает уроки для класса студента на указанную дату.
   - Возвращает представление 'studentLessons' с уроками и дополнительной информацией.

   6. Если пользователь - учитель:
   - Получает информацию о преподавателе из базы данных.
   - Получает уроки, проведенные преподавателем, на указанную дату.
   - Возвращает представление 'teacherLessons' с уроками и дополнительной информацией.

   7. Если пользователь - администратор:
   - Получает все уроки на указанную дату.
   - Возвращает представление 'adminLessons' с уроками и дополнительной информацией.

- В случае ошибки, перенаправляет на предыдущую страницу.

```
public function store(Request $request)
{
    $lessonId = $request->lessonId;
    $studentId = $request->studentId;
    $task = TaskModel::where('LessonID', $lessonId)->first();
    $existingMark = Mark::where('TaskId', $task->id)
                        ->where('StudentId', $studentId)
                        ->first();
    $SolTask = SolutionTaskModel::where('TaskId', $task->id)
                        ->where('StudentId', $studentId)->first();

    $SolTask->verified = 1;

    $SolTask->save();
    if ($existingMark) {

        return redirect()->back()->withErrors([
            'message' => '
                                    .',
        ]);
    }

    $mark = new Mark();
    $mark->MarkNumber = $request->mark;
```

```
24      $mark->MarkDate = now()->toDateString();
25      $mark->TaskId = $task->id;
26      $mark->StudentId = $studentId;
27      $mark->save();
28
29      return redirect()->back()->with('success', '
                          .');
30  }
```

Функционал

store(Request *request*) :

- Получает идентификатор урока (*lessonId*) и идентификатор студента (*studentId*) из запроса.
- Извлекает задание для урока из модели *TaskModel* на основе *lessonId*.
- Проверяет, существует ли уже оценка для данного задания и студента в модели *Mark*.
- Извлекает запись о решении задания для данного задания и студента из модели *SolutionTaskModel*.
- Устанавливает флаг *verified* в 1 для решения задания в *SolutionTaskModel*.
- Если оценка уже существует, возвращает ошибку с сообщением "Оценка за это задание уже была поставлена."
- Создает новую запись об оценке (*Mark*) для задания, устанавливает номер оценки, текущую дату, идентификатор задания и идентификатор студента.
- Возвращает успешное перенаправление назад с сообщением "Оценка успешно добавлена."

```
1  public function login(Request $request)
2      {
3
4          $credentials = $request->only('username', 'password');
5
6          if (Auth::attempt($credentials)) {
7              return redirect()->intended('/');
8          }
9
10         return back()->withErrors([
11             'message' => 'Invalid credentials',
12         ])->withInput($request->except('password'));
13     }
```

Функциональность

login(Request *request*) :

- Получает учетные данные пользователя из запроса (*username* и *password*).
- Использует фасад *Auth* для попытки аутентификации пользователя с предоставленными учетными данными.
- Если аутентификация проходит успешно, перенаправляет пользователя на запрошенный ранее маршрут или на главную страницу.
- В случае неудачной аутентификации возвращает пользователя на предыдущую страницу с ошибкой "Invalid credentials"и включает вводимые ранее учетные данные за исключением пароля.

```
public function uploadTaskFile(Request $request)
       {
           if ($request->hasFile('taskFile')) {
               $currentUser = Auth::user();
               $teacher = Teacher::where('Id', $currentUser->UserId)->
   first();
               $file = $request->file('taskFile');
               $fileName = time() . '_' . $file->getClientOriginalName();
               $filePath = $file->storeAs('task_files', $fileName);

               $taskCheck = TaskModel::where('lessonId', $request->
   lessonId)->first();
               if($taskCheck)
               {
                   $taskCheck->taskFilePath = $filePath;
                   $task->save();
                   return redirect()->back();
               }
               else
               {
                   $task = new TaskModel();
                   $task->lessonId = $request->lessonId;
                   $task->subjectId = $teacher->SubjectID;
                   $task->classId = $request->classId;
                   $task->taskFilePath = $filePath;
                   $task->deadline = $request->deadline;
                   $task->save();

                   return redirect()->back();
               }
           }
           else {
               return redirect()->back();
           }
       }
```

Функциональность

uploadTaskFile(Request *request*) :

- Проверяет наличие загруженного файла в запросе.
- Получает текущего пользователя из системы аутентификации.
- Ищет учителя с использованием идентификатора пользователя текущего пользователя.
- Если файл присутствует, генерирует уникальное имя файла, сохраняет его и получает путь к сохраненному файлу.
- Проверяет наличие задания для урока с использованием идентификатора урока в запросе.
- Если задание уже существует, обновляет путь к файлу в существующем задании и сохраняет изменения.
- В противном случае, создает новое задание, устанавливает необходимые параметры и сохраняет его.
- После завершения операции перенаправляет пользователя на предыдущую страницу.
- Если файл не был загружен, также перенаправляет пользователя на предыдущую страницу.

```
1
2  public function uploadSolutionFile(Request $request)
3          {
4              if ($request->hasFile('file')) {
5                  $file = $request->file('file');
6                  $fileName = time() . '_' . $file->getClientOriginalName();
7                  $filePath = $file->storeAs('solution_files', $fileName);
8                  $taskId = $request->taskId;
9                  $solTaskCheck = SolutionTaskModel::where('TaskId', $request
       ->taskId)->where('StudentId', Auth::user()->UserId)->first();
10                 if($solTaskCheck)
11                 {
12                     $solTaskCheck->SolutionFilePath = $filePath;
13                     $solTaskCheck->save();
14                     return redirect()->back();
15                 }
16                 else
17                 {
18
19
20                     $task = new SolutionTaskModel();
21                     $task->TaskId = $taskId;
22                     $task->SolutionfilePath = $filePath;
23                     $task->StudentId = Auth::user()->UserId;
24                     $task->downloaded = true;
```

```
                    $task->save();

                    return redirect()->back();
                }

            } else {
                return redirect()->back();
            }
        }
```

uploadSolutionFile(Request *request*):

- Проверяет, содержит ли запрос файл под ключом 'file'.
- Если файл существует, получает объект файла, генерирует новое имя файла с помощью текущего временного штампа и оригинального имени файла и хранит файл в каталоге 'solution$_files'., ID, .

- Если решение существует, обновляет путь к решению в базе данных и перенаправляет пользователя на предыдущую страницу.
- Если решение не существует, создает новую запись задачи решения с ID задачи, путем к файлу, ID текущего авторизованного пользователя и полем 'downloaded' установленным в 'true'.
- Если запрос не содержит файл, перенаправляет пользователя на предыдущую страницу.

```
public function downloadSolutionFile(Request $request)
        {
            $studId = $request->StudentId;
            $lessonId = $request->lessonId;

            $task = TaskModel::where('lessonId', $lessonId)->first();
            $SolTask = SolutionTaskModel::where('StudentId', $studId)
            ->where('TaskId', $task->id)
            ->first();

            if (!$SolTask) {
                return redirect()->back()->withErrors(['message' => '
    Solution file not found.']);
            }
            $filePath = $SolTask->SolutionFilePath;
            $file = storage_path('app/' . $filePath);

            if (!file_exists($file)) {
                return redirect()->back()->withErrors(['message' => 'File
    not found.']);
            }

            return response()->download($file);
        }
```

downloadSolutionFile(Request *request*) :

- Извлекает идентификатор студента и урок из запроса.
- Находит связанное с указанным уроком задание, используя модель задания.
- Проверяет наличие записи о решении задания для текущего задания и указанного студента.
- Если запись о решении задания не найдена, возвращает сообщение об ошибке и перенаправляет пользователя на предыдущую страницу.
- Если запись о решении задания найдена, извлекает путь к файлу решения из записи о решении задания.
- Получает файл решения, используя путь к файлу и проверяет, существует ли файл.
- Если файл не существует, возвращает сообщение об ошибке и перенаправляет пользователя на предыдущую страницу.
- Если файл существует, отправляет файл в качестве загружаемого ответа.

```
1
2  public function showChatPage($teacherId = 0)
3      {
4          $curUser = Auth::user();
5          $userChats = Chat::where('abonent_1', $curUser->id)
6              ->orWhere('abonent_2', $curUser->id)
7              ->get();
8
9          if($teacherId == 0)
10         {
11             $selectedChat = $userChats->first();
12             if($selectedChat->abonent_1 == $curUser->id)
13             {
14                 $recepient = User::find($selectedChat->abonent_2);
15             }
16             else
17             {
18                 $recepient = User::find($selectedChat->abonent_1);
19             }
20         }
21         else
22         {
23             //                selectedChat       $teacherId
24             $TeachUser = User::where('UserId', $teacherId)->first();
25             $TeachUserId = $TeachUser->id;
26
27             $recepient = $TeachUser;
28             $selectedChat = Chat::where(function ($query) use ($curUser,
       $TeachUserId) {
```

```
                $query->where('abonent_1', $curUser->id)
                    ->where('abonent_2', $TeachUserId);
            })->orWhere(function ($query) use ($curUser, $TeachUserId) {
                $query->where('abonent_1', $TeachUserId)
                    ->where('abonent_2', $curUser->id);
            })->first();

        }
        if($curUser->UserType == 'teacher') {
            $userClass = Teacher::where('Id', $curUser->UserId)->first();
            $teachers = $userClass->getStudents();
        }
        if($curUser->UserType == 'student') {
            $userClassId = Student::where('Id', $curUser->UserId)->first()
->ClassId;
            $teachers = Teacher::whereHas('lessons', function ($query) use
($userClassId) {
                $query->where('classId', $userClassId);
            })->get();
        }
        $allChats = Chat::where('abonent_1', $curUser->id)->get();

        if($allChats->count() > 0)
        {
            $teacherIds = $allChats->pluck('abonent_2')->unique();
            if($selectedChat)
            {
                $messages = Message::where('chat_id', $selectedChat->id)->
get();
            }
            else
            {

                $messages = [];
            }
            $userTempIds = User::whereIn('id', $teacherIds);
            if($curUser->UserType == 'teacher')
            {
                $teachersChange = Student::whereIn('Id', $userTempIds->
pluck('UserId'))->get();

            }
            else if($curUser->UserType == 'student')
            {
                $teachersChange = Teacher::whereIn('Id', $userTempIds->
pluck('UserId'))->get();
            }
        }
        else
        {
```

```
74              $allChats = Chat::where('abonent_2', $curUser->id)->get();
75              $teacherIds = $allChats->pluck('abonent_1')->unique();
76              $messages = Message::where('chat_id', $selectedChat->id)->get()
    ;
77              $userTempIds = User::whereIn('id', $teacherIds);
78              if($curUser->UserType == 'teacher')
79              {
80                  $teachersChange = Student::whereIn('Id', $userTempIds->
    pluck('UserId'))->get();
81
82              }
83              else if($curUser->UserType == 'student')
84              {
85                  $teachersChange = Teacher::whereIn('Id', $userTempIds->
    pluck('UserId'))->get();
86              }
87
88          }
89          //dd($selectedChat->id);
90          return view('chat', [
91              'recepient' => $recepient,
92              'teachersChange' => $teachersChange,
93              'currentUser' => $curUser,
94              'userType' => $curUser->UserType,
95              'teachers' => $teachers,
96              'userChats' => $userChats,
97              'selectedChat' => $selectedChat,
98              'messages' => $messages,
99          ]);
100     }
```

showChatPage($teacherId = 0):

- Получает текущего пользователя из авторизации.
- Извлекает чаты пользователя, где он участвует как абонент 1 или абонент 2.
- Если указан $teacherId(), .().

- Если не указан $teacherId, ., − , , .
- В случае, если текущий пользователь - студент, извлекает учителей, связанных с ним через уроки.
- Извлекает все активные чаты пользователя.
- Если есть чаты, извлекает их уникальные идентификаторы абонентов и соответствующих собеседников.
- Определяет выбранный чат и извлекает сообщения для него.
- Определяет участников чата на основе роли текущего пользователя (учитель или студент).

Список литературы

[Bea15] Martin Bean. *Laravel 5 Essentials*. Packt Publishing, 2015.

[Kı14] Arda Kılıçdağı. *Laravel Design Patterns and Best Practices*. Packt Publishing, 2014.

[Otw16] Taylor Otwell. *Laravel: Up and Running*. O'Reilly Media, 2016.

[Pec17] Christopher John Pecoraro. *Learning Laravel's Eloquent*. Leanpub, 2017.

[Wat17] Adam Wathan. *Test-Driven Laravel: Test-Driven Development with Laravel*. Leanpub, 2017.

В руководстве [Otw16] приведены основы работы с Laravel.
Дизайн-паттерны и лучшие практики в Laravel описаны в книге [Kı14].
Книга [Pec17] предоставляет материал по изучению Eloquent в Laravel.
В Essentials Laravel 5 [Bea15] вы найдете ключевые аспекты разработки на Laravel 5.
В книге [Wat17] автор Adam Wathan представляет Test-Driven Development с использованием Laravel.