|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **\_\_\_\_*ИУ-КФ «Информатика и управление»*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

КАФЕДРА ***ИУ4-КФ «Программное обеспечение ЭВМ, информационные технологии»*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

***\_Разработка корпоративного новостного портала с возможностью просмотра git-репозиториев\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_ИТД.Б-82\_\_ **\_\_\_\_\_\_\_\_\_\_05.06.2020\_\_И.С. Голубев\_\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР **\_\_\_\_\_\_\_\_\_\_05.06.2020\_\_\_К.А. Амеличева\_\_**

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **\_\_\_\_\_\_\_\_\_\_\_05.06.2020\_\_С.С. Гришунов\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2020 г.*

**РЕФЕРАТ**

Расчётно-пояснительная записка 47 с., 24 рис., 4 источника. Разработка цифрового хранилища портфолио.

Предметной областью являются студенты, соискатели , работники hr, процесс документооборота портфолио.

Целью работы является разработка автоматизированного цифрового хранилища для документов связанных с портфолио.

Поставленные задачи решаются путем проектирования и разработки веб-сайта, доступ к которому будет разграничен ролью пользователей и работодателей.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 7](#_Toc74137590)

[1. МЕТОДЫ И ИНСТРУМЕНТЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ 8](#_Toc74137591)

[*1.1.* Основные требования 8](#_Toc74137592)

[*1.2.* Основания для разработки 9](#_Toc74137593)

[*1.3.* Анализ существующих аналогов 10](#_Toc74137594)

[*1.4.* Обоснование выбора языка программирования серверной части 10](#_Toc74137595)

[*1.5.* Обоснование выбора фреймворка 14](#_Toc74137596)

[*1.6.* Обоснование выбора СУБД 19](#_Toc74137597)

[2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА 23](#_Toc74137598)

[*2.1.* Общие сведения 23](#_Toc74137599)

[*2.2.* Проектирование функциональной модели 23](#_Toc74137600)

[*2.3.* Описание работы контроллера 25](#_Toc74137601)

[*2.4.* Описание компонента Клиент 29](#_Toc74137602)

[*2.5.* Описание Работы с Моделями и базой данных 30](#_Toc74137603)

[3. ТЕСТИРОВАНИЕ И ИНТЕГРАЦИЯ КОМПОНЕНТОВ ПРОГРАММНОГО ИЗДЕЛИЯ 32](#_Toc74137604)

[3.1. Разработка тестов и результаты тестирования 32](#_Toc74137605)

[3.2. Тестирование основных пользовательских сценариев 32](#_Toc74137606)

[**Руководство администратора** 37](#_Toc74137607)

## ВВЕДЕНИЕ

Актуальность темы выпускной квалификационной работы обусловлена тем, что Одной из главных проблем современного общества является бумажный документооборот. Огромные архивы, ненадежность носителя, неудобство передачи - все это затрудняет удобную работу с документами. Существующие аналоги в Российском IT сегменте не предоставляют необходимые инструменты.

Основанием для разработки данной системы является возможность избавить документооборот портфолио от минусов бумажного носителя, путем создания облачного централизованного хранилища.

Объектом выпускной квалификационной работы является веб-приложение, в котором back-end реализован на PHP, а front-end на JavaScript.

Предметом исследования выпускной квалификационной работы является возможность создания полноценного приложения, с реализованной логикой Back-end и Front-end.

Целью выпускной квалификационной работы является проектирование и разработка веб-приложения, состоящего из back-end и front-end части.

Для достижения поставленной цели решаются следующие задачи:

1. Изучение аналогов на Российском IT сегменте.
2. Изучение языка JavaScript, его библиотек и фреймворков.
3. Изучение языка PHP.
4. Изучение Apache2, ознакомление с его возможностями и применение в данном проекте.
5. Изучение фреймворка Laravel, ознакомление с паттерном MVC.
6. Проектирование компонент комплекса приложений в ERwin для наглядной демонстрации строения.
7. Программная реализация и тестирование программного продукта.

## МЕТОДЫ И ИНСТРУМЕНТЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

### Основные требования

**Требования к функциональным характеристикам приложения**

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* пользователь должен иметь возможность регистрации и авторизации в приложении;
* разрабатываемая система должна работать в мобильной версии браузера;
* при неверном вводе логина или пароля пользователь должен видеть сообщение об ошибке входа в приложение;
* разрабатываемое приложение должно соответствовать принципу расширяемости, т.е. иметь возможность наращивания своей функциональности, добавления новых функций;
* пользователь должен иметь возможность загружать новые и удалять уже загруженные документы;
* разрабатываемое приложение должно иметь удобный интерфейс для взаимодействия с пользователем, быть клиент-ориентированным и интуитивно понятным в использовании;

**Требования к защите информации и программ**

Разрабатываемое приложение должно быть безопасным и гарантировать пользователю защиту его данных, а также обеспечить полную конфиденциальность непубличной информации пользователей.

**Климатические условия эксплуатации**

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации [2].

**Требования к физическому серверу**

* Процессор: двухъядерный с тактовой частотой 2.8 ГГц или лучше;
* Оперативная память: не менее 4 Гб;
* Свободное место на жестком диске: не менее 1 Гб с возможностью расширения;

**Требования к программному обеспечению сервера**

* Операционная система: amd64 дистрибутив Linux.
* Необходимое ПО: PHP,APACHE2, MySQL-server.
* Дополнительные требования: Docker: версия 1.10 или выше;

**Требования к программному обеспечению клиента**

* Любой современный браузер с поддержкой работы JS кода.

**Требования к квалификации и численности персонала**

Система требует наличие двух категорий пользователей: работодатели , осуществляющие поиск соискателей и соискатели, создающих свои портфолио.

### Основания для разработки

В настоящее время многие аспекты нашей жизни переходят в цифровое пространство, цифровизируются. Общение перешло в интернет, аналогично и работа. В бизнес сфере происходит огромный документооборот. Важные договора, личные дела и другие документы проходят через огромной количество рук. Бумажные носители имеют недостатки: способны потеряться, испортиться, их трудно отредактировать и т.д. Многие уже сейчас используют цифровые подписи для цифровых документов, чтобы облегчить работу с важными бумагами. В рамках данной работы разрабатывается система для хранения, обмена и редактирования документов, относящихся к портфолио, а так же дополнительные сервисы для удобной работы с этой системой.

### Анализ существующих аналогов

В настоящее время на просторах интернета существует множество различных веб-приложений, которые предоставляют подобный, но не полный функционал, например:HeadHuter.ru(рис. 1.1), Сareer.habr.com и т.д. Среди них есть и сайты с приоритетным графическим интерфейсом, и с приоритетным текстовым. Исходя из недостатков, а также достоинств такого рода веб- приложений, было принято создать такой проект, который бы собрал в себе лучшее и дополнен новыми функциями. Поэтому разрабатываемый сайт будет иметь современный графический интерфейс, создание портфолио и резюме по евро стандарту, быстрый и удобный доступ к данным портфолио, а так же возможность встраивания виджетов в другие веб-сайты.

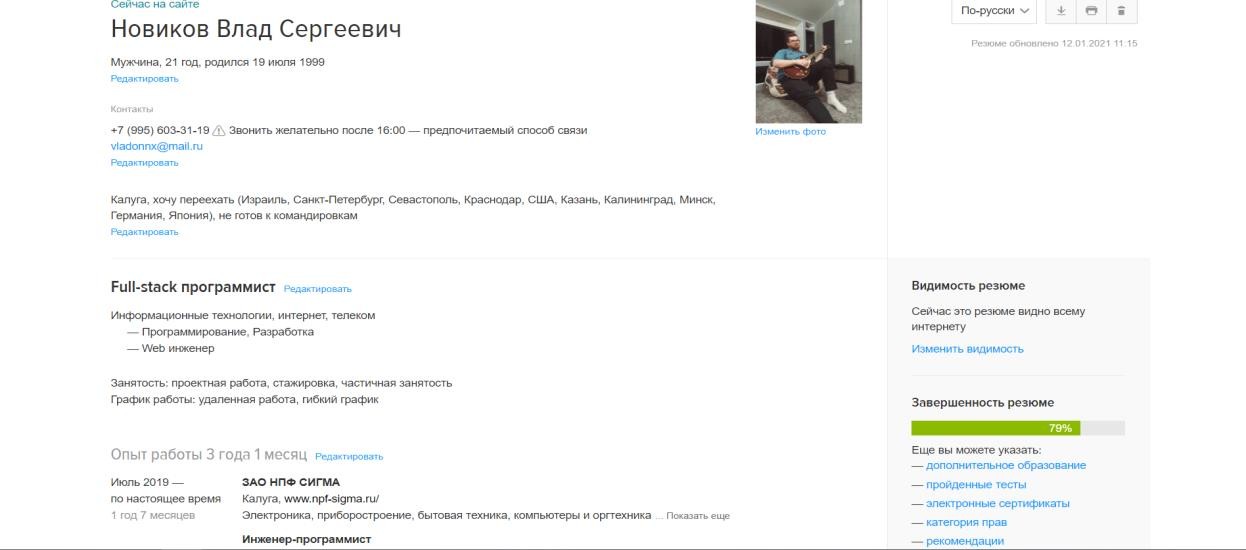


Рис. 1.1 Резюме на HeadHunte

### Обоснование выбора языка программирования серверной части

**С и C++**

С++ и С — строго типизированные объектно-ориентированные языки программирования. Они поддерживает также такие парадигмы программирования, как процедурное, функциональное и обобщённое. Стоит отметить, что C++ унаследовал классические массивы от языка Си, в котором отсутствуют проверки при индексации массива при передаче, например, в процедуру. В С++ существует много возможностей, нарушающих принципы типобезопасности, что приводит к труднонаходимым ошибкам.

Плохая поддержка модульности в С++ (в классическом Си модульность на уровне языка отсутствует, её обеспечение возложено на компоновщик). Подключение интерфейса внешнего модуля через препроцессорную вставку заголовочного файла (#include) серьёзно замедляет компиляцию при подключении большого количества модулей.

Нередко С++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит, антивирусов. Указатели или адреса в памяти – одна из мощнейших черт данных языков.

Преимущества:

* высокая скорость работы;
* наличие указателей.

К недостатку использования языков С++ и С для разработки данного проекта можно отнести отсутствие фреймворка для разработки веб- приложения.

**PHP**

PHP (рекурсивный акроним словосочетания PHP: Hypertext Preprocessor) - это распространенный язык программирования общего назначения с открытым исходным кодом. PHP специально сконструирован для веб-разработок и его код может внедряться непосредственно в HTML. Кроме того, PHP крайне прост для освоения.

Имеет ряд преимуществ:

* Низкий порог входа. PHP очень простой язык программирования, которому можно научиться в максимально короткие сроки;
* Скрипты выполняются на сервере;
* Код внедряется в HTML. Клиент не сможет понять, получили ли они HTML файл или результат скрипта;
* Структурированная и полная документация с большим количеством примеров;

А также имеет ряд недостатков:

* Многопоточность. Один процесс-один запрос, а вызовы ввода- вывода блокируют выполнение скрипта. Не каждый сервер выдержит большое количество параллельных запросов;
* Сложная масштабируемость. Большой проект сложно изменить или ещё расширить. Новые версии языка перестают поддерживать обратную совместимость.

**Golang (GO)**

Golang – компилируемый многопоточный язык программирования, разработанный внутри компании Google, является одни из перспективных языков программирования. Go разрабатывался как язык программирования для создания высокоэффективных программ, работающих на современных распределённых системах и многоядерных процессорах.

Golang – объектно-ориентированный язык программирования, применяющий фундаментальные принципы ООП, но использует несколько иные механизмы в отличие от других языков программирования.

Имеет ряд преимуществ:

* Строгая статическая типизация;
* Средства параллельного программирования: встроенные в язык потоки (go routines), взаимодействие потоков через каналы и другие средства организации многопоточных программ;
* Структурированная и полная документация. Она находится в сети Интернет на онлайн ресурсе https://godoc.org/. Это один из языков программирования, чья документация собрана в одном месте официально;
* Средства параллельного программирования: встроенные в язык потоки (goroutines), взаимодействие потоков через каналы и другие средства организации многопоточных программ;
* Крайне высокая скорость компиляции и работы программ;
* Удобная система обработки ошибок, предусмотренная на уровне языка.

А также имеет ряд недостатков:

* Непопулярность. Язык разработан в 2007 году и не успел войти в широкое применение, поэтому кроме официальной документации сложно найти примеры сложных программ;
* Жесткое форматирование кода;
* Все пакеты во время компоновки встраиваются в бинарный файл, размер которого стремительно растет.

**JavaScript**

JavaScript - это мультипарадигменный язык программирования, обычно используемый в качестве встроенного инструмента для программного доступа к различным объектам приложения. С точки зрения веб-разработки невозможно создать современный интерактивный сайт без этой технологии. Язык JS предоставляет эффекты анимации для разметки страницы (HTML) и пользовательских функций (CMS) веб-сайта. С помощью этого языка можно реагировать на поведение посетителя на странице или ее различных элементах. JavaScript - это основной язык программирования современных браузеров. Он полностью совместим с операционными системами Windows, Linux, Mac OS и всеми популярными мобильными платформами. Преимуществами этого языка можно считать:

* Все современный системы и браузеры поддерживают JavaScript;
* Даже если вы не являетесь экспертом, вы можете работать с библиотекам и скриптами, написанными на JavaScript;
* Полезные и удобные функциональные настройки;
* Язык, который постоянно совершенствуется;
* Вы даже можете взаимодействовать с приложениями через текстовые редакторы (Microsoft Office и Open Office).
* Перспектива применения языка в процессе обучения программированию и информатике.

Таким образом, проанализировав преимущества и недостатки, представленных выше языков, самым удобным и подходящим для back-end части проекта будет PHP. Т.к. достаточно экономичен и прост, подходит для создания ИТ-проектов любого типа, в том числе и для информационного портала. Он совместим со всеми основными платформами: от Windows, Unix и Linux до MacOS; поддерживает большинство серверов (включая Apache, Microsoft IIS, Netscape, iPlanet, Caudium, Xitami и Tornado) и более 20 баз данных (среди которых — MySQL, MongoDB, PostgreSQL и др.). Что позволяет привлечь больше пользователей и разработчиков к данному проекту. А для front-end был выбран язык JavaScript, так как является простым и удобный в разработке.

### Обоснование выбора фреймворка

В качестве инструментов для реализации данного курсового проекта были рассмотрены такие фреймворки, как Laravel, Symfony, Yii2, Codeigniter. Именно они способны дать максимально эффективный набор инструментов

для разработки данного проекта.

**Laravel**

Laravel − это бесплатный PHP фреймворк с открытым исходным кодом, созданный Тейлором Ответом для разработки веб-приложений по архитектурному шаблону MVC. Является своеобразным каркасом для разработки веб-проектов, независимо от объема и сложности, и состоящих из разнообразных элементов.

Он был создан как альтернатива фреймворку Codeigniter (о нем будет рассказано ниже), в котором было недостаточно полезных функций для разработки веб-приложений. В качестве основы Laravel выступают компоненты другого фреймворка - Symfony (о нем также, будет рассказано ниже).

С помощью менеджера пакетов Composer, фреймворк Laravel позволяет легко устанавливать и подключать различные компоненты для использования в веб-приложении.

В Laravel есть встроенная поддержка движка шаблонов Blade, с помощью которого можно делать простые представления веб-приложения используя специальный синтаксис.

Плюсы:

Гибкая система маршрутизации, позволяющая составить самые разные проверки маршрута веб-приложения.

В синтаксис API фреймворка нет длинных и сложных конструкций, а только краткие и продуманные названия функций.

Laravel содержит удобный механизм обработки ошибок и исключений.

Фреймворк включает в себя встроенные механизмы аутентификации и авторизации пользователей, которую можно перенастроить под свои потребности. Laravel предоставляет из коробки механизмы для кэширования веб- приложения с помощью Memcached и Redis. Кроме этого есть удобные функции для использования простого файлового кэширования данных.

Laravel предоставляет чистый и простой API поверх популярной библиотеки SwiftMailer с драйверами для SMTP, Mailgun, SparkPost, Amazon SES и sendmail, чтобы сделать отправку почты через локальную или облачную службу по выбору. В том числе есть механизм для построения очередей отправки почты.

Минусы:

Синтаксический сахар в Laravel как плюс, так может быть и минусом. Легко привыкнуть к нему и забыть, как пишутся чистые запросы и функции.

Не логичное расположение каталогов и файлов. Например, по умолчанию в прямо в каталоге /app расположена модель User.php, которую логичней было бы расположить в каталоге /app/Models. Каталог resources с файлами представления размещен в корне приложения, хотя логичней будет его разместить в /app/resources.

Большой функционал работает через фасады и IDE-системы не видят методов и свойств в некоторых классах, показывая предупреждения.

Мало официальной документации на русском языке

Нет встроенных генераторов интерфейсов

**Codeigniter**

CodeIgniter - это популярный PHP микро-фреймворк с открытым исходным кодом, для разработки веб-систем и приложений. Разработан компанией EllisLab, а также Риком Эллисом и Полом Бурдиком.

В CodeIgniter компоненты загружаются и процедуры выполняются только по запросу, а не глобально. Компоненты фреймворка слабо связаны между собой и не зависят друг от друга. Чем меньше компонентов зависит друг от друга, тем более гибкой и многоразовой становится система.

CodeIgniter работает довольно быстро, но объем динамической информации, отображаемой на страницах, будет напрямую зависеть от используемых ресурсов сервера, памяти и циклов обработки, которые влияют на скорость загрузки страниц. Поэтому CodeIgniter позволяет кэшировать страницы для достижения максимальной производительности. с помощью встроенного компонента кэширования.

Плюсы:

Высокая производительность фреймворка.

Небольшой размер фреймворка.

Предоставляет легкие и простые решения для разработки.

Подходит для быстрой разработки небольших сайтов и веб-приложений.

Структура фреймворка не требует строгих правил кодирования.

Не требует сложной настройки, почти нулевая конфигурация.

MVC-архитектура веб-приложения.

Слабая связанность компонентов.

Множество подключаемых библиотек и помощников.

Минусы:

Сложности с регистрацией и авторизацией

Сложности с разграничением доступа в различных версиях CodeIgniter.

**Symfony**

Symfony - свободный PHP фреймворк для быстрой разработки веб- приложений и решения рутинных задач веб-программистов. Разработка и поддержка фреймворка спонсируется французской компанией Sensio. Symfony состоит из набора не связанных между собой компонентов, которые можно использовать повторно в проектах. В том числе, Symfony повлиял на разработку фреймворка Laravel, где были задействованы его компоненты.

Symfony позволяет устанавливать сторонние пакеты, библиотеки, компоненты и настраивать их с помощью конфигурации в форматах YAML, XML, PHP, а также .env файлах.

Symfony не обеспечивает компонент для работы с базой данных, но обеспечивает тесную интеграцию с библиотекой Doctrine.

Symfony предлагает систему логирования ошибок приложения, а также подключить библиотеку логирования Monolog.

Плюсы:

Встроенная поддержка Codeception позволяет писать функциональные и приёмочные тесты

YAML как плюс фреймворка

Много готовых модулей (бандлов)

Подробная и понятная документация

Достаточно высокая скорость работы ядра

Низкая связность компонентов Минусы:

Symfony использует очень ресурсоемкий ОРМ.

Высокий порог входа

Перенасыщенность разного рода сущностями

Интегрированный аннотационный синтаксис

Таким образом, из перечисленных выше фреймворков наиболее удобным и простым является фреймворк Laravel, т.к. он содержит множество полезных инструментов, которых нет у остальных фреймворков. Например, функции, которые позволяют легко реализовать авторизацию, что предотвращает доступ посторонних лиц к защищенному контент. Полезный набор библиотек Laravel расширяет функциональность фреймворка и позволяет создавать высококачественные пользовательские интерфейсы. Поэтому Laravel и был выбран в качестве фреймворка для реализации данного проекта.

### Обоснование выбора СУБД

Разрабатываемая система предполагает хранение и обработку данных. Наилучшим способом работы с информацией, учитывая эти цели, является использование базы данных. Для взаимодействия с базами данных используются системы управления базами данных (СУБД). Для создания базы данных было необходимо выбрать наиболее подходящую СУБД для поставленной задачи. В силу распространенности, для рассмотрения были выбраны: SQLite, Firebird, MySQL.

SQLite - встраиваемая компактная реляционная база данных. Особенность SQLite в том, что она не использует парадигму клиент-сервер, поэтому её и называют «встраиваемой». Движок SQLite не отдельно работающий процесс, а библиотека, с которой программа компонуется и движок становится составной частью программы.

Имеет ряд преимуществ:

* Файловая структура - вся база данных состоит из одного файла, поэтому её очень легко переносить на разные машины;
* Полностью свободная лицензия;
* Кроссплатформенность;
* Безопасность. БД хранится в одном файле, права доступа к которому можно контролировать стандартными средствами ОС.

А также имеет ряд недостатков:

* Отсутствие системы пользователей - более крупные СУБД включают в свой состав системы управления правами доступа пользователей. Обычно применения этой функции не так критично, так как эта СУБД используется в небольших приложениях;
* Отсутствие возможности увеличения производительности - опять, исходя из проектирования, довольно сложно выжать что-то более производительное из этой СУБД.

FireBird – свободно распространяемая СУБД. Поддерживает стандарты ANSI в синтаксисе языка SQL и позволяет работать на многих операционных системах. Firebird используется в различных промышленных системах (складские и хозяйственные, финансовый и государственный сектора)

Имеет ряд преимуществ: 1. Многоверсионная архитектура (параллельная обработка оперативных и аналитических запросов: читающие пользователи не блокируют пишущих); 2. Соответствие требованиям ACID.

А также имеет ряд недостатков:

* Отсутствие кэша итогов запросов, полнотекстовых индексов;
* Значительное падение производительности при росте внутренней фрагментации базы.

MySQL - одна из самых популярных баз данных для веб-приложений. Она включает в себя бесплатный пакет программ, однако новые версии выходят постоянно, расширяя функционал и улучшая безопасность. В бесплатной версии наибольший упор делается на скорость и надежность, а не на полноту функционала, который может стать и достоинством и недостатком - в зависимости от области внедрения

В силу того, что MySQL является серверной СУБД, приложения для доступа к данным, в отличии от SQLite работают со службами MySQL. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Имеет ряд преимуществ:

* Простота. MySQL легко устанавливается и проста в настройке и использовании;
* Безопасность. Имеет много функций, обеспечивающих безопасность и поддерживаемых по умолчанию;
* Масштабируемость. MySQL легко работает с большими объемами данных и просто масштабируется.
* В качестве СУБД для разработки была выбрана MySQL

## ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА

### Общие сведения

Данный курсовой проект представляет собой веб-приложение для пользователей, желающим хранить свою портфолио в сети интернет, а так же работодателям, которые ищут соискателей по интересующим их навыкам. Веб- приложение содержит страницы портфолио, поисковые страницы, а так же отдельные страницы с файлами. Приложение так же предоставляет возможность создания, редактирования и удаления данных.

### Проектирование функциональной модели

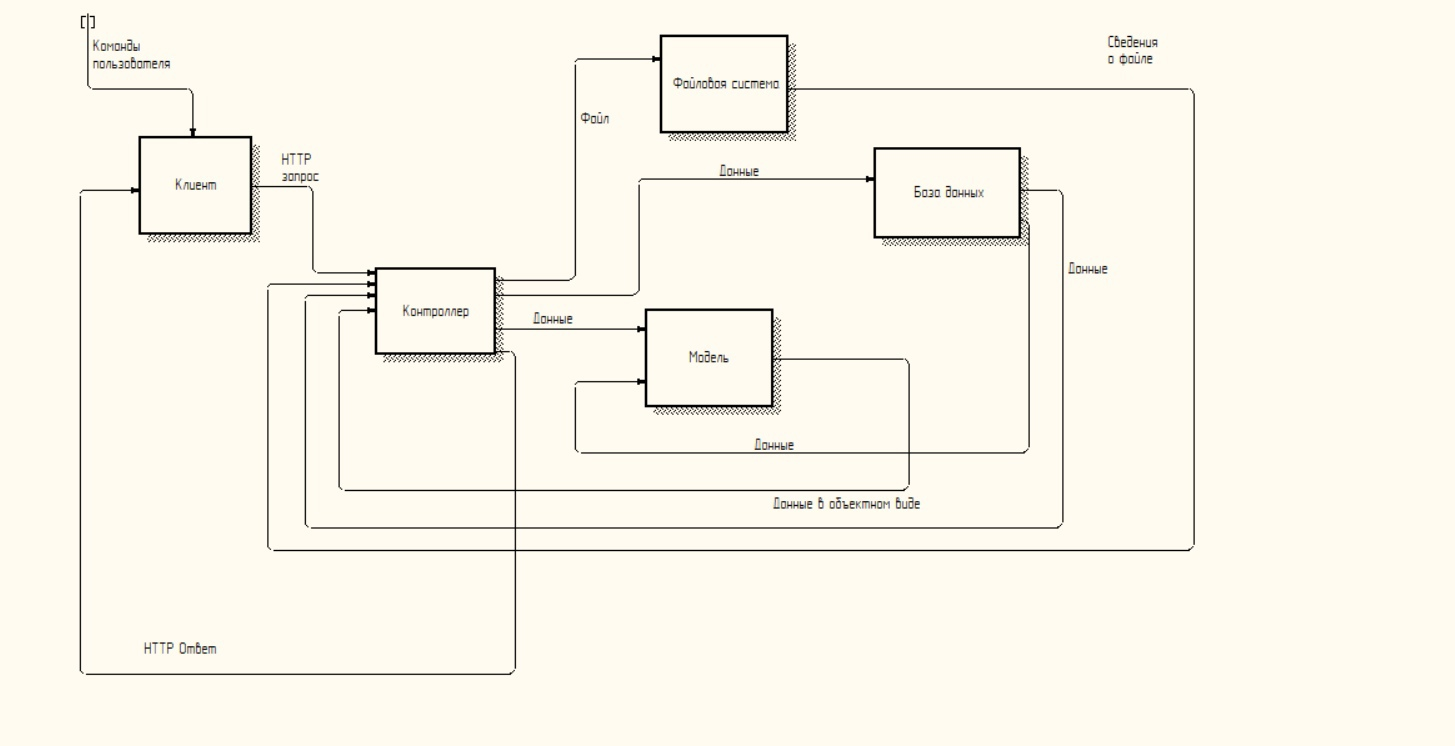
При проектировании была определена модель работы приложения (рис. 2.1)

Рис. 2.1. Модель работы приложения

На данной схеме отчетливо видно, что главным элементом приложения является контроллер. Контроллер регулирует работу всех запросов приходящих на Beck-end. Контроллер связан с Базой данной как напрямую, имея возможность создать запрос или очередь, так и посредством моделей - объектным представлением данных из БД. Модели имеют встроенные функция для работы с базой данных. Через контроллер осуществляется взаимодействие с файловой системой сервера: запись, удаление, извлечение и т.д. Через дополнительный драйвер можно подключить удаленное хранилищ, например драйвер FTP или Amazon S3, система будет считать диски локальными и работать точно так же как с корневой файловой системой. Разрабатываемая система полностью соответствует паттерну MVC.

На следующем рисунке (Рис. 2.2) представлена декомпозиции одного из контроллеров. Из нее можно понять принцип работы приложения. Пользователь проходя через систему аутентификации( В нашем случае Middleware) перенаправляется к панели управления профилем(Визуальная - клиентская часть. Представляет собой HTML страницу отредеренную на сервере). Через HTTP запрос отправляем файл на сервер, где она записывается в Storage( файловую систему). На выходе получаем страницу портфолио, которую можно выгрузить в формате PDF.

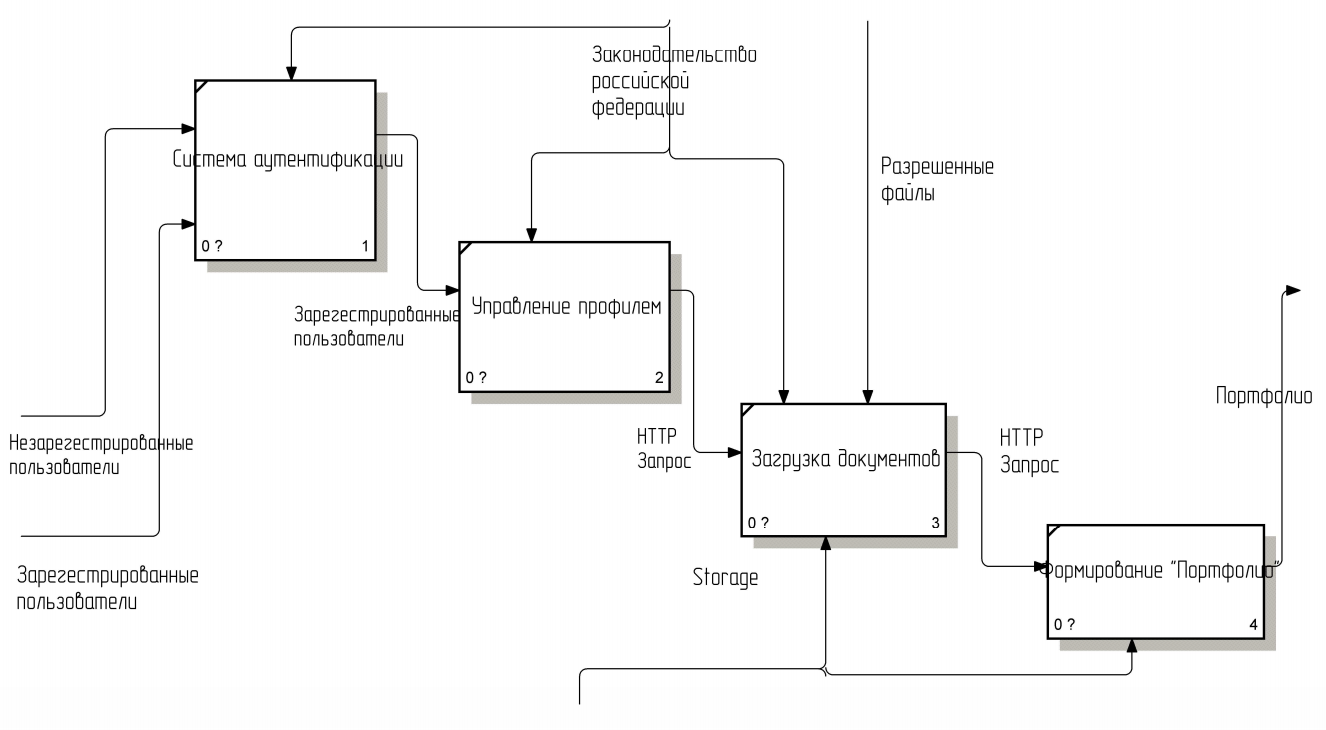


Рис. 2.2 Функциональная модель

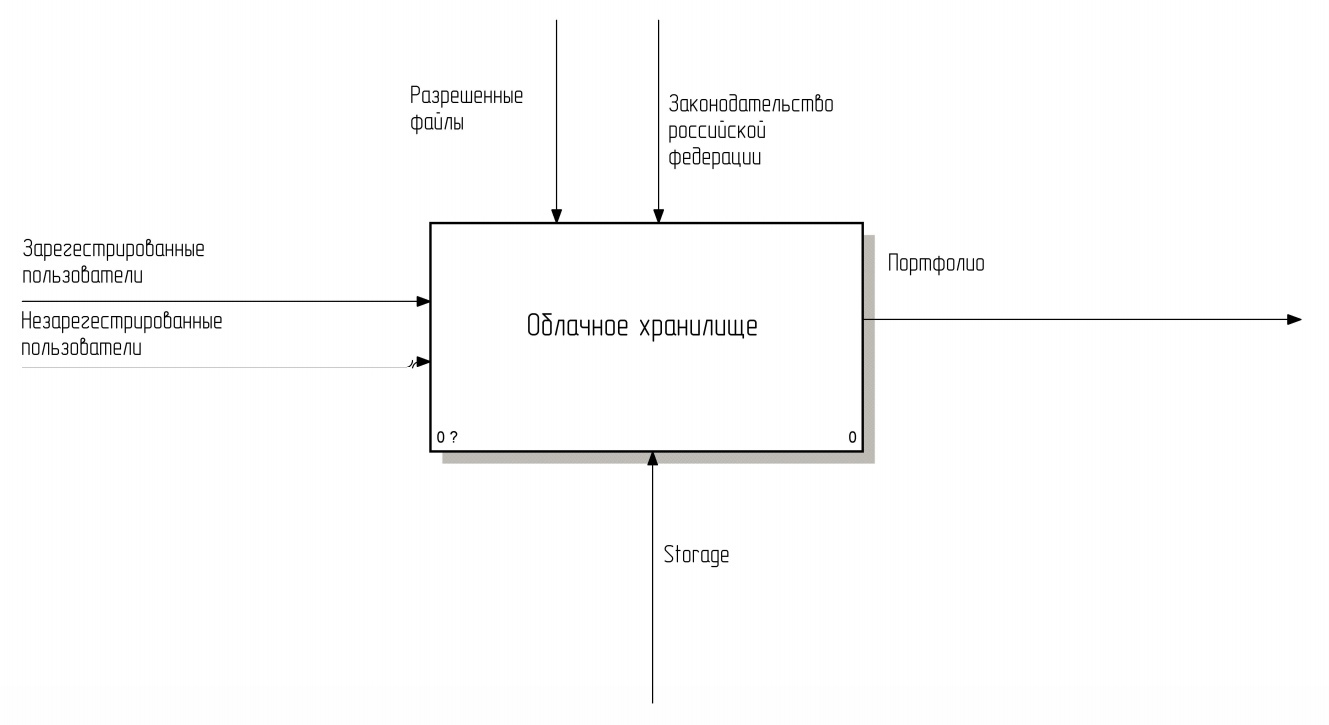


Рис. 2.3 Логическая модель

Разберем по порядку используемые функциональные модули.

**Структура разрабатываемых классов и контроллеров UML Схема**

На схеме(рис. 2.3) ниже находится UML схема разработанного приложение. Из Схемы видно наследование и агрегация некоторых классов(прим. Model,DB,Portfolio). Классы без полей являются встроенным классами и нужны для сборки более комплексных классов. Фреймворк автоматически доконструирует отнаследованные классы.

Классы отнаследованные от Models являются не жестко типизированными и в зависимости от информации меняют поля(При. Obj стал name,phone,about и т.д.). Это конструирование происходит при вызове конструктора исходного класса и вся информация подтягивается из Базы данных.

В классе Web есть всего один метод, в котором перечисляются привязки для вызова ивентов по запросам(прим. Запрос “Get /portfolio” Привязка к вызову метода контроллера PortfolioController -> MainView())

Рис. 2.3 UML

**Класс Routes**

Routes используется для определения действий веб-приложения в зависимости от URL-адреса (См. Блок-схема 1).

1. « / » – возвращает view главной страницы
2. « /dashboard » − возвращает view страницы, содержащую в себе портфолио.
3. « /get/{user}/{id} » − просмотр pdf страницы с портфолио.
4. « /load/{user}/{id} » − загрузка pdf страницы с портфолио.
5. « /search» − страница с полнословным поиском по сайту.

<?php

use App\Http\Controllers\OrijinxPageController; use App\Models\User;

*//Маршрутк, отображающий главную страницу*

Route::any('/', [App\Http\Controllers\MainController::class, 'IndexView']);

Auth::routes();

Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])->name('home');

Route::get('/logout', function () { Auth::logout();

*return* back();

});

Route::middleware(['auth:sanctum', 'verified'])->get('/home', [App\Http\Controllers\PortfolioController::class,'HomeView'])-

>name('dashboard');

*/\**

*/*

*Далее находится блок для работы с портфолио.*

*Все обновление, вывод, удаление и чтение будет осуществляться через следующие руты.*

*Главный Контроллер отвечающий за работу этого блока:*

*PortfolioConroller.php*

*/*

*\*/*Route::get('/dashboard', [App\Http\Controllers\PortfolioController::class,'HomeView']);

Route::post('/UpdatePortfolio',[App\Http\Controllers\PortfolioContr oller::class,'UpdatePortfolio'])->name('upd');

route::get('/user/{id}',function($id){

$data = User::where('id',$id)->first();

*return* view('PDF.export',compact('data'));

});

//Поиск Route::get('/search',function(){

*return* view('search\_result');

});

Route::get('/result', [App\Http\Controllers\PortfolioController::class,'Search'])-

>name('searchSimple');

//// Route::post('/loadfile',[App\Http\Controllers\PortfolioController::

class,'LoadFiles']);

Route::get('get/upload/{email}/{name}',[App\Http\Controllers\Portfo lioController::class,'GetFile']);

Route::get('load/upload/{email}/{name}',[App\Http\Controllers\Portf olioController::class,'DownLoadFile']);

Route::get('/pdf',function(){

*return* App\Http\Controllers\Export::createPDF();

});

**Контроллер PortfolioController**

PortfolioController используется в проекте для отображения страниц и работы с портфолио.

MainView − возвращает view главной страницы. Для получения данных из таблицы Users, используется функция для запроса к БД. “ Users::where( )” формирует сортированный запрос “SELECT”, в котором данные будут отсортированы по текущему пользователю. Обращение к методу “get( )” возвращает коллекцию объектов в виде класса “user”. Функция “View( )” возвращает отрисованную страницу Html. Также через второй параметр можно передать скомпонованную коллекцию параметров передаваемых в страницу переопределенные данные для шаблонизатора), через функцию компоновки “compact( )”. Коллекцию можно скомпоновать самостоятельно, обернув

данные в массив.

*//Функция, которая возвращает вид страницы*

public function MainView()

{

*if*(Auth::check()){

$skills = skill::all();

$user\_skill = skills\_connect::where('user\_id',Auth::user()->id)->get();

$vault\_conn= files\_conn::where('user\_id',Auth::user()-

>id)->orderBy('id','desc')->limit('5')->get();

$vault\_=array();

*foreach*($vault\_conn as $conn){ array\_push($vault\_, vault::where('id',$conn-

>files\_id)->first());

}

*return*

view('ProfilePage.portfolio',compact('skills','user\_skill','vault\_'));

}

*else*{

*return* redirect('/')->with('fail','Вы не аутентифицированы!');

}

}

Для других функций рендера страниц используются аналогичные методы.

* UpdatePortfolio − из request получает данные (POST запрос) обновляет данные в таблице User
* LoadFiles – загружает файл(ы) из request в файловую систему storage и создает новую запись в таблица portfolio\_files и portfolio\_files\_connects.

*//Функция, которая сохраняет пользовательские изменения*

public function LoadFiles(Request $request)

{*f* (Auth::Check()) {

$user = Auth::user();

*///Проверяем блок с основной информацией*

*if* ($request->hasFile('file\_load')) { *пользователя*

$filess=$request>file('fil*reach* ($filess as $fil) {$vault = new vault(); *//Получили текущего*

$vault\_conn = new files\_conn();

$vault->name =$fil->getClientOriginalName();

$vault->file\_path =Storage::putFileAs('upload/'.$user->email, $fil,$vault->name);

$vault->save();

$vault\_conn->user\_id =Auth::user()->id;

$vault\_conn->files\_id =$vault->id;

$vault\_conn->save(); }

*return* back()->with('success','Успешно!');

}*else*{

*return* back()->with('fail','Что-то пошло не так.

Попробуйте еще раз!'); }}

*else*{*return* back()->with('fail','Вы не авторизованы!');}}

* + - Search – функция поиска. Для поиска используется AI. Вычисления происходят на облаке, после обработки возвращаются индексы с подходящими записями из таблицы Users.

**Контроллер Export**

Export используется для работы с pdf.

* + CreatePDF – создает pdf файл из шаблоны html и используемых данных, возвращает файл для скачивания.

*// Генерация PDF*

static public function createPDF() {

$data = Auth::user();

$pdf = PDF::loadView('PDF.export',compact('data'));

*return* $pdf->download(Auth::user()->email.'.pdf');}} ViewPDF – возвращает view PDF файла( улучшенный тэг embed).

**Вывод**

Реализованная клиент-серверная архитектура полностью удовлетворяет требованиям разработки заявленного приложения. Разработанные методы и инструменты полностью позволяют грамотно и эффективно обеспечить взаимодействие между клиентом и сервером (организация запросов с помощью сервера Apache), а также между сервером и данными (с помощью фреймворка Laravel и инструмента ORM), также необходимо отметить о проектировании клиентской части программы: все компоненты полностью взаимодействую друг с другом, а также правильно работают с информацией, поступающей с сервера.

### Описание работы back-end контроллеров

Контроллером в приложении являются структура, которая регулирует выполнение функционала в зависимости от запроса. В разрабатываемом приложение контроллеры разбиты на несколько модулей, выполняющих похожие логические функции. В контроллерах осуществляется запись, удаление, добавление редактирование базы данных, рендеринг страниц, управление локальным и облачным хранилищем.

В таблице 2.1 приведены примеры запросов с конечной точкой выполнения.

Таблица 2.1. Список запросов:

|  |  |  |
| --- | --- | --- |
| Адрес запроса | Head | Конечная точка |
| / | Remember\_token, CSRF\_TOKEN | Рендер главной страницы |
|
|
|
|
| /portfolio | Remember\_token, CSRF\_TOKEN | Рендер страницы с профилем-портфолио |
|
|
|
|
|
|
|
|
| /uploadfile | Remember\_token, CSRF\_TOKEN, FILE | Загрука файла FILE на сервер, создание записи в базе данных |
| /delfile | Remember\_token, CSRF\_TOKEN  Поля: file\_id | Удаление файла |
|

Через контроллеры реализуется логика форм и страниц представленных на следующих скриншотах (Рис. 2.4 - 2. )

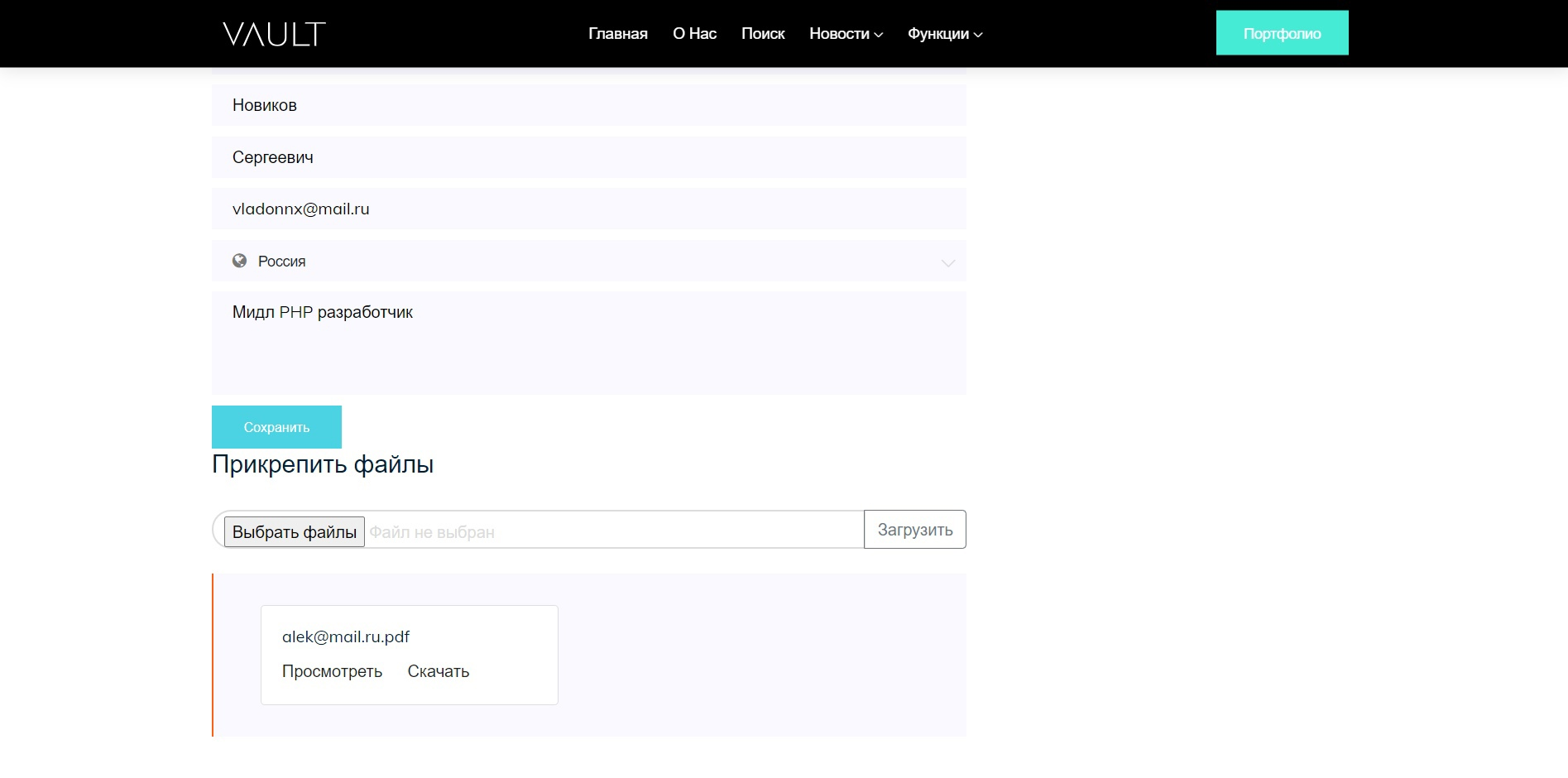
Рис 2.4 - Форма аутентификации

Рис. 2.5 - Форма загрузки файлов

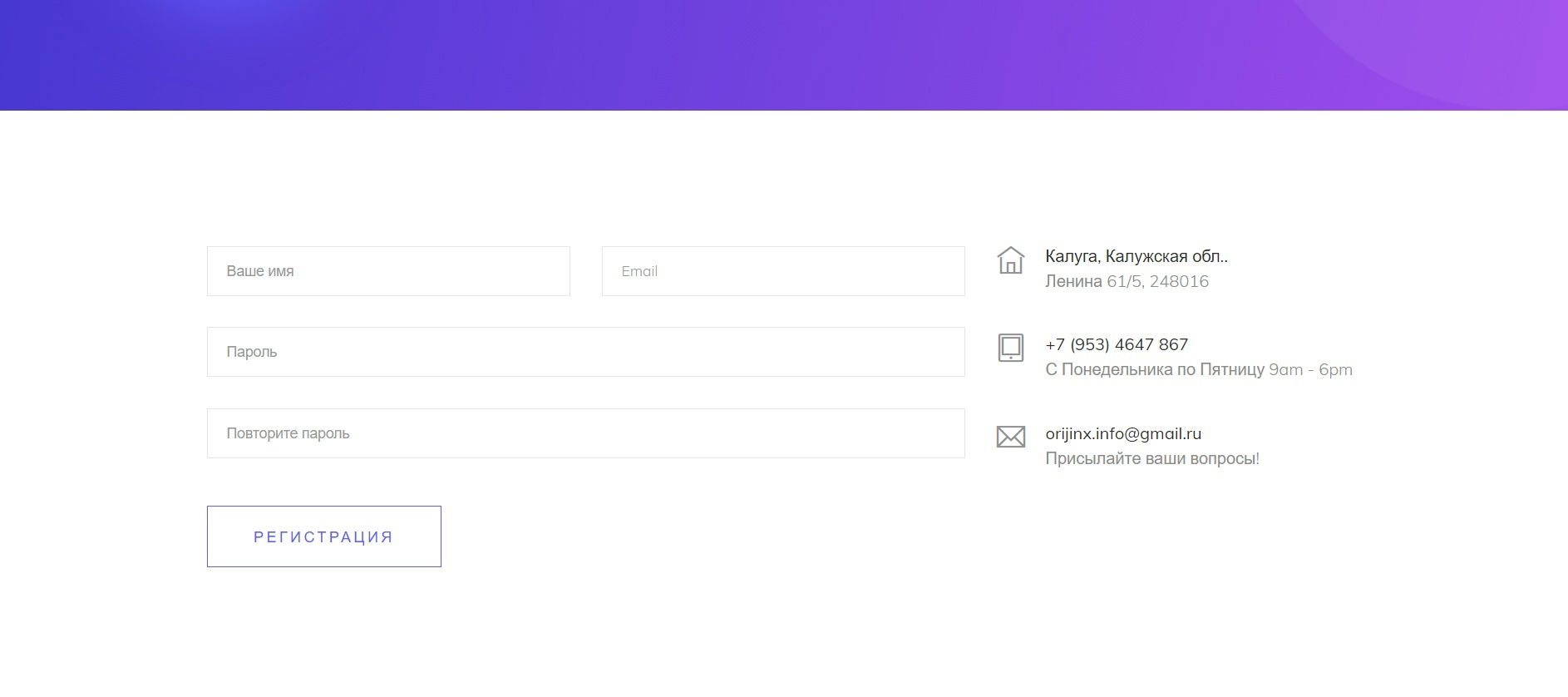
Рис. 2.6 - Форма регистрации

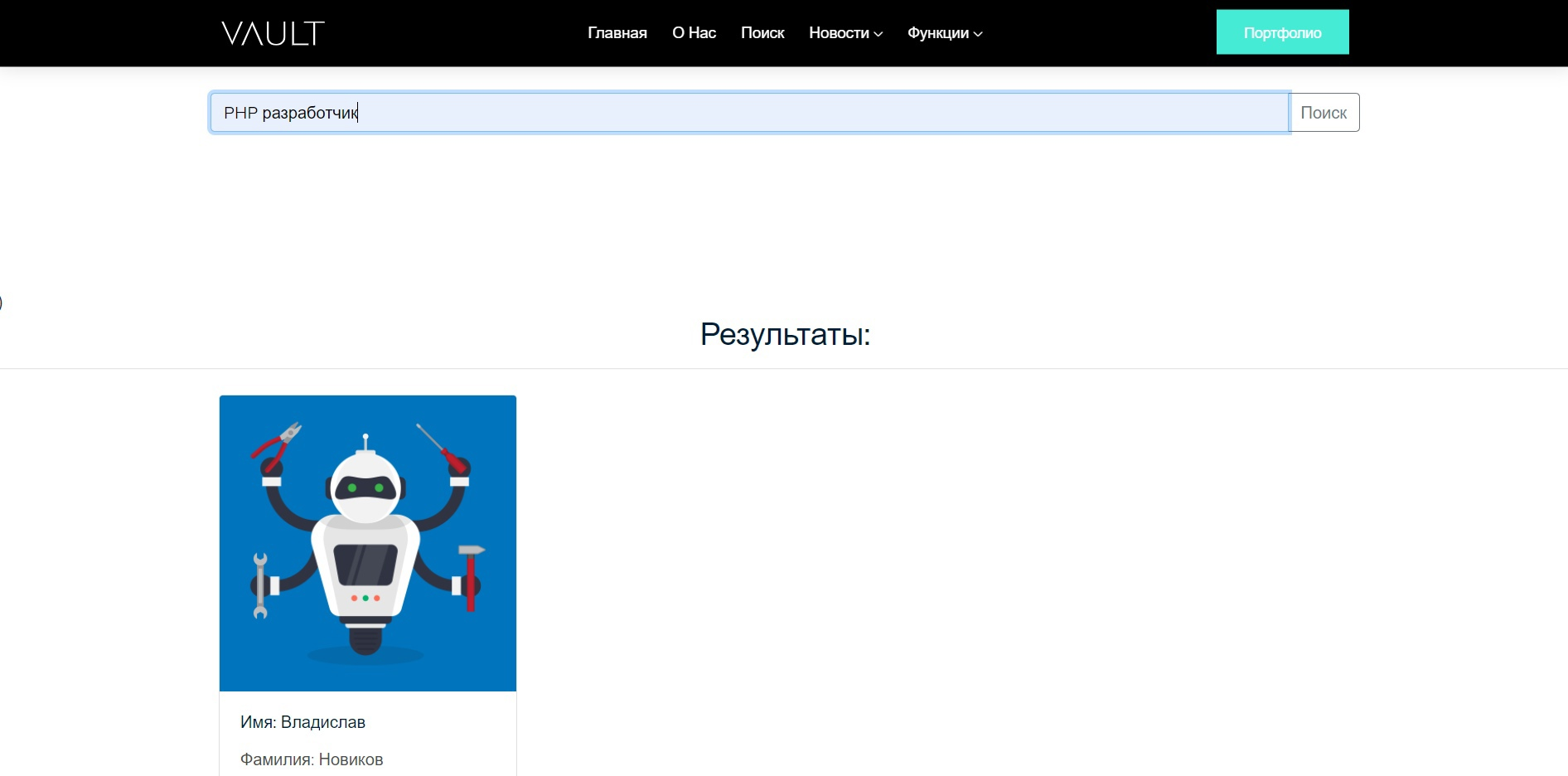
Рис 2.7 - Форма поиска



Рис. 2.8 - Рендер главной страницы

На схеме 2.1 отображена логика работы алгоритма загрузки файла на сервер. После отправки запроса происходит проверка middlewear, которая проверяет аутентифицирован ли пользователь. В зависимости от исхода результат обработки запроса разница.



Схема 2.1 – Алгоритм загрузки файла на сервер

### Описание компонента front-end

В разрабатываемом ПО, клиентом может являться любое приложение, имеющее возможность отправлять запросы, обрабатывать, манипулировать и отображать содержания [веб-сайтов](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B0%D0%B9%D1%82) или веб-приложений.

В данном случае мы представляем, что клиентом будет являться мобильное приложение. Мобильное ПО способно отправлять запросы на сервер и обрабатывать полученные от сервера данные. Возможность работы с мобильными приложениями реализуется через API, а авторизация через Json Token (Рис 2.4)

Рис 2.9 - Экран входа в мобильном приложении

### Описание Работы с Моделями и базой данных

Компонент “Модель” и компонент “База данных” тесно связаны. База данных дает возможность приложению сохранять информаци, а посредством SQL запросов ее обрабатывать. Для упрощения работы с базой данных реализуется блок моделей, который замещает SQL запросы обычными методами языка программирования. В таблице 2.2 ниже будет приведены примеры упрощения запросов на языке PHP.

Таблица 2.2. SQL запросы и их аналоги в моделях

|  |  |
| --- | --- |
| SQL запрос | Метод модели |
| SELECT \* FROM PRODUCT | $product->all(); |
| SELECT \* FROM PRODUCT WHERE ID = 1 | $product->where(“id”,1)->get(); |
| INSERT INTO PRODUCT … | $product->save(); |
|
| UPDATE PRODUCT |
| DELET FROM PRODUCT WHERE ID =1 | $product->destroy(1); |

**Вывод**

Таким образом, видно, что модули логически разделены по типам и функциям выполняемых задач. В купе, модули являются легко расширяемыми и удобными в управлении. В отдельности были просмотрены возможности компонентов системы на примерах.

## ТЕСТИРОВАНИЕ И ИНТЕГРАЦИЯ КОМПОНЕНТОВ ПРОГРАММНОГО ИЗДЕЛИЯ

### Разработка тестов и результаты тестирования

Тестирование приложения производилось в трех основных направлениях:

загрузка файлов, изменение информации в базе данных, работа с API. Для проведения тестирования использовалось Модульное тестирование. Модульное тестирование подразумевает под собой разбиение тестируемого продукта на модули – элементы законченной логики, имеющие какие-либо значимые значения, показывающие ход выполнения программного кода. Подобный вид тестирования избыточен для разрабатываемого продукта, так как в случае тестирующего приложения, тестирование отдельных его ком

### Тестирование основных пользовательских сценариев

Для тестирования основных пользовательских сценариев были разработаны сами сценарии для каждого способа запуска тестов и составлены чек-листы содержащие входные значения, контрольные точки и выходные результаты, ожидаемый и фактический.

**Запуск тестов через среду разработки**

Сценарий для загрузки файлов на сервер (рис. 3.1.):

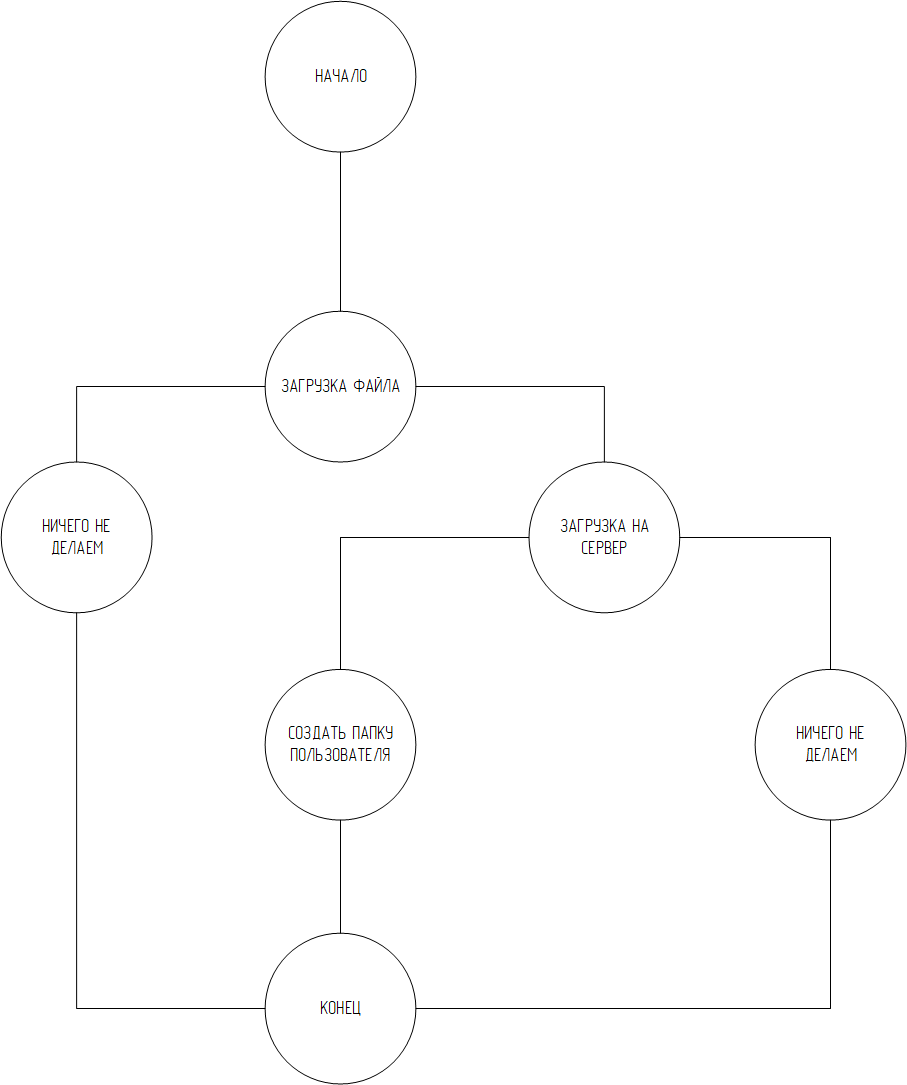


Рис. 3.1. Сценарий загрузки файлов на сервер

Чек-листы и результаты тестирования приведены в таблице 3.1.

**Результаты выполнения тестов для Модуль загрузки нового файла:**

Таблица 3.1 Результаты тестов для модуля загрузки файлов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Критерий**  **формирования теста** | **№**  **теста** | **Исходные**  **данные** | **Ожидаемый**  **результат** | **Фактический**  **результат** |
| Загрузка файла | 1 | Запрос с прикрепленным файлом и правильным HEAD | Загрузка файла на сервер | Загрузка файла на сервер |
| Неккоректная загрузка | 2 | Запрос с прикрепленным файлом, HEAD отсутствует | Вывод сообщения об ошибки | Вывод сообщения об ошибки |
| Неккоректная форма заголовка запроса | 3 | Запрос с прикрепленным файлом, Неккоректный HEAD | Вывод ошибки  (Page expired 419) | Вывод ошибки  (Page expired 419) |
| Отсутсвет файл прикрипленный файл | 4 | Запрос без файла | Вывод ошибки об отстуствии прикрипленных файлов | Вывод ошибки об отстуствии прикрипленных файлов |

Таблица 3.1 Продолжение

**Запуск тестов через консоль**

Сценарий работы API (рис. 3.2.):

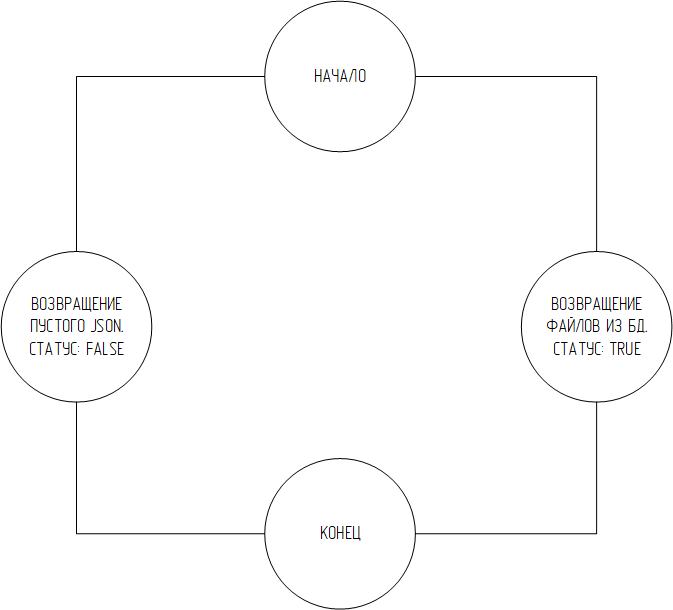


Рис. 3.2. Сценарий работы API

Чек-листы и результаты тестирования приведены в таблице 3.2.

Таблица 3.2. Чек-лист для запуска тестов через консоль

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Критерий**  **формирования теста** | **№**  **теста** | **Исходные**  **данные** | **Ожидаемый**  **результат** | **Фактический**  **результат** |
| Запись есть в бд, корректный запрос | 1 | Авторизованны  запрос, есть записи в бд | JSON с листом файлов | JSON с листом файлов |
| Отсутсвует записьв бд, корректный запрос | 2 | Авторизованны  запрос, пустые записи в бд | JSON со статусом FALSE | JSON со статусом FALSE |
| Записи в бд, неккоректный запрос | 3 | Неавторизированный запрос,  есть записи в БД | JSON со статусом NOT AUTH | JSON со статусом NOT AUTH |

**Изменение главной информации о пользователе**

Сценарий для Изменения главной информации о пользователе

(рис. 3.3.):



Рис. 3.3. Сценарий для Изменения главной информации о пользователе

Чек-листы и результаты тестирования приведены в таблице 3.3.

Таблица 3.3. Чек-лист для запуска тестов через клиентское приложение

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Критерий**  **формирования теста** | **№**  **теста** | **Исходные**  **данные** | **Ожидаемый**  **результат** | **Фактический**  **результат** |
| Все поля заполнены корректно | 1 | Введенные  данные | Изменение данных пользователя | Изменение данных пользователя |
| Введено пустое поле имени | 2 | Введенные  данные | Изменение только введеных данных | Изменение только введеных данных |
| Введено пустое поле фамилии | 3 | Введенные  данные | Изменение только введеных данных | Изменение только введеных данных |
| Все поля не заполенны | 4 | Нет | Нет ошибки | Нет ошибки |
| Введено пустое поле телефона | 5 | Введенные  данные | Изменение только введеных данных | Изменение только введеных данных |
| Неавторизированный запрос | 6 | Введены данные | Вывод ошибки авторизации | Вывод ошибки авторизации |

**Вывод**

Было проведено тестирование основных пользовательских сценариев. Данное тестирование не является полным и допускает оставление незамеченными некоторые неточности в работе программы, но является остаточно широким для допустимого покрытия приложения.

### Руководство администратора

Для развертывания приложения на сервере, нужно распокавать архив на сервере Apache (Установка и настройка сервера Apache:<https://httpd.apache.org/docs/2.4/install.html>).

После распаковки в папке public проекта найстройте файл .htaccess:

Allow from localhost

<IfModule mod\_rewrite.c>

<IfModule mod\_negotiation.c>

Options -MultiViews -Indexes

</IfModule>

RewriteEngine On

# Handle Authorization Header

RewriteCond %{HTTP:Authorization} .

RewriteRule .\* - [E=HTTP\_AUTHORIZATION:%{HTTP:Authorization}]

# Redirect Trailing Slashes If Not A Folder...

RewriteCond %{REQUEST\_FILENAME}

!-d RewriteCond %{REQUEST\_URI} (.+)/$

RewriteRule ^ %1 [L,R=301]

# Send Requests To Front Controller...

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteCond %{REQUEST\_FILENAME} !-f

RewriteRule ^ index.php [L]

</IfModule>

После настройки сервера, необходимо настроить приложение. Для Этого, в корне проекта нужно изменить пункты в файле .env (Пример найстройки):

APP\_NAME=Laravel APP\_ENV=local

APP\_KEY=base64:Ghsq7oEAFqG81HytAfDtDc6foFehkRTBbnmm+2GAxkY=

APP\_DEBUG=false

APP\_URL={URL}

LOG\_CHANNEL=stack

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE={db name}

DB\_USERNAME={username

DB\_PASSWORD={password}

BROADCAST\_DRIVER=log

CACHE\_DRIVER=file

QUEUE\_CONNECTION=sync

SESSION\_DRIVER=database

SESSION\_LIFETIME=120

REDIS\_HOST=127.0.0.1

REDIS\_PASSWORD=null

REDIS\_PORT=6379

MAIL\_MAILER=smtp

MAIL\_HOST=smtp.mailtrap.io

MAIL\_PORT=2525

MAIL\_USERNAME=null

MAIL\_PASSWORD=null

MAIL\_ENCRYPTION=null

MAIL\_FROM\_ADDRESS=null

MAIL\_FROM\_NAME="${APP\_NAME}"

AWS\_ACCESS\_KEY\_ID=

AWS\_SECRET\_ACCESS\_KEY=

AWS\_DEFAULT\_REGION=us-east-1

AWS\_BUCKET=

PUSHER\_APP\_ID=

PUSHER\_APP\_KEY=

PUSHER\_APP\_SECRET= PUSHER\_APP\_CLUSTER=mt1

MIX\_PUSHER\_APP\_KEY="${PUSHER\_APP\_KEY}"

MIX\_PUSHER\_APP\_CLUSTER="${PUSHER\_APP\_CLUSTER}"

SCOUT\_IDENTIFY=true

ALGOLIA\_APP\_ID = FESXZRY06O

ALGOLIA\_SECRET={SECRETKEY}

Укажите данные для подключения к базе данных, отлючите Debug и укажите родительский URL приложения, настройте SMTP соединение, если необходимо регенерируйте ключ приложение (Используется при хэшировании), добавьте приложение Algolia.

# Руководство пользователя

Для начала работы, пользователю предстоит регистрация. Здесь вводятся начальные данные: имя, почта и пароль (Рисунок 3.1).

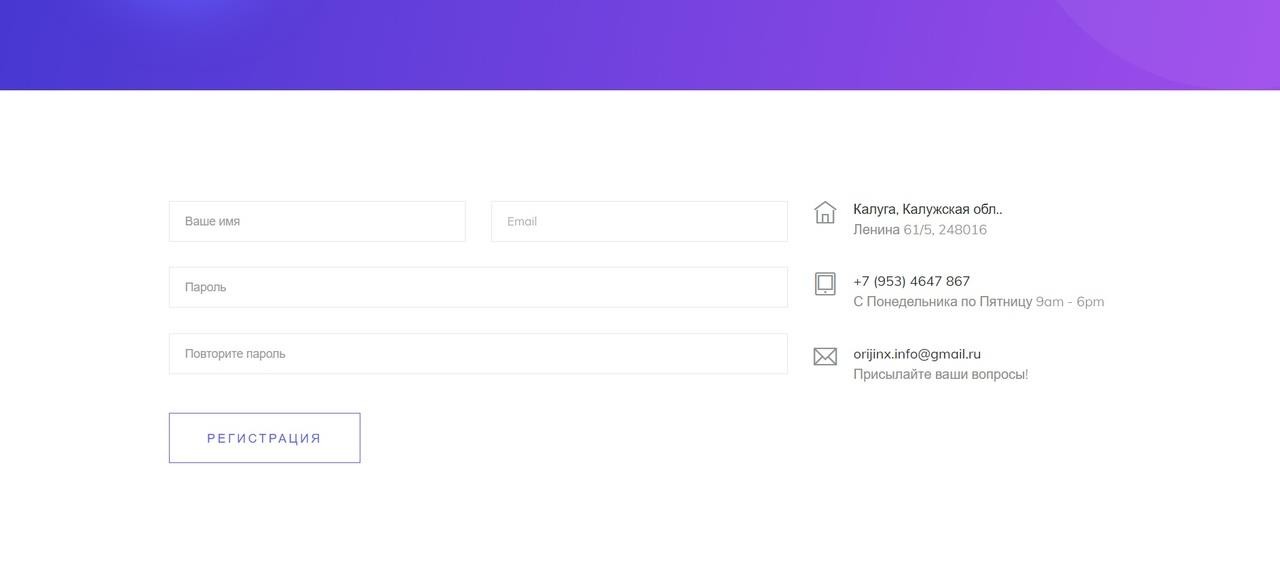


Рис. 3.1. Регистрация

Для редактирования портфолио достаточно изменить любой пункт, доступный на странице портфолио и нажать кнопку сохранить. Сервер обработает запрос и запишет/перезапишет данные в БД (Рис. 3.2.).

Для загрузки файлов, нажмите кнопку «выбрать файлы», выделите в открывшемся окне нужные к загрузке файлы( Формат Jpg,png, pdf,word) и после нажать загрузить(Рис. 3.2). Загруженные файлы появятся ниже для их просмотра и загрузки. Так же здесь можно загрузить PDF портфолио(Рис. 3.3)

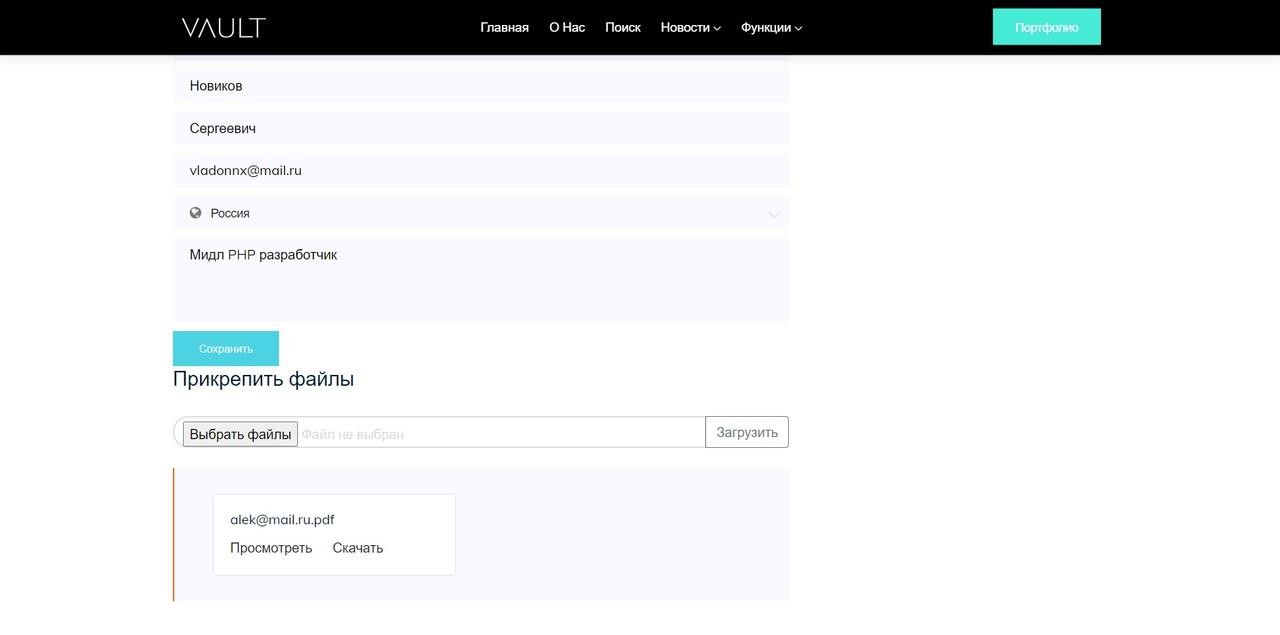


Рис. 3.2 Пункт добавления записей

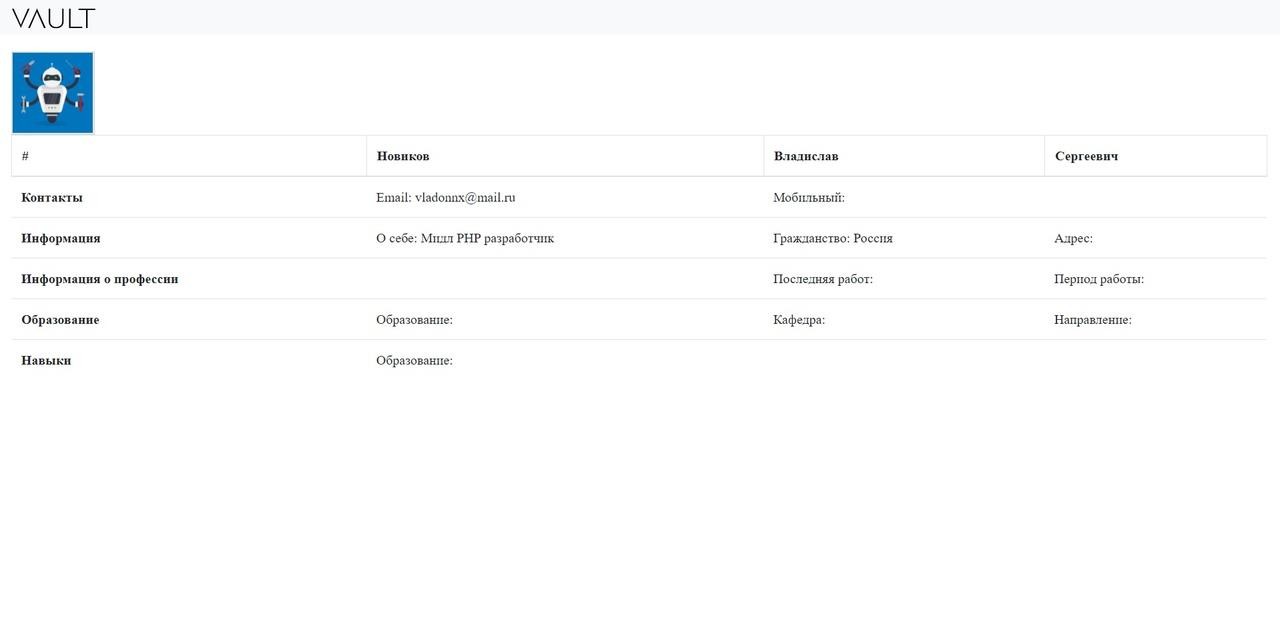


Рис. 3.3 Вывод PDF онлайн

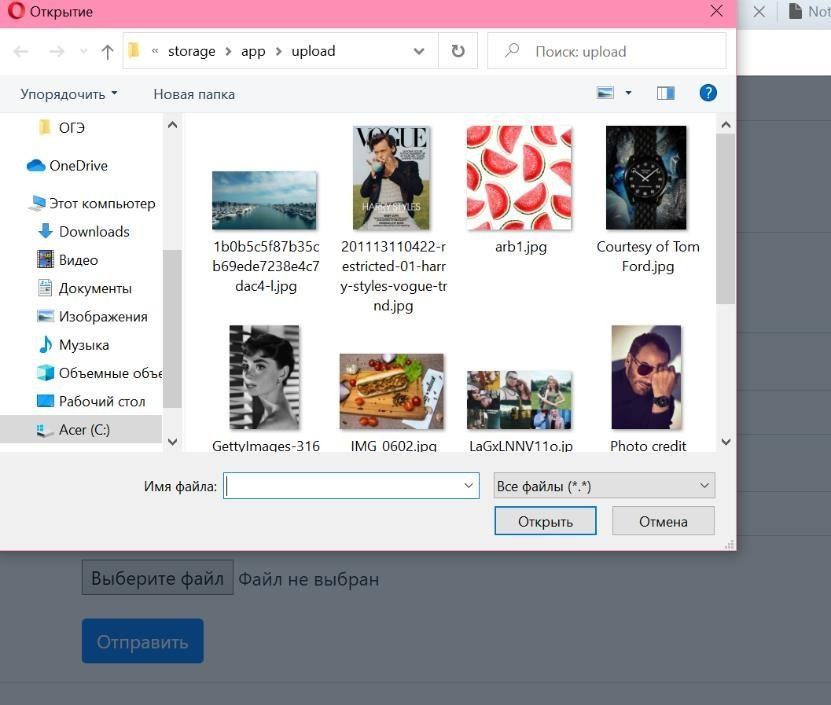


Рис. 3.4 Добавление файлов

Для поиска по сайту, нужно перейти на страницу «/search» и в строке ввести интересующую вас информацию. Поиск происходит по полям Name, Email, Profession, skills (Рис 3.5.) После поиска сервер обработает информацию и вы увидите результаты.

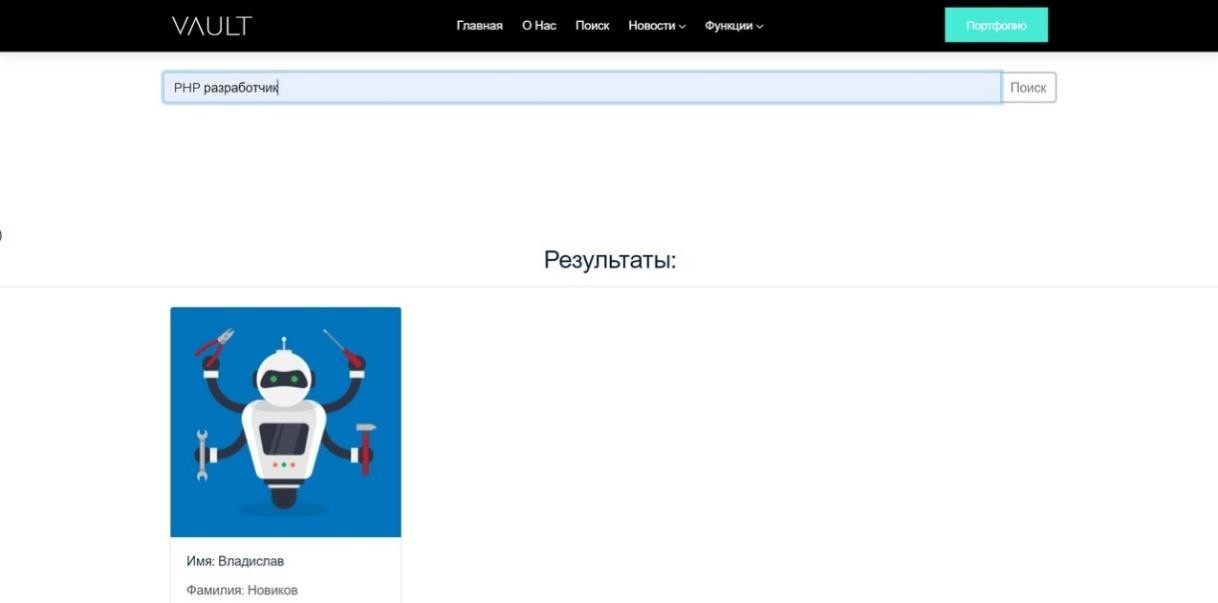


Рис. 3.5 Результат поиска

### Описание реализованной архитектуры

Реализована монолитная структура паттерного программирования MVC.

Архитектура представляет собой цельный монолит состоящий из модулей: Model - модель, View - Вид, Controller - контроллер.

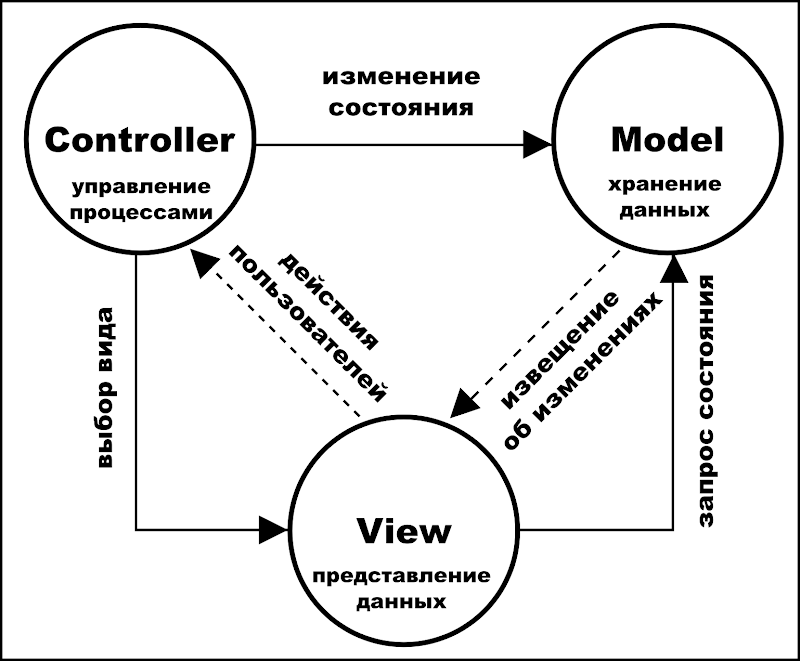


Рисунок 3.4 Архитектура приложения

На схеме указана парадигма MVC.

Пользователь отправляет запрос, через визуальную составляющую (view), контроллер обрабатывает запрос, запрашивает объектный вид записи в базе данных у модeли (model) и возвращает пользователю результат обработки запроса обратно в визуальную часть.

**Вывод**

Было написано руководство администратор и пользователя, включающие в себя настройку приложения при развертывании, а так же алгоритм использования веб-приложения. Так же было разработано описание реализованной архитектуры.

Было проведено тестирование основных пользовательских сценариев в зависимости от роли пользователя в системе с использованием разработанного приложения. Данное тестирование не является полным и допускает оставление незамеченными некоторые неточности в работе программы, но является остаточно широким для допустимого покрытия приложения.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения ВКР было выполнено проектирование и разработано веб-приложение, представляющее собой монолит работающий по парадигме MVC, позволяющий добавлять, удалять файлы портфолио, редактировать, создавать и экспортировать свое портфолио.

Были изучены аналоги представленных на рынке приложений портфолио, в результате чего был сделан следующий вывод: разработка проекта по теме централизированное хранилище портфолио, является актуальной.

Далее были изучены и рассмотрены средства реализации, в результате чего были выбраны следующие решения.

* Для создания web-приложения будет использоваться PHP, потому что, используя данный язык, легко и понятно можно написать MVC , с помощью которого будут происходить вся логика работы сервиса.
* Выбранная СУБД —MySQL, так как она обеспечивает стабильную работу при высоких нагрузках и является бесплатной.
* Использовать язык JavaScript в изначальном виде в качестве основы для front-end. Так как он является изначальной основой для всех рассмотренных ответвлений языков.
* Использовать Visual Studio Code (VS Code) в качестве редактора кода. Так как разработка производилась на компьютере с операционной системой windows.

Также был спроектирован комплекс приложений с использованием Erwin для демонстрации строения.

После был выбран оптимальный способ тестирования работоспособности приложения, результаты которого отражены в части 3 ВКР.

Результат соответствует всем заранее определенным целям и задачам ВКР. Разработанная система обладает гибкостью и удобством в эксплуатации.

**Список использованных источников**

1. Моделирование информационных ресурсов [Электронный ресурс]: учебно-методический/ Составитель Огнев Э.Н. - Кемерово : Кемеровский государственный университет культуры и искусств, 2013. - 36 с. : ил., табл. - URL: <http://biblioclub.ru/index.php?page=book&id=274218>
2. Коваленко, Ю.В. Информационно-поисковые системы [Электронный ресурс]: учебно-методическое пособие / Ю.В. Коваленко, Т.А. Сергиенко. — Омск: Омская юридическая академия, 2017. — 38 c.— Режим доступа: <http://www.iprbookshop.ru/66817.html>
3. Рыжков, И.Б. Основы научных исследований и изобретательства[Электронный ресурс] : учебное пособие / И.Б. Рыжков. — 3-е изд., стер. — Санкт-Петербург : Лань, 2019. — 224 с.— URL: <https://e.lanbook.com/book/116011> (УМО по образованию в области природообустройства и водопользования)
4. Антонов, А.В. Системный анализ [Текст]: учебник / А.В. Антонов. М.: Высш.шк., 2006. – 454 с. (УМО вузов по университетскому политехническому образованию) 19 экз.
5. Мокий, М.С. Методология научных исследований [Текст]: учебник / М.С. Мокий, А.Л. Никифоров, В.С. Мокий. - М.: Юрайт, 2015. - 255 с. (УМО вузов по экономическим направлениям и специальностям) 10 экз.
6. Рогов, В.А. Методика и практика технических экспериментов [Текст]: учеб.пособие / В.А.Рогов,А.В. Антонов, Г.Г. Поздняк. – М.: Академия, 2005. – 288 с. (УМО вузов по экономическим направлениям и специальностям) 10 экз.
7. Моделирование систем [Текст]: учебник для вузов / С.И. Дворецкий, Ю.Л. Муромцев, В.А. Погонин, А.Г. Схиртладзе. – М.: Академия, 2009. – 320 с. (Министерство образования и науки РФ) 23 экз.
8. Смелянский, Р.Л. Компьютерные сети. В 2 т. Т. 1. Системы передачи данных: учебник для вузов /Р.Л. Смелянский М.: Изд. центр «Академия», 2011 .-304 с.
9. Сейерс, Э. Х. Docker на практике / Э. Х. Сейерс, А. Милл ; перевод с английского Д. А. Беликов. — Москва : ДМК Пресс, 2020. — 516 с. — ISBN 978-5-97060-772-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131719 (дата обращения: 13.06.2020). — Режим доступа: для авториз. пользователей.
10. Митина О.А. Методы и средства проектирования информационных систем и технологий [Электронный ресурс]: курс лекций / О.А. Митина. — М. : Московская государственная академия водного транспорта, 2016. — 75 c. — Режим доступа: <http://www.iprbookshop.ru/65666.html>
11. Никифоров, А.Д. Современные проблемы науки в области технологии машиностроения [Текст]: учеб.пособие / А.Д. Никифоров. – М.: Высш.шк., 2006. – 392 с. ( Министерство образования и науки РФ) 20 экз.
12. Воронова Л.И. Big Data. Методы и средства анализа [Электронный ресурс]: учебное пособие / Л.И. Воронова, В.И. Воронов. — Электрон. текстовые данные. — М. : Московский технический университет связи и информатики, 2016 — 33 c. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/61463.html>
13. Рихтер Д. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C#. СПб: Питер, 2017. Технологии коммутации и маршрутизации в локальных компьютерных сетях: учеб, пособие для вузов / А.В. Пролетарский, Е.В. Смирнова и др.: под ред. А.В. Пролетарского.- М.: Изд-во МГТУ им. Н.Э. Баумана, 2013,- 392 с.
14. Антонов В.Ф. Методы и средства проектирования информационных систем [Электронный ресурс] : учебное пособие / В.Ф. Антонов, А.А. Москвитин. — Ставрополь: Северо-Кавказский федеральный университет, 2016. — 342 c. — Режим доступа: <http://www.iprbookshop.ru/66080.html>
15. Коузен, К. Современный Java: рецепты программирования / К. Коузен. — Москва : ДМК Пресс, 2018. — 275 с. — ISBN 978-5-97060-134-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/116121 (дата обращения: 13.06.2020). — Режим доступа: для авториз. пользователей.
16. Батчер, М. Go на практике : руководство / М. Батчер, М. Фарина ; научный редактор А. Н. Киселев ; перевод с английского Р. Н. Рагимова. — Москва : ДМК Пресс, 2017. — 374 с. — ISBN 978-5-97060-477-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/97351 (дата обращения: 13.06.2020). — Режим доступа: для авториз. пользователей.
17. Хэррон, Д. Node.js. Разработка серверных веб-приложений в JavaScript / Д. Хэррон ; перевод с английского А. А. Слинкина. — Москва : ДМК Пресс, 2012. — 144 с. — ISBN 978-5-94074-809-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/50571 (дата обращения: 13.06.2020). — Режим доступа: для авториз. пользователей.
18. Джош, Л. Современный PHP. Новые возможности и передовой опыт / Л. Джош ; перевод с английского Р. Н. Рагимов. — Москва : ДМК Пресс, 2016. — 304 с. — ISBN 978-5-97060-184-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/93269 (дата обращения: 13.06.2020). — Режим доступа: для авториз. пользователей.
19. Советов, Б. Я. Базы данных : учебник для среднего профессионального образования / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2019. — 420 с. — (Профессиональное образование). — ISBN 978-5-534-09324-7. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: <https://biblio-online.ru/bcode/438438>
20. Карпова, Т.С. Базы данных: модели, разработка, реализация : учебное пособие / Т.С. Карпова. - 2-е изд., исправ. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 241 с. : ил. ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=429003>
21. Шёниг, Г. -. PostgreSQL 11. Мастерство разработки / Г. -. Шёниг ; перевод с английского А. А. Слинкина. — Москва : ДМК Пресс, 2020. — 352 с. — ISBN 978-5-97060-671-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131714 (дата обращения: 13.06.2020). — Режим доступа: для авториз. пользователей.
22. Ржеуцкая С.Ю. - Базы данных. Язык SQL, Вологда: ВоГТУ, 2010.
23. Албахари Б., Албахари Д. C# 7.0. Справочник. Полное описание язык. М: Вильямс, 2018.
24. Государев И. Б. Введение в веб-разработку на языке JavaScript: Учебное пособие. – СПб.: Издательство «Лань», 2019. – 144с.— Режим доступа: <https://e.lanbook.com/reader/book/118648/#2>
25. Хортон А., Вайс Р. Разработка веб-приложений в ReactJS: пер. с англ. Рагимова Р. Н. – М.: ДМК Пресс, 2016. – 254 с.— Режим доступа: <https://e.lanbook.com/reader/book/97339/#4>
26. Вишневская, Т. И. Технология программирования : учебно-методическое пособие / Т. И. Вишневская, Т. Н. Романова. — Москва : МГТУ им. Н.Э. Баумана, [б. г.]. — Часть 2 — 2010. — 46 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/52411 (дата обращения: 13.06.2020). — Режим доступа: для авториз. пользователей.

**Приложение А**

Графическая часть выпускной квалификационной работы

В графическую часть выпускной квалификационной работы входят:

1. Демонстрационный чертёж (лист 1);
2. Структура БД (лист 2);
3. Функциональная модель приложения (лист 3-4);
4. Архитектура приложения (лист 5);
5. Схема выполнения запроса (лист 6);
6. Сценарии тестирования (лист 7);

