МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра IT-инжиниринг

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
PhD, доцент
_____ Т.С. Картбаев
«____» _____ 2019 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Проектирование и разработка веб-сайта «WorkPoint» для центра занятости

Специальность: 5В070400 – «Вычислительная техника и программное обеспечение»

Выполнила: Бактыгереева А.К. Группа: ВТ-15-2
Научный руководитель: к.т.н., доцент Турганбаев Е.С.

Консультанты:
по экономической части: к.э.н., профессор _____ Ж.Г. Аренбаева
«13» мая 2019 г.

по безопасности
жизнедеятельности: д.т.н., ст. преп. _____ Ш.Ш. Бекбасаров
«14» мая 2019 г.

по применению
вычислительной техники: ст. преп. _____ М.Н. Майкотов
«14» мая 2019 г.

Нормоконтролер: ст. преп. _____ А.А. Айтказина
«15» мая 2019 г.

Рецензент: д.т.н., профессор _____ Балгабаева Л.Ш.
«____» _____ 2019 г.

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра IT-инжиниринг

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Бактыгереевой Амангуль Кайратовне

Тема проекта: Проектирование и разработка веб-сайта «WorkPoint» для центра занятости

Утверждена приказом по университету № 124 от «26» октября 2018 г.

Срок сдачи законченного проекта «24» мая 2019 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): Руководство системы менеджмента качества на предприятии; международные стандарты ИСО-9001, данные преддипломной практики.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:

- аналитическая часть;
- проектная часть;
- экспериментальная часть;
- экономическая часть;
- безопасность жизнедеятельности;
- приложение А. Техническое задание;
- приложение Б. Листинг программы;
- приложение В. Акт внедрения.

Перечень графического материала (с точным указанием обязательных чертежей): представлены 10 таблиц, 27 иллюстрации.

Основная рекомендуемая литература:

1 Ларри Ульман Основы программирования на PHP [Электронный ресурс]/ Ларри Ульман— Электрон. текстовые данные.— М.: ДМК Пресс, 2007.— 286 с.

2 Панфилов К.С. Создание веб-сайта от замысла до реализации [Электронный ресурс]/ Панфилов К.С.— Электрон. текстовые данные.— М.: ДМК Пресс, 2009.— 440 с.

3 Горнаков С.Г. Осваиваем популярные системы управления сайтом. CMS [Электронный ресурс]/ Горнаков С.Г.— Электрон. текстовые данные.— М.: ДМК Пресс, 2009.— 336 с.— Режим доступа: http://www.iprbookshop.ru/7926.— ЭБС «IPRbooks».

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Аренбаева Ж.Г.	04.03.2019 – 13.05.2019	
Безопасность жизнедеятельности	Бекбасаров Ш.Ш.	04.03.2019 – 14.05.2019	
Программное обеспечение	Майкотов М.Н.	04.03.2019 14.05.2019	
Нормоконтролер	Айтказина А.А.	02.04 - 15.05.2019	

ГРАФИК
подготовки дипломной проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Аналитическая часть	15.01.2019	
Проектная часть	30.01.2019	
Экспериментальная часть	15.05.2019	

Дата выдачи задания «29» октября 2018 г.

Заведующий кафедрой _____ Т.С. Картбаев

Научный руководитель проекта _____ Е.С. Турганбаев

Задание принял к
исполнению студент _____ А.К. Бактыгереева

Аңдатпа

Дипломдық жобаның тақырыбы «Жұмыспен қамту орталығына арналған “WorkPOINT” веб-сайтын құрастыру және әзірлеу».

Бұл жұмыстың мақсаты: үй шаруашылықтары мен іскерлік мәселелерді шешу үшін жеке кәсіпкерлер үшін онлайн-іздеу қызметін құру. Платформа кез-келген жұмысты орындауға мұқтаж клиенттерді, сондай-ақ қосымша жұмыс іздеген немесе қосымша табыс іздейтін білікті орындаушыларды біріктіреді. Алға қойған мақсатқа жету үшін келесідей технологиялар қолданылды:

– клиенттік интерфейсті құру үшін: HTML, CSS, JavaScript, Angular;
– мәліметтер базасын жобалау үшін: MySQL;
– серверлік логиканы құру үшін: Java, Spring MVC, MyBatis.

Дипломдық жоба кіріспеден, 5 бөлімнен және қорытындыдан тұрады.

Кіріспеде таңдалған тақырыптың өзектілігі ашылады, өңдеу мақсаты қойылады және орындауға қажетті міндеттер, өңдеуді қолдану аймағы анықталады.

Бірінші бөлімде есепті шешуге арналған технологияларға талдау жасау мен таңдау келтірілген.

Екінші бөлімде қосымшаны жобалау сипатталады.

Үшінші бөлім қосымшаны құруды сипаттаудан, қойылған мақсатты практикалық шешудің қадамдық сипатталуынан тұрады.

Төртінші бөлімде өңделген жобаның экономикалық мақсаттылығын негіздеу қарастырылған.

Бесінші бөлімде жүзеге асырылатын жоба аясында еңбек шарттарын жақсарту бойынша шаралар қарастырылған.

Қорытындыда клиент-серверлік веб-қосымшаларды құрудың қазіргі заманғы құралдары меңгерілген және қойылған мақсатқа сай өңдеу орындалған, ары қарай қосымшаны толыққанды өнімге дейін жақсарту стратегиялары құрылған.

Аннотация

Тема дипломного проекта: «Проектирование и разработка веб-сайта “WorkPOINT” для Центра занятости».

Целью данной работы является: создание онлайн сервис поиска частных специалистов для решения бытовых и бизнес задач. Площадка объединяющая заказчиков услуг, которым необходимо выполнить какую-либо работу, и компетентных исполнителей, ищущих подработку или дополнительный заработок. Для достижения поставленной цели использовались технологии:

– для разработки клиентского интерфейса: HTML, CSS, JavaScript, Angular;

– для проектирования базы данных: MySQL;

– для разработки серверной логики: Java, Spring MVC, MyBatis.

В дипломный проект входит введение, 5 глав и итоговое заключение.

Во введении раскрывается актуальность выбранной темы, ставится цель разработки и задачи, которые необходимо выполнить, определяется область применения разработки.

В первой главе производится анализ и выбор технологий для решения задачи.

Во второй главе описывается проектирование приложения.

Третья глава представляет собой описание разработки приложения, содержится пошаговое описание практического решения поставленной задачи.

В четвертой главе производится обоснование экономической целесообразности разрабатываемого проекта.

В пятой главе рассматриваются мероприятия по улучшению условий труда в рамках реализуемого проекта.

Заключение освещает результаты исследования и разработки:

– изучены современные средства разработки клиент-серверных веб-приложений и выполнена разработка согласно поставленной цели;

– разработана дальнейшая стратегия улучшения приложения до полноценного продукта.

Annotation

Theme of the graduation project: «Designing and developing the ”WorkPOINT” website for the Employment Center».

The purpose of this work is: the creation of an online search service for private professionals to solve household and business problems. The platform unites customers who need to perform any work, and competent performers who are looking for a side job or additional income. Technologies used to achieve the goal:

– development of the client user interface: HTML, CSS, JavaScript and Angular;

– creation of the database: DBMS PostgreSQL;

– development of the server logic: Java, Spring MVC, MyBatis.

The graduation project includes an introduction, 5 chapters and a conclusion.

The introduction reveals the relevance of the chosen topic, sets the development goal and the tasks that need to be performed, and determines the scope of the development.

The first chapter analyzes and selects technologies for solving the problem.

The second chapter describes the design of the application.

The third chapter is a description of the development stage of the application and contains a step-by-step description of the practical solution of the task.

The fourth chapter examines measures to improve working conditions in the framework of the ongoing project.

The conclusion highlights research and development results such as:

– studied modern development tools for client-server web applications and developed according to the set goal;

– further strategies to improve the application in to a full product.

Содержание

Введение

Web-сайты используются как механизм общения между владельцами сайта и его пользователями, а иногда - между самими пользователями. Владельцы сайтов обычно ставят задачу и определяют основные правила взаимодействия, в то время как пользователи — это те люди, что посещают сайт и пытаются пользоваться представленным на нем содержимым или его возможностями. Канал связи между владельцем сайта и его посетителем может изменяться. Зачастую владельцы сайтов предоставляют пользователям информацию для ее потребления, делая из этого отчасти одностороннее взаимодействие.

Для этого необходимо было создать сайт, отвечающий следующим требованиям:

- современный дизайн;
- дружественный интерфейс;
- удобство загрузки.

Целью данной работы является: создание онлайн сервис поиска частных специалистов для решения бытовых и бизнес задач. Площадка объединяющая заказчиков услуг, которым необходимо выполнить какую-либо работу, и компетентных исполнителей, ищущих подработку или дополнительный заработок. При этом соответствующего современному положению вещей в сфере рекламы с учетом особенностей Интернета.

Актуальностью темы является то, что в последнее время удаленная работа привлекает не только тех самозанятых, кто называет себя «freelancer». Ею интересуются люди творческих и технических профессий, у которых есть постоянная работа или кто не может приходить в офис, но готовые выполнять дополнительную работу, а также организации и компании, которые нуждаются в удаленных сотрудниках.

Ни печатная продукция, ни тем более радио ли телевидение не могут подавать информацию в таком удобном, презентабельном и убеждающем виде, как это делает WEB-сайт. Исчерпывающая информация, возможность просмотра вакансий, размещения резюме, рубрика «ответы на часто задаваемые вопросы», размещенные на WEB-сайте, избавляют от необходимости разъяснять одни и те же тривиальные вопросы по телефону или факсу, позволит высвободить рабочее время сотрудников фирмы для более эффективного решения насущных задач. Именно поэтому создание WEB-сайта для организации продиктовано современной необходимостью.

Сам сайт создается для центра занятости города Алматы, которая осуществляет трудоустройство безработных граждан, помощь в организации собственного дела, подбор кандидатов на вакантное место для работодателей, проводит обучение с целью дальнейшего трудоустройства.

Основными задачами, которые должны быть решены в дипломном проекте, являются:

- исследование предметной области;
- определение определение функциональности: регистрация, просмотр вакансии, заполнение анкеты;
- определение технических решений;
- определение системный решений;
- определение программного обеспечения;
- определение технологического обеспечения;
- разработка пользовательского интерфейса: тестер, отладка;
- размещение сайта на хостинге
- внедрение;

Результатом дипломной работы стал сайт, который представляет собой множество логически взаимосвязанных между собой страниц. Основным принципом дизайна, реализованного в работе, стал принцип простоты и доступности. Методом исследования является изучение и анализ существующих сайтов. Научная новизна и теоретическая значимость исследования состоит в выделении и систематизации методики разработки сайта. Практическая значимость исследования заключается в том, что на основе полученных результатов исследованы критерии и методика разработки веб-сайта.

1 Основная часть

1.1 Описание предметной области

Большинство дискуссий относительно Web-дизайна, как правило, быстро уходит от своей основной темы, т.к. этот термин разные люди понимают по-своему. Хотя каждый имеет некоторое представление о том, что такое Web-дизайн, лишь немногие способны дать ему точное определение. Определенные компоненты, такие как графический дизайн или программирование, затрагиваются в каждой дискуссии, однако их важность в процессе создания сайтов варьирует от человека к человеку и от сайта к сайту. Некоторые считают наиболее важной стороной Web-дизайна создание и структурирование содержимого — или, выражаясь более формально, информационную архитектуру. Другие факторы — среди множества прочих, простота использования, ценность и назначение сайта в рамках общей деятельности организации, доставка сайта - определенно пребывают в сфере Web-дизайна. Набравшись влияний из библиотечного дела, графического дизайна; программирования, организации сетей, проектирования пользовательских интерфейсов, удобства использования и множества других источников, Web-дизайн является мультидисциплинарной областью деятельности.

В основе использования Интернет-ресурсов для привлечения потребителей, наряду с прочими маркетинговыми инструментами, лежит принцип прямого отклика. Поэтому первое принципиальное условие для построения Web-сайта, призванного продать что-либо – это возможность заинтриговать посетителя, а, впоследствии, и побудить его предпринять какиелибо действия. Несмотря на очевидные факты, множество веб-разработчиков игнорируют данный момент. Как следствие – посетитель сайта, просмотрев его стартовую страницу и не заинтересовавшись ее содержимым, покидает ресурс.

Информационный сайт – это огромный портал, организованный как многоуровневое объединение разных ресурсов и сервисов, обновление которых происходит в реальном времени. Информационный портал содержит колоссальное множество количество контента, как правило, уникального и рассчитан на большие нагрузки по посещаемости.

Каждый квалифицированный Web-разработчик должен помнить о том факте, что в век интерактивных технологий на любого пользователя сети Интернет, являющегося потенциальным потребителем, обрушивается масса всевозможной информации, в частности, сообщений, носящих рекламный характер. Поэтому в области маркетинга существует жесткая конкуренция, выдержать которую, привлекая потенциального потребителя, можно только,

путем создания действительно качественного Web-сайта. Следовательно, Web-сайт должен иметь не только привлекательный, яркий дизайн, но и содержать полезную и интригующую потенциального потребителя информацию. Содержание главной страницы сайта должно не только стимулировать посетителя перейти со стартовой страницы на оставшиеся, а в дальнейшем и совершить покупку, но и посещать ресурс впоследствии, рекомендуя его своим близким.

1.1.1 О программном продукте

Фриланс – работа по принципу свободного художника. Работник сам ищет себе заказчиков и клиентов, сам договаривается о цене, самостоятельно выполняет работу и сдаёт её. В отличие от штатных сотрудников фрилансеры выполняют только конкретные задачи, получая разовую выплату. Направления:

- работа с текстами;
- графика и дизайн;
- разработка и продвижение сайтов;
- контент-менеджмент;
- тестирование и визуализация игр;
- дизайн и архитектура;
- журналистика.

Первый Казахстанский сайт для удаленной работы фрилансеров. На данном сайте смогут работать заказчики и фрилансеры со всего мира. Для сайта Workpoint расстояние не имеет никого значения, поэтому исполнители получают работу, а заказчики – ценных специалистов со всего мира. Сайт сделает удаленное сотрудничество удобным, специалисты смогут покинуть тесные офисы и смогут реализовать себя в интересных проектах в казахстанских и заграничных компаниях. У заказчиков будет огромный выбор исполнителей, готовых работать здесь и сейчас, а фрилансерам – неиссякаемый поток проектов.

Впрочем, наш фриланс - проект создан не только с целью помочь желающим найти подходящую удаленную работу, но и для того, чтобы каждый самозанятый и фрилансер имел возможность встретиться здесь со своими коллегами, обсудить профессиональные вопросы, выбрать необходимый инструментарий и просто получить удовольствие от общения с друзьями, рассказать о своих достижениях, поделиться своими проблемами, посмотреть удаленная работа вакансии.

1.1.2 Определение Web-дизайна

Пять областей охватывают основные аспекты Web-дизайна:

- содержимое;

Сюда входят форма и организация содержимого сайта. Возможный диапазон — от того, как написан текст до того, как он организован, представлен и структурирован с помощью технологии разметки, такой как HTML;

- зрительные образы;

Это относится к компоновке экранного пространства на сайте. Эта компоновка обычно создается с помощью HTML, CSS, JavaScript, Flash и может включать графические элементы, выполняющие функции украшения или навигации. Визуальная сторона сайта — это наиболее очевидный аспект Web-дизайна, но не единственная, и не самая важная, сторона дисциплины;

- технология;

Хотя применение разнообразных базовых Web-технологий вроде HTML или CSS попадает в эту категорию, под технологией в этом контексте чаще подразумеваются различные интерактивные элементы сайта, в особенности созданные с использованием программных методов. Это могут быть элементы в диапазоне от языков сценариев, работающих на стороне клиента, наподобие JavaScript, до серверных приложений, таких как Java-сервлеты;

- доставка;

Скорость и безотказность доставки сайта по сети Internet или внутренней корпоративной сети связаны с применяемым аппаратным или программным обеспечением и задействованной сетевой архитектурой;

- назначение;

Причина, по которой сайт существует, часто связанная с экономическими вопросами, вероятно, является наиболее важной частью Web-дизайна. Этот элемент следует учитывать при принятии любых решений, затрагивающих другие области. Степень, в которой каждая сторона Web-дизайна оказывает воздействие на сайт, может изменяться в зависимости от типа создаваемого сайта. Личная домашняя страничка обычно не связана с экономическими соображениями, характерными для веб-сайта. Внутренняя сеть производственной компании может не попадать под влияние соображений, 6 связанных с визуальным представлением, важных для общедоступного Web-сайта, рекламирующего остросюжетный фильм;

1.2 Назначение разработки

В общем случае для успешной работы сайта необходимо, чтобы он выполнял несколько задач одновременно. Представьте, что ваш сайт — это офис, в котором в одном отделе сидят менеджеры, которые дают бесплатные консультации, во втором — маркетологи, в третьем — менеджеры, которые выставляют счета. Причем все эти отделы между собой успешно

взаимодействуют. Еще очень важно, чтобы в ваш виртуальный офис было несложно попасть. Для этого необходимы эффективная реклама и легко запоминающийся адрес. Консультацию по этим и другим вопросам можно получить по телефону или email. Такой сайт используется для:

- распространение основной информации – сайт используется для распространения информации о продукции и услугах, предоставляемых организацией. Другая предоставляемая информация в основном включает данные о том, как связаться с фирмой способами, отличными от Web;
- поддержку – часть сайта может быть предназначена для оказания клиентам информационной помощи относительно эффективного пользования продуктами и услугами, предоставляемых организацией;
- связи с инвесторами – открытые акционерные общества или компании, ищущие внешних инвесторов, могут создать сайт или раздел сайта для опубликования информации о текущей экономической ситуации в компании, а также о будущих возможностях для вложения инвестиций;
- связь с общественностью – многие фирмы используют Web-сайты для предоставления информации различным организациям, занимающимся сбором данных, а также добровольного обнародования основной информации о фирме;
- поиск служащих – Web-сайты часто используются для размещения объявлений о приёме на работу и преимуществах работы в компании.

Само приложение включает в себя три части:

– регистрация в системе:

Данный пункт выполняется лишь однажды, в первые рабочие дни после трудоустройства. Также этот пункт может быть включен кадровыми работниками компании в так называемый процесс адаптации.

При регистрации пользователю будет необходимо ввести самый минимум информации о себе. Вся информация, введенная пользователем при регистрации, необходима для получения доступа к аккаунту пользователя, а также его корректных идентификационных данных.

Предполагается, что данное приложение будет устанавливаться на локальный сервер компании, поэтому имея доступ к базе данных, проблему утери пароля или имени пользователя будет решать системный администратор компании, которая использует приложение.

Регистрация позволяет каждому из сотрудников индивидуально использовать приложения, и данные, вводимые им в приложение, будут защищены от просмотра другими участниками компании.

Логином для входа будет являться номер телефона сотрудника. По совместительству номер телефона будет выполнять функции уникального идентификатора для каждого пользователя в системе.

С помощью кнопок, расположенных рядом с каждым элементом списка, пользователь может редактировать задачи:

– перевести задачу в статус выполненной: отмечая галочкой чекбокс находящийся вначале строки мы можем изменить статус задачу на выполненную. Таким образом, когда пользователь в следующий раз будет запрашивать список с этой задачей, то она уже будет отмечена как выполненная;

– изменить описание состояния задачи: так, только добавленная задача имеет описание «Не выполнялась». После пользователь может описывать задачу как «в процессе», если она выполняется на данный момент, либо «отложено», если пользователь не успел закончить задачу в течение рабочего дня, либо потребовалось срочное выполнение другой задачи. И финальным описанием задачи является «Выполнена». Это описание появляется у задачи тогда, когда статус задачи переведен в выполненный.

– нажав на иконку, расположенную справа от любой из задач, пользователь может удалить эту задачу. Это может понадобиться пользователю в том случае если эта задача ему больше не актуальна для выполнения, или в том случае, если он, например, просто допустил ошибку в тексте задачи и хочет добавить ее заново, уже исправленную.

Такой способ планирования рабочего дня приводит к тому, что человек четко видит, что он делает и как быстро (сколько задач за день он может выполнять). Это очень влияет на его мотивацию к работе и стремлению выполнять задачи еще лучше и быстрее. А это в свою очередь влияет на коммерческий успех компании, в которой этот человек работает.

Помимо этого, проект выполнен в формате веб-приложения. На данный момент большинство разработчиков стараются делать свои приложения, во-первых, клиент-серверными: это позволяет снизить вычислительную нагрузку на машину пользователя, а во-вторых, веб-приложениями, чтобы пользователям не было необходимости скачивать клиентскую часть приложения, и сразу приступать к работе.

В скором времени такой подход к разработке программного обеспечения станет доминирующим, поскольку происходит значительная экономия вычислительных ресурсов, а кроссбраузерная и адаптивная верстки избавляют нас от написания уникальных программ для разных операционных систем.

Таким образом приложение сможет работать под управлением любой операционной системы, которая имеет браузер в своем арсенале.

1.2.1 Требования к надежности и эффективности

Web-сайт должен иметь дружественный пользовательский интерфейс и должен позволять работать пользователю, не обладающему специальными знаниями информационных технологий.

Для обеспечения устойчивого функционирования Web-сайта необходимо:

- осуществлять проверку его совместимости с различными браузерами;
- осуществлять проверку его совместимости с различными кодировками операционных систем;
- осуществлять проверку его совместимости с разрешениями экрана;
- для надежности сохранять данные в сессии, в специальных файлах, которые содержат информацию, хранящуюся в базе данных;
- обеспечивать восстановление информации утраченной в результате сбоя в работе сервера.

Для эффективной работы Web-сайта необходимы следующие условия:

- качество работы сервера, где происходит хранение Web-сайта;
- обеспечивать восстановление информации утраченной в результате сбоя в работе сервера;
- возможность отката или отмены внесенных изменений, а также возможность перехода к изначальным входным данным.

Web-сайт должен быть легко проверяем на предмет работоспособности, должны быть подготовлены специальные тесты с заранее известными результатами правильной работы сайта. Тесты должны предусматривать проверку работы форума – вход пользователя, регистрацию, поиск по форуму и добавление новых сообщений, проверку гостевой книги, проверку работоспособности анкеты – её реакцию на ввод неверных типов данных, проверку работы прейскуранта – изменение и ввод данных, проверку счётчика посещений и проверку взаимодействия администраторской панели с пользовательской частью.

1.2.2 Требования к составу выполняемых разделов

Web-сайт необходимо разбить на несколько разделов.

Раздел «Главная» - отображает краткую деятельность компании, последние новости.

Раздел «О компании» - один из наиболее важных разделов сайта - отображает список готовых работ компании, необходим для наглядной демонстрации дизайнерских возможностей фирмы.

Раздел «FAQ» - это раздел необходимый для интерактивного общения и дискуссий между пользователями сайта, также содержит аналитическую информацию для потенциальных клиентов или просто заинтересованных лиц

Раздел «Проекты» - имеющиеся вакансии в компании. Для удобства пользователя и администратора предусмотрена анкета. После заполнения пользователем анкета помещается в базу данных для дальнейшего хранения или просмотра.

Раздел «Контакты» - пользователь имеет возможность просмотреть, где находится компания и как с ней связаться, содержит в себе информацию о сотрудниках и деятельности компании в целом.

Раздел «Аккаунты» - можно будет зарегистрироваться любому фрилансеру или компании.

1.2.3 Требования к организации входных и выходных данных

Для отображения Web-сайта в окне пользовательского браузера необходимо, чтобы происходил обмен входными и выходными данными между пользователем и сервером – этот процесс называется доставкой.

Доставка сайта пользователю так же важна, как и его построение. Популярность сайта очень сильно зависит от времени его отклика на действия посетителя, что в свою очередь отражается на отношении к нему конечного пользователя. Большинство дизайнеров осознает необходимость в скорости. Даже в этом случае, что бы объяснить низкую скорость загрузки сайта, дизайнеры обычно акцентируют внимание на нескольких аспектах: размер графических файлов сайта или скорость подключения конечного пользователя. Действительные причины задержки могут быть не столь очевидными. Скорость загрузки может определяться множеством факторов, например, особенностями сети: трафик, вид протокола, вид сервера, а также содержание сайта. Дизайнеры должны определять все аспекты доставки сайта, потому что конечный пользователь не может разделять отдельные компоненты передачи информации, а рассматривает сайт как единую систему.

Даже когда все передается нормально, работа над Web-сайтом может продолжаться и отнимать достаточно много времени. Web-мастер всегда найдет, чем заняться: обслуживанием содержания, восстановлением неработающих ссылок, а также проверкой доступности сайта. Один из интересных аспектов сопровождения сайтов - анализ его использования. В Web существует возможность понять, что делают пользователи при посещении сайта, путем анализа файлов системного журнала. Можно проанализировать эти данные для улучшения дизайна.

Процесс запроса страницы проходит приблизительно следующим образом:

- пользователь вводит адрес URL;
- формируется HTTP-запрос;
- происходит разрешение доменного имени;

- если разрешение прошло удачно, HTTP-запрос передается на сервер средствами TCP/IP;

- сервер принимает запрос и через некоторое время отвечает;

- формируется и возвращается браузеру либо успешный, либо аварийный ответ;

- браузер анализирует поступивший ответ, отображает или сохраняет данные;

- если ответ содержит другие объекты, процесс повторяется.

1.3 Этапы разработки веб-сайта

Основные этапы создания веб-сайта:

- постановка целей и задач сайта;

Далеко не каждый клиент нашей веб-студии видит различия в видах веб-сайта, способен самостоятельно определить его цели и задачи. Анализируем для чего клиенту сайт, какие задачи он должен решать: выполнять функцию представительства компании в Интернете или же продавать товар и услуги, стать качественной рекламой компании или быть стартапом для молодой фирмы. Все это оговаривается с заказчиком. Он должен сам понимать зачем ему сайт, в чем он поможет его компании и с чем справиться не сможет. После того, как поставлены задачи, определяется целевая аудитория;

- создание, проработка технического задания (ТЗ);

Техническое задание - это основа, на которую полагается каждый специалист, участвующий в разработке. Поэтому в составлении ТЗ участие заказчика – необходимо. Оно оговаривается и редактируется до тех пор, пока клиент даст согласие и подпишет документ. Только после этого проект переходит в стадию создания. На этапе формирования и обсуждения ТЗ также разрабатывается структура сайта, его навигация, количество категорий и подкатегорий, их последовательность размещения и т.д. Вся информация также включается в документ, который подписывается заказчиком;

- создание макета дизайна сайта;

Все этапы создания сайта важны для качественного функционирования ресурса, но разработка дизайна – один из главных. Ведь дизайн – это то, что видит посетитель в первую очередь, оценивает его и принимает решение остаться на странице или закрыть вкладку браузера.

Дизайнер основывается на техническом задании, рисует кнопки, баннеры и другие графические элементы. Другими словами, тот прототип, который был создан на первом этапе разработки сайта, получает эстетичный внешний вид, производится в цветах, выбранными заказчиком. Если компания имеет

корпоративный стиль, то дизайн разрабатывается в его соответствии. Или же может сначала разработаться фирменный стиль, а после - дизайн на его основе.

Важно отметить, что дизайнер рисует дизайн не каждой страницы, а шаблоны нескольких основных, используя тенденции веб-дизайна. Готовый макет передается клиенту и ожидает своего одобрения. Если необходимы доработки, дизайнер выполняет их и снова показывает макет заказчику. Дизайн дорабатывается до тех пор, пока он не будет утвержден.

- верстка и программирование;

Верстальщик с помощью языка HTML переводит готовый дизайн в рабочий проект. Ресурс получает жизнь, становится динамичным, все кнопки работают. Ресурс становится кроссбраузерным и правильно отображается во всех существующих интернет-браузерах. На это же этапе создаются стили CSS.

- наполнение контентом;

Наполнение страниц графическим и информационным контентом, получается размещается видео, фото, тексты и другая информация, которую сможет увидеть или прочитать посетитель. На основе семантического ядра пишутся SEO-статьи, контент-менеджер размещает графические элементы в логической структуре. Страницы проходят внутреннюю оптимизацию.

- тестирование;

Выполняет работу тестировщик, который мониторит функциональность ресурса по ряду критериев и выявляет ошибки, которые должны быть устранены.

1.4 Выбор инструмента для реализации проекта

Данный подраздел повествует о том, какие технологии были использованы при разработке дипломного проекта, их основные достоинства и недостатки. Также поясняется, почему были выбраны именно эти технологии, и за какую часть в приложении они несут ответственность.

1.4.1 HTML

Html-документы могут просматриваться различными типами браузеров. Когда документ создан с использованием html, веб-браузер может интерпретировать html для выделения различных элементов документа и первичной их обработки. Использование html позволяет форматировать документы для их представления с использованием шрифтов, линий и других графических элементов на любой системе, их просматривающей. Язык HTML имеет собственный набор символов, с помощью которых Web-браузеры отображают страницу. Эти символы, называемые дескрипторами, включают в себя элементы, необходимые для создания гиперссылок. Одной из

отличительных особенностей HTML-документов является то, что сам документ содержит только текст, а все остальные объекты встраиваются в документ в момент его отображения Браузером с помощью специальных тэгов и хранятся отдельно. При сохранении HTML-файла в месте размещения документа создается папка, в которую помещаются сопутствующие ему графические элементы оформления.

1.4.2 PHP

PHP - это распространенный язык программирования общего назначения с открытым исходным кодом. PHP специально сконструирован для веб-разработок и его код может внедряться непосредственно в HTML.

PHP отличается от JavaScript тем, что PHP-скрипты выполняются на сервере и генерируют HTML, который посылается клиенту. Если бы у вас на сервере был размещен скрипт, подобный вышеприведенному, клиент получил бы только результат его выполнения, но не смог бы выяснить, какой именно код его произвел.

В первую очередь PHP используется для создания скриптов, работающих на стороне сервера, для этого его, собственно, и придумали. PHP способен решать те же задачи, что и любые другие CGI-скрипты, в том числе обрабатывать данные html-форм, динамически генерировать html страницы и тому подобное. Но есть и другие области, где может использоваться PHP. Вторая область – это создание скриптов, выполняющихся в командной строке. То есть с помощью PHP можно создавать такие скрипты, которые будут исполняться, вне зависимости от web-сервера и браузера, на конкретной машине. И последняя область – это создание GUI-приложений (графических интерфейсов), выполняющихся на стороне клиента.

1.4.3 CSS

CMS – это очень широкое понятие, которое включает как очень простые (состоящие из нескольких скриптов), так и весьма сложные универсальные системы, предназначенные для решения самых разнообразных задач при создании сайтов. CMS – это настраиваемая система, предназначенная для простого и эффективного управления содержимым сайта.

CMS возникли из стандартизации основных элементов сайтов: структура, рубрикация, метод подачи материалов, на основе которой и были разработаны первые системы для построения и сопровождения сайтов. Системы CMS стали очень популярны, когда при разработке стала применяться модульная структура, при которой CMS модифицируется под те или иные направления сайтов. Эти

системы приобрели популярность в силу того, что за небольшие деньги клиентам предлагалась достаточно мощная функциональность.

Современная система управления контентом представляет собой конструктор, с помощью которого можно создавать и сопровождать свой сайт (обновлять и добавлять материалы, заводить новые рубрики). Любая CMS исходит из принципа, что большинство сайтов очень похожи по структуре и рубрикам, но отличаются наполнением и дизайном. Поэтому в каждую CMS входят определённые пункты, которые можно включить в сайт: новости, статьи, информация о фирме, контактные данные, прайс-лист и т. д., которыми можно управлять в разделе администрирования сайта.

1.4.4 JavaScript

JavaScript – это полноценный динамический язык программирования, который позволяет, применяя его к HTML-документу, обеспечивать интерактивность создаваемых web-страниц. Этот язык невероятно многофункционален. С помощью него можно как создавать какие-либо галереи просмотра изображений, изменяющиеся макеты, так и более сложные элементы: 2D и 3D графика, полномасштабные приложения с базами данных, а также игры различной сложности.

- JavaScript сам по себе довольно компактный, но очень гибкий. Разработчиками написано большое количество инструментов поверх основного языка JavaScript, которые дают доступ к огромному количеству дополнительных функций, стоит лишь приложить немного усилий. К ним относятся:

- программные интерфейсы приложения (API), встроенные в браузеры, обеспечивающие различные функциональные возможности, такие как динамическое создание HTML и установку CSS стилей, захват и манипуляция видеопотоком, работа с веб-камерой пользователя или генерация 3D графики и аудио-сэмплов;

- сторонние API позволяют разработчикам внедрять функциональность в свои сайты от других разработчиков, таких как Twitter или Facebook;

- также мы можем применить к нашему HTML сторонние фреймворки и библиотеки, что позволит нам ускорить создание сайтов и приложений.

1.4.5 MYSQL

MySQL - это самая популярная (а некоторые добавят, что еще и самая лучшая в мире) СУБД с открытым исходным кодом. На самом деле MySQL составляет все более значительную конкуренцию таким дорогостоящим

гигантам, как Oracle и Microsoft SQL Server. MySQL создана и до сих пор поддерживается шведской компанией MySQL AB.

MySQL - это реляционная система управления базами данных. С технической точки зрения MySQL - программа, управляющая файлами, которые составляют базу данных, но часто термин «база данных» (БД) применяется и к самой программе, и к этому набору файлов. БД - это просто совокупность взаимосвязанных данных (текстовых, числовых, двоичных), за хранение и организацию которых отвечает СУБД. MySQL - это программа с открытым исходным кодом, равно как и PHP и некоторые версии системы UNIX& Это означает, что вы можете бесплатно устанавливать, запускать программу и модифицировать ее исходный код (который, как и ему саму, можно запустить из Сети). Подробнее об этом можно узнать в условиях лицензирования MySQL. MySQL состоит из нескольких частей, в том числе сервера MySQL, клиента MySQL и многочисленных служебных утилит для обслуживания базы данных и иных целей. Работу с MySQL можно вести, пользуясь многими распространенными языками программирования, включая PHP, Perl и Java.

1.4.6 Angular

Angular – это JavaScript-фреймворк, предназначенный для расширения браузерных приложений на основе шаблона MVC, а также позволяет делать процессы разработки и тестирования более простыми и быстрыми [8].

Angular предоставляет такую функциональность, как двустороннее связывание, позволяющее динамически изменять данные в одном месте интерфейса при изменении данных модели в другом, шаблоны, маршрутизация и так далее.

Одной из ключевых особенностей Angular является то, что он использует в качестве языка программирования TypeScript. Поэтому перед началом работы рекомендуется ознакомиться с основами данного языка.

Используя внедренные зависимости, данный фреймворк снижает нагрузку на сервер и web-приложение становится легче.

1.4.7 TypeScript

TypeScript — это надмножество JavaScript, то есть, любой код на JS является правильным с точки зрения TypeScript. Однако, TypeScript обладает некоторыми дополнительными возможностями, которые не входят в JavaScript. Среди них — строгая типизация (то есть, указание типа переменной при её объявлении, что позволяет сделать поведение кода более предсказуемым и упростить отладку), механизмы объектно-ориентированного программирования

и многое другое. Браузеры не поддерживают TypeScript напрямую, поэтому код на TS надо транспилировать в JavaScript.

TypeScript отличается от JavaScript возможностью явного статического назначения типов, поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках), а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использование кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции, и, возможно, ускорить выполнение программ. TypeScript применяется для разработки веб-приложений с использованием фреймворка Angular.

1.4.8 IntelliJ IDEA

IntelliJ IDEA — интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains.

Для оформления отчета по проделанной работе был использовал пакет программ Microsoft Office и веб-приложение draw.io, которое использовалось для построения структурных и других схем.

IntelliJ IDEA Community Edition - бесплатная версия самой умной среды разработки на основе открытого кода:

– умное автодополнение, инструменты для анализа качества кода, удобная навигация, расширенные рефакторинги и форматирование для Java, Groovy, Scala, Clojure и Erlang;

– профессиональный набор инструментов для разработки Android-приложений;

– поддержка JavaFX 2.0, интеграция с SceneBuilder; Дизайнер интерфейса для Swing;

– интеграция с автоматизированными инструментами сборки и управления проектом, включа Maven, Gradle, Ant и другими;

– инструменты для тестирования с поддержкой JUnit, TestNG, Spock, ScalaTest и spec2;

– интеграция с системами управления версиями, включая Git, Subversion, Mercurial и CSV.

1.4.9 PostgreSQL

PostgreSQL — это объектно-реляционная система управления базами данных, основанная на POSTGRES, Version 4.2 — программе, разработанной на факультете компьютерных наук Калифорнийского университета в Беркли. В

POSTGRES появилось множество новшеств, которые были реализованы в некоторых коммерческих СУБД гораздо позднее.

PostgreSQL — СУБД с открытым исходным кодом, основой которого был код, написанный в Беркли. Она поддерживает большую часть стандарта SQL и предлагает множество современных функций:

- сложные запросы;
- внешние ключи;
- триггеры;
- изменяемые представления;
- транзакционная целостность;
- многоверсионность.

Кроме того, пользователи могут всячески расширять возможности PostgreSQL, например, создавая свои:

- типы данных;
- функции;
- операторы;
- агрегатные функции;
- методы индексирования;
- процедурные языки.

1.5 Постановка цели и задач

Достижение цели не является одномоментным мероприятием. Достижение цели представляет собой решение совокупности задач, связанных между собой по смыслу и направленных для достижения единой цели.

Каждая из задач направлена на изучение или разработку каких-либо вещей в проекте. Таким образом, получается, что задачи – своеобразные «кирпичики» при выполнении которых достигается определенная цель.

Ниже приведены цель дипломного проекта и задачи, которые необходимо реализовать для достижения поставленной цели.

Целью дипломного проекта является разработка программного продукта, который позволит повысить продуктивность работы профессионалов.

Основные задачи дипломного проекта:

- изучение предметной области;
- определение основных требований к программному продукту;
- выбор и обоснование технологий для разработки программного продукта;
- разработка пользовательского веб-интерфейса;
- проектирование базы данных для хранения данных пользователей;
- добавление функциональности на стороне сервера: регистрация, аутентификация, чат, планировщик задач;

- обоснование экономической целесообразности разрабатываемого продукта;

- предложение мероприятий по улучшению условий труда в рамках реализуемого проекта.

2 Проектирование программного продукта

Данная глава представляет собой описание проектирования базы данных. Также раскроются понятия структура ПО и функциональной структуры приложения. Важно отметить, почему мы выбираем именно этот тип базы данных и каким образом этот выбор сможет повлиять на дальнейшее развитие приложения. В связи с этим необходимо предусмотреть все преимущества и недостатки выбранного типа базы данных, чтобы в дальнейшем учесть их при проектировании

Также описание изучения предметной области: какие таблицы и сущности должны присутствовать в базе данных для корректной работы приложения.

База данных должна содержать в себе четкие, упорядоченные таблицы, данные в которые разделены по группам и связаны между собой.

2.1 Проектирование структуры сайта

Структура программного обеспечения включает в себя любое программное обеспечение, которое было использовано при создании определенного приложения или пакета приложений.

Все программное обеспечение, используемое в создании ПО, можно разделить на 3 группы:

– системное ПО: программное обеспечение общего пользования. Сюда относятся операционные системы и их оболочки, различные специфические драйверы и утилиты, системы, которые предназначены для технического обслуживания машины разработчика;

– прикладное ПО: сюда относятся различные прикладные программы и пакеты этих программ. Например, это программы для работы с текстовой информацией, её оформлением, программы для просмотра веб-страниц или веб-браузеры и т. д.;

– инструментальное ПО: также называют системами программирования. Это инструменты, которые разработчик приложения использует для просмотра исходного кода программы, его редактирования и дальнейшего запуска. В настоящее время такие системы чаще всего называют интегрированными средами разработки, поскольку они включают в себя большой перечень инструментов для работы с различными языками программирования и СУБД.

Также существует 4 группа ПО, которую редко упоминают, — это базовое ПО. Базовым является то ПО, которое втроенно в аппаратное обеспечение компьютера. Чаще всего к данной категории относят базовую систему ввода-вывода (BIOS).

Исходя из схемы на рисунке 2.1.1, видно, что разработка приложения происходила под управлением операционной системы Linux Ubuntu последней версии. Данная версия операционной системы по умолчанию имеет графическую оболочку Gnome.

Далее приведена схема структуры разрабатываемого приложения:

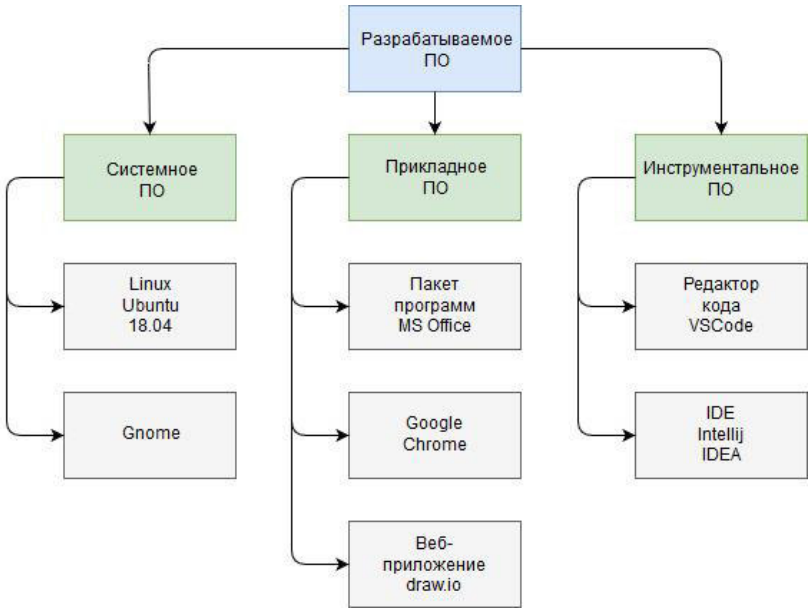

Рисунок 2.1.1 – Структура ПО

Такие продукты, как интегрированные среды разработки, позволяют сделать разработку программного обеспечения более простой и наглядной. Также они влияют на скорость разработки программного обеспечения и более простую реализацию каких-либо сложных моментов.

Структура разрабатываемого сайта не стандартная. Административная часть имеет большое количество страниц, в то время как клиентская часть содержит всего 5 вида:

- главная страница;
- о компании;
- FAQ;
- контакты;
- аккаунт.

Главная страница сайта содержит основную информацию о компании, их особенностях, как работает сайт и основные контактные данные компании.

Административная часть сайта позволяет редактировать информацию:

- о компании;
- о зарегистрированных фрилансеров;

- о проектах;
- выполнилось ли та или иная работа, есть ли притензии от компании;

Подробная структура страниц сайта представлена на рисунке 2.1.2.

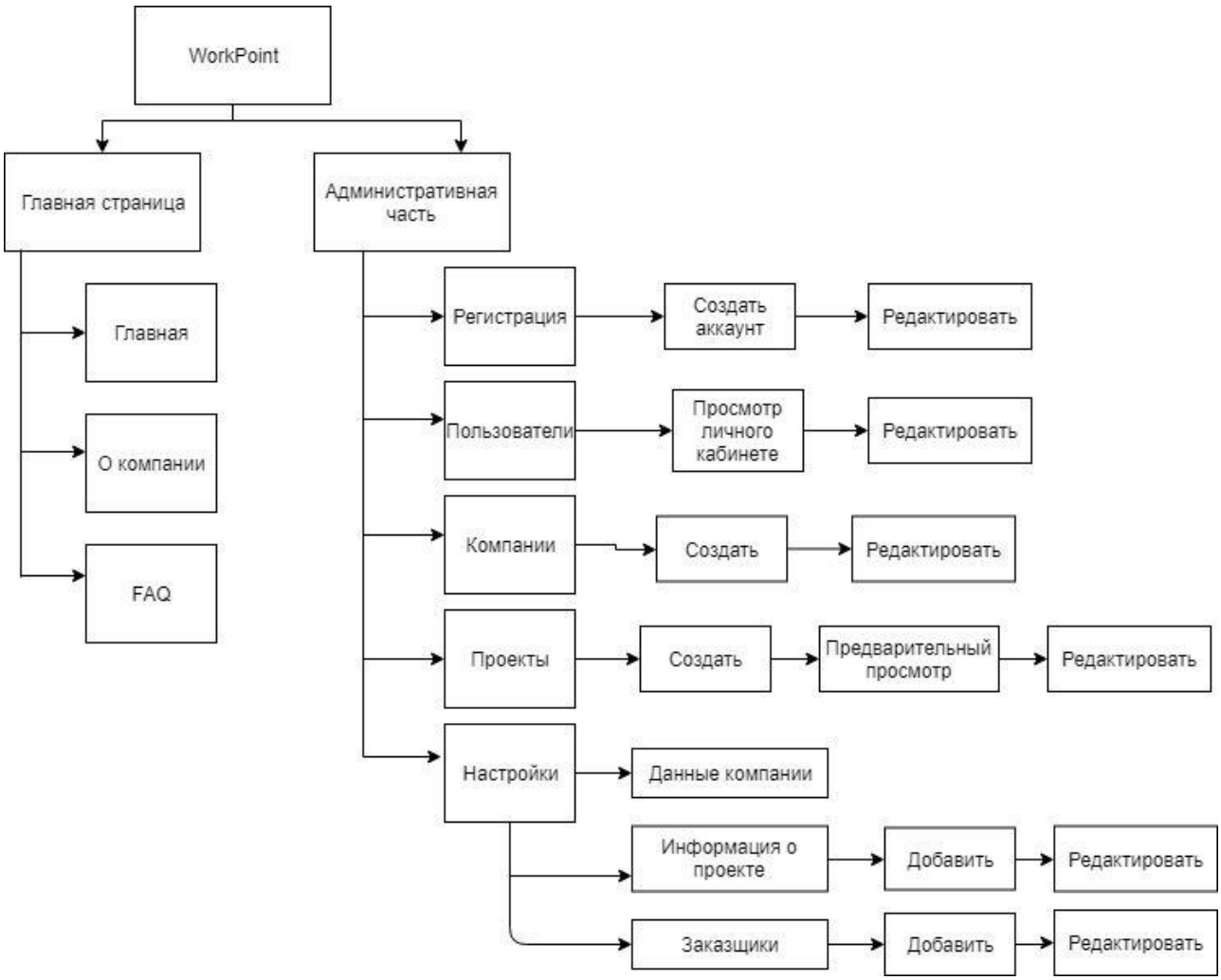

Рисунок 2.1.2 - Структура страниц сайта

2.2Разработка UML диаграммы

UML – унифицированный язык моделирования – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем. Словарь UML включает три вида строительных блоков:

- диаграммы;
- сущности;
- связи.

Сущности – это абстракции, которые являются основными элементами модели, связи соединяют их между собой, а диаграммы группируют представляющие интерес наборы сущностей. Диаграмма – это графическое представление набора элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей).

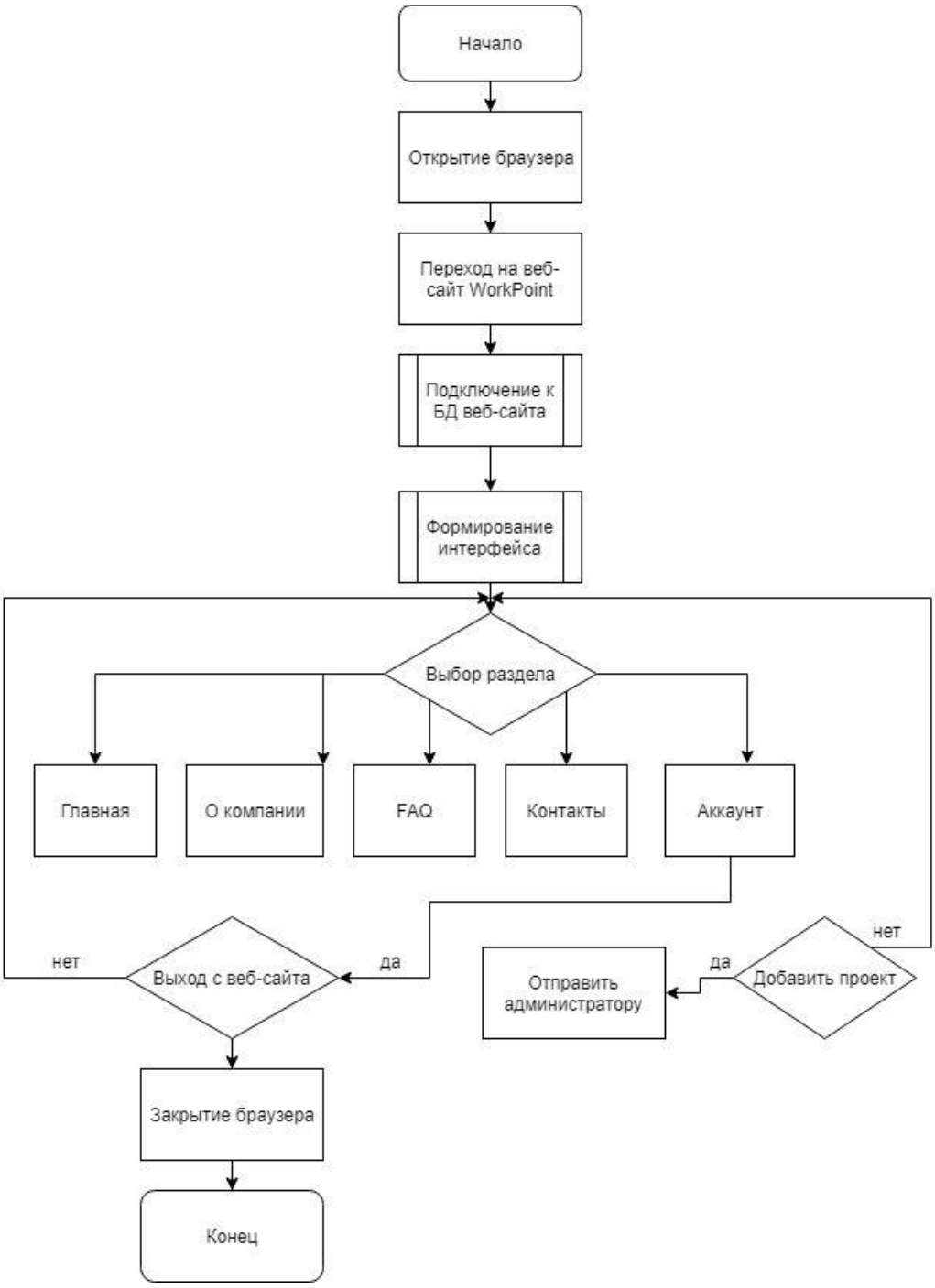

Рисунок 2.2 - UML диаграмма

2.3 Разработка макета сайта

Создание макета сайта, так называемого «скелета», подразумевает разработку структуры его страниц. После завершения данной стадии, в целом, становится понятно, как будет выглядеть сайт.

Создание макета имеет огромное значение с точки зрения юзабилити сайта, и просто необходима для разработки грамотной системы навигации по будущему сайту.

Макеты разрабатываются отдельно для каждой из основных страниц сайта. Далее устраняются все возникшие замечания и учитываются пожелания заказчика, в результате чего ему на утверждение предлагается окончательный вариант макета.

В моем случае были разработаны макеты двух основных страниц сайта. В них были учтены основные пожелания заказчика, как с точки зрения дизайна, так и с точки зрения функциональности. Так же были проанализированы данные из разделов требований к сайту данного. Мы добились максимально простого, легкого для восприятия и в то же время запоминающегося дизайна.

В результате проведенных работ мы получили макеты страниц («Главная страница», «Проекты») в трех вариантах отображения:

- для персональных компьютеров;
- для планшетных компьютеров;
- для мобильных телефонов;

Макет главной страницы сайта представлена на рисунке 2.3.1.

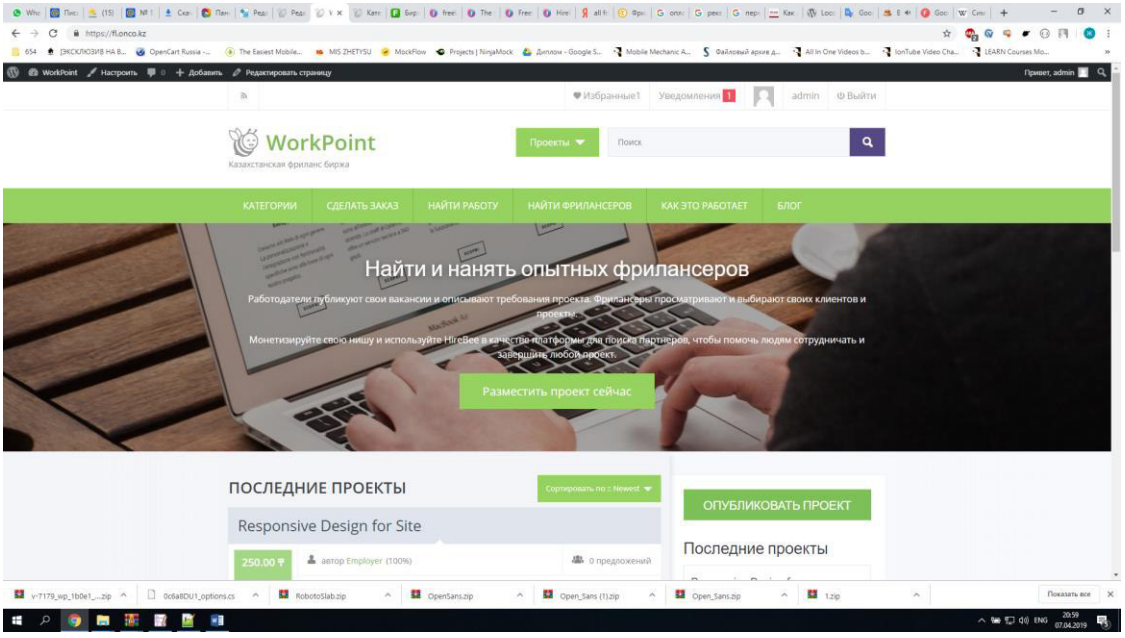

Рисунок 2.3.1 – Главная страница

Макет страницы заполнения проектов сайта представлена на рисунке 2.3.2.

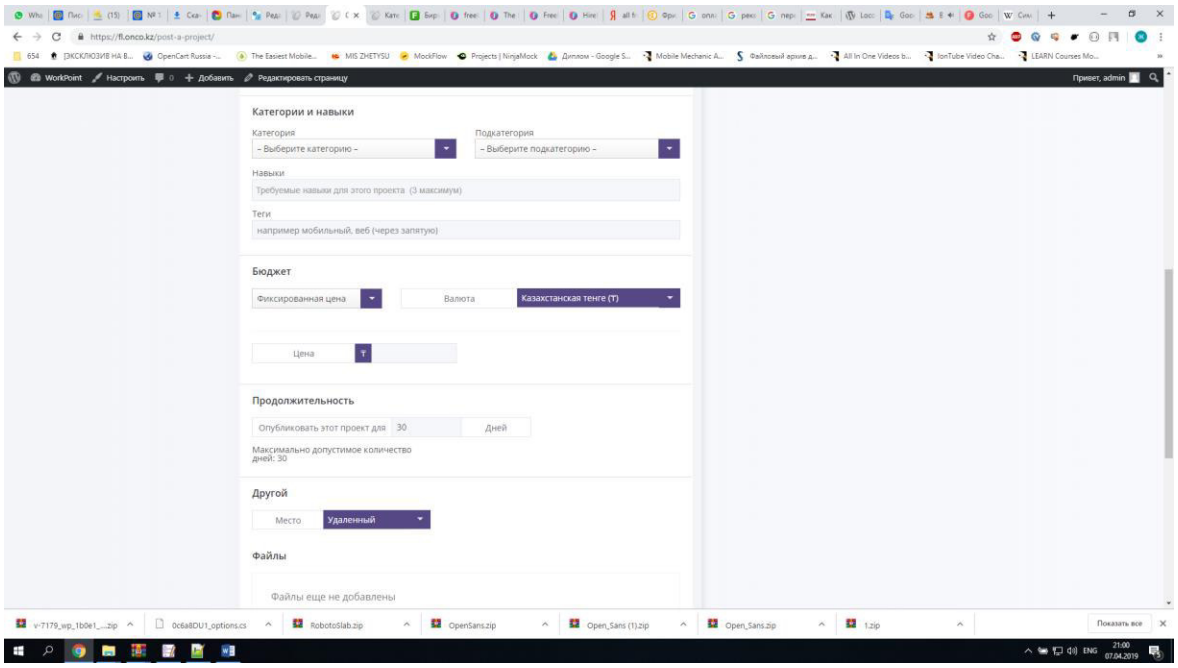

Рисунок 2.3.2 – Главная страница

2.4 Проектирование базы данных

Проектирование базы данных — это всегда достаточно сложный процесс, который требует не только внимательности разработчика, но и проявление его умственных способностей. От того насколько точно будет продумана логика приложения, а затем и перенесена в таблицы базы данных, зависит насколько просто потом будет разработчику получать данные из этой таблицы.

При разработке предлагаемого приложения возникла необходимость создать свою базу данных, поскольку пользователю необходимо хранить данные, загружаемые в приложение и после его выхода из аккаунта, и отключения браузера. База данных предназначается для того, чтобы хранить в ней все данные пользователя, его переписку с другими пользователями, задачи пользователя вместе с их статусами, которые определяют степень выполнения задачи в данный момент.

Система управления базами данных — это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

СУБД — это комплекс программ, позволяющих создать базу данных и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система

обеспечивает безопасность, надёжность хранения и целостность данных, а также предоставляет средства для администрирования БД.

Данная СУБД поддерживает большую часть официального стандарта SQL.

По мимо этого здесь реализован ряд важных современных функций для работы с базами данных: внешние ключи, сложные запросы, изменяемые представления, триггеры и т. п.

В качестве клиента используется специальное клиентское приложение, которое имеет возможность выполнять запросы в базе данных. Клиентские приложения очень разнообразны: различные текстовые утилиты, графические приложения, либо другой специализированный инструмент для работы с базой данных.

Для проектирования базы данных в дипломном проекте была выбрана СУБД PostgreSQL. PostgreSQL – это одна из самых популярных объектно-реляционных систем управления базами данных на сегодняшний день и заслужила симпатии программистов по всему миру. Система была разработана в Калифорнийском университете в 1996 году. Эта СУБД является одной из самых новаторских.

Решения, которые были разработаны для работы с данными в этой СУБД, гораздо позднее появились в аналогичных системах управления. К тому же данная система является системой с открытым исходным кодом, а это в первую очередь влияет на скорость ее развития, так как чем больше человек задействовано в разработке ПО, тем более неожиданные решения могут появиться в перечне функциональных возможностей системы.

В процессе выбора реляционной СУБД для работы с проектом уделялось внимание лицензии, которая предоставлялась разработчиками. PostgreSQL имеет свободную лицензию, что позволяет использовать данную СУБД для любых необходимых целей, а также позволяет видоизменять функционал приложения опираясь на свои нужды. К тому же из всех известных СУБД PostgreSQL является самой развитой системой управления. Также свободная лицензия – это большое преимущество перед другими, платными, СУБД, потому что стоимость лицензии других популярных СУБД может доходить до нескольких сотен тысяч долларов, что просто неприемлемо для небольших компаний.

Данная СУБД поддерживает большую часть официального стандарта SQL.

По мимо этого здесь реализован ряд важных современных функций для работы с базами данных: внешние ключи, сложные запросы, изменяемые представления, триггеры и т. п.

После выбора СУБД можно переходить к анализу предметной области и проектированию схемы базы данных.

Анализ предметной области представляет собой один из наиболее важных этапов в разработке приложения. Этот этап позволяет выявить все данные,

которые потребуются для корректного взаимодействия пользователя с приложением, а также учтут какие данные пользователь захочет хранить в базе данных.

Первое взаимодействие с системой: вход в приложение.

Для того, чтобы пользователь мог осуществить вход в систему ему необходимо иметь собственный идентификатор и пароль. В качестве идентификатора может выступать какой-либо уникальный номер, имя пользователя, или, например, номер телефона. В данном случае мы будем использовать номер телефона, потому что он уникален для каждого пользователя, а также потому, что при помощи вывода этого поля рядом с именем пользователя можно предоставить номер пользователя, который будет использован для срочной связи служащих посредством мобильного телефона.

Пароль – какой-либо набор символов, придуманный пользователем. Единственное особое условие этого поля состоит в том, что пароль пользователя хранится в базе данных в зашифрованном виде, и недоступен для просмотра системному администратору команды.

В том случае, если пользователь не зарегистрирован, то от формы входа он направится к форме регистрации. Здесь уже будут использоваться два ранее рассмотренных свойства: номер телефона, который служит идентификатором пользователя и пароль от аккаунта пользователя. Также здесь необходимо будет добавить поля имени и фамилии пользователя, для удобства поиска необходимого сотрудника, его должность.

В итоге подошел момент к созданию первой таблицы базы данных с названием «Пользователи». Эта таблица будет хранить в себе все данные относительно каждого пользователя в полях, названия которых были перечислены выше: имя, фамилия, почта, проект, оплата, пароль.

Запрос на создание таблицы в базе данных выглядит следующим образом:

```
CREATE TABLE `wp_commentmeta` (
  `meta_id` bigint(20) UNSIGNED NOT NULL,
  `comment_id` bigint(20) UNSIGNED NOT NULL DEFAULT '0',
  `meta_key` varchar(255) COLLATE utf8mb4_unicode_520_ci DEFAULT NULL,
  `meta_value` longtext COLLATE utf8mb4_unicode_520_ci
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_520_ci;
```

Рисунок 2.4.1 – Запрос для создания таблицы Пользователи

Результат запроса на все данные из таблицы, которая уже заполнена тестовыми значениями выглядит следующим образом:

```
INSERT INTO `wp_comments` (`comment_ID`, `comment_post_ID`, `comment_author`, `comment_
(1, 1, 'Автор комментария', 'wapuu@wordpress.example', 'https://wordpress.org/', '', '2
(9, 510, 'Mr WordPress', '', 'https://wordpress.org/', '', '2014-11-07 04:49:37', '2014
(12, 365, 'Jason', 'jasom@gmail.com', 'http://', '14.161.32.213', '2014-11-17 09:39:22'
(13, 368, 'Sharon', 'sharon@gmail.com', 'http://', '14.161.32.213', '2014-11-17 09:03:2
(14, 373, 'Albert', 'albert@gmail.com', 'http://', '14.161.32.213', '2014-11-17 09:09:4
(15, 376, 'Mary', 'mary@gmail.com', 'http://', '14.161.32.213', '2014-11-17 08:43:46',
(16, 380, 'Jesse', 'Jesse@gmail.com', 'http://', '14.161.32.213', '2014-11-17 09:43:36'
(17, 381, 'Stephan', 'stephan@gmail.com', 'http://', '14.161.32.213', '2014-11-17 09:47
(18, 382, 'Richard', 'Richard@gmail.com', 'http://', '14.161.32.213', '2014-11-17 09:53
(19, 54, 'George', 'George@gmail.com', '', '14.161.32.213', '2014-11-14 06:57:31', '201
(20, 201, 'Mark', 'Mark@gmail.com', 'http://', '14.161.32.213', '2014-11-17 09:20:44',
(21, 622, 'zendarol', 'zendarol@mail.ru', '', '', '2019-05-07 23:25:46', '2019-05-07 20
```

Рисунок 2.4.2 – Данные таблицы Пользователи

После входа в систему пользователь может добавить проект, пройти курсы.

Для того, чтобы в базе данных могли храниться все сообщения из всех переписок пользователей, необходимо, чтобы каждое сообщение хранило о себе следующие данные: отправитель сообщения, получатель сообщения, сам текст сообщения, идентификатор сообщения, а также время отправки сообщения. Далее рассматривается каждое из поле и его пригодность.

Поле, которое хранит в себе идентификатор отправителя сообщения хранится для того, чтобы у получателя корректно отображалось от кого пришло данное сообщение, а также при распределении сообщений между чатами будет сразу понятно, чату с каким пользователем принадлежит данное сообщение.

Запрос на создание таблицы в базе данных выглядит следующим образом:

```
CREATE TABLE `wp_options` (
  `option_id` bigint(20) UNSIGNED NOT NULL,
  `option_name` varchar(191) COLLATE utf8mb4_unicode_520_ci NOT NULL DEFAULT '',
  `option_value` longtext COLLATE utf8mb4_unicode_520_ci NOT NULL,
  `autoload` varchar(20) COLLATE utf8mb4_unicode_520_ci NOT NULL DEFAULT 'yes'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_520_ci;
```

Рисунок 2.4.3 – Запрос для создания таблицы Опции

Для того, чтобы в базе данных могли храниться все сообщения из всех переписок пользователей, необходимо, чтобы каждое сообщение хранило о себе следующие данные: отправитель сообщения, получатель сообщения, сам текст сообщения, идентификатор сообщения, а также время отправки сообщения. Далее рассматривается каждое из поле и его пригодность. Результат запроса на все данные из таблицы, которая уже заполнена тестовыми значениями выглядит следующим образом:

```
INSERT INTO `wp_options` (`option_id`, `option_name`, `option_value`, `autoload`) VALUES
(1, 'siteurl', 'http://freelance.loc', 'yes'),
(2, 'home', 'http://freelance.loc', 'yes'),
(3, 'blogname', 'Work Point', 'yes'),
(4, 'blogdescription', 'Первый казахстанский фриланс проект', 'yes'),
(5, 'users_can_register', '0', 'yes'),
(6, 'admin_email', 'admin@admin.ru', 'yes'),
(7, 'start_of_week', '1', 'yes'),
(8, 'use_balanceTags', '0', 'yes'),
(9, 'use_smilies', '1', 'yes'),
(10, 'require_name_email', '', 'yes'),
(11, 'comments_notify', '', 'yes'),
(12, 'posts_per_rss', '10', 'yes'),
(13, 'rss_use_excerpt', '0', 'yes'),
(14, 'mailserver_url', 'mail.example.com', 'yes'),
(15, 'mailserver_login', 'login@example.com', 'yes'),
(16, 'mailserver_pass', 'password', 'yes'),
(17, 'mailserver_port', '110', 'yes'),
(18, 'default_category', '1', 'yes'),
(19, 'default_comment_status', 'closed', 'yes'),
(20, 'default_ping_status', 'closed', 'yes'),
(21, 'default_pingback_flag', '', 'yes'),
(22, 'posts_per_page', '10', 'yes'),
(23, 'date_format', 'd.m.Y', 'yes'),
(24, 'time_format', 'H:i', 'yes'),
(25, 'links_updated_date_format', 'd.m.Y H:i', 'yes'),
(26, 'comment_moderation', '', 'yes'),
(27, 'moderation_notify', '', 'yes'),
(28, 'permalink_structure', '/%postname%/', 'yes'),
(29, 'rewrite_rules', 'a:275:{s:11:\"projects/?$\";s:27:\"index.php?post_type=project\";s:41:
(30, 'hack_file', '0', 'yes'),
(31, 'blog_charset', 'UTF-8', 'yes'),
(32, 'moderation_keys', '', 'no'),
```

Рисунок 2.4.4 – Данные таблицы Опции

Далее представлена структура базы данных, которая показывает связи между таблицами в базе данных, и обобщенный вид всех таблиц, включенных в базу данных.

Создание таких таблиц позволяет наглядно продемонстрировать не только саму структуру базы данных, но и описывает все данные, которые в дальнейшем должны использоваться в приложении. Каждая таблица на схеме выделена в отдельный прямоугольник, в котором указывается название таблицы и все ее поля. Ключевые поля в таблицах выделяются специальным символом слева от названия поля.

3 Разработка программного продукта

Клиент-серверное приложение представляет собой приложение, которое имеет две части: клиент и сервер. Такие приложения в основном отображаются и взаимодействуют с пользователем через веб-браузер.

Клиент, это та часть приложения, которая отображается пользователю, выполняется в веб-браузере и взаимодействует визуально с пользователем. На этой стороне работают такие языки разметки, стилей и программирования как HTML, CSS и JavaScript.

Серверная часть приложения не имеет собственного визуального представления и взаимодействует с пользователем через веб-браузер. Название этой части вытекает из того, что все действия выполняются на сервере — специальном компьютере, который может быть расположен как за тысячи километров от браузера, так и в непосредственной близости, вплоть до одной машины. На сервере обычно располагается база данных и оперируют такие языки как Java, PHP, C# и т. д. Данное приложение разрабатывается на языке программирования Java.

В начале разработки программного продукта необходимо создать проект, который будет включать в себя все необходимые средства для разработки данного проекта.

Для этого нужно зайти на сайт start.spring.io и выбрать все необходимые параметры для создания проекта. Сборщиком проектов выбран Gradle, язык программирования Java, версия Spring Boot 2.1.4. Далее идут метаданные, которые указываются для проекта. Артефакт проекта по сути является названием проекта.

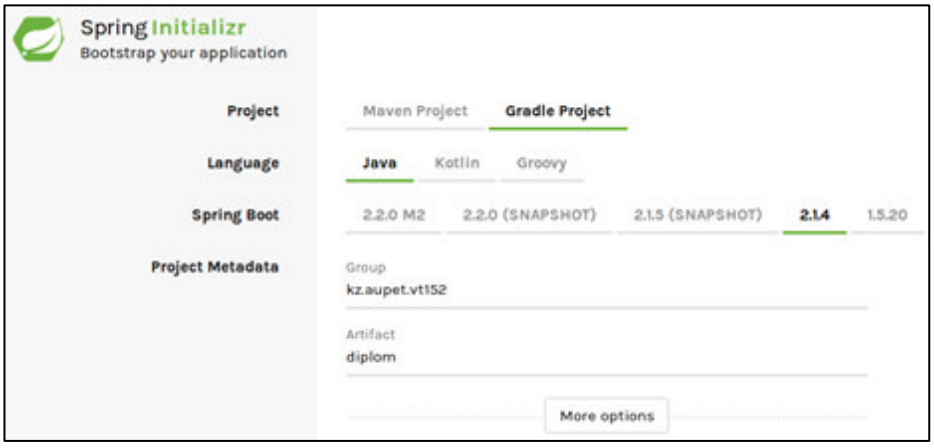

Рисунок 3.1 – Настройка сборки проекта на start.spring.io

После того, как мы заполнили первую часть формы переходим ниже и выбираем все технологии, которые необходимы для разработки:

Рисунок 3.2 – Выбор необходимых технологий

Подключив в проект все необходимые зависимости генерируется проект командой «alt+Enter» и скачивается на компьютер.

Вся дальнейшая разработка проводится в интегрированной среде разработки Intellij IDEA.

В первую очередь в новом проекте необходимо создать клиентскую часть проекта. Для этого прописывается команда:

```
Aspire-E1:-/git/diplom$ ng new client
```

Рисунок 3.3 – Команда создания клиентской части проекта

Этой командой в папке с данным проектом создается папка client, в которой будет храниться вся клиентская часть. Как раз в этот момент к работе подключается фреймворк Angular, который внутри папки client создает главный компонент приложения app.component. Далее в папке client подтягиваются все необходимые зависимости следующей командой:

```
Aspire-E1:-/git/diplom/client$ npm install
```

Рисунок 3.4 – Команда сборки необходимых зависимостей

Формирование контента на страницах разрабатываемого web-сайта происходит динамически с помощью функции include формируя шаблон страницы.

Конструкция include предназначена для включения файлов в код сценария PHP во время исполнения сценария PHP.

В отличие от конструкции require конструкция include позволяет включать файлы в код PHP скрипта во времявыполнения сценария. Синтаксис конструкции include выглядит следующим образом:

include имя_файла;

Так как используется фреймворк Angular, то все части проекта создавались в виде отдельных компонентов. Таким образом в ходе разработке получилось 4 компонентов, каждый из которых ответственен за свою роль в функционировании приложения:

- компонент main: компонент, который отвечает за формирование шапки сайта на каждой из страниц приложения и входа-выхода в страницу;
- компонент registration: компонент, отвечающий за регистрацию пользователя в приложении;
- компонент freelance: компонент, отвечающий за функционирование пользователя;
- компонент admin: компонент, отвечающий за функционирование заказщика.

В Angular главная страница или app.component создается лишь однажды на весь проект, другие компоненты лишь подставляются в него в зависимости от действий пользователя.

За корректное переключение между компонентами в браузере отвечает маршрутизация, специальная возможность фреймворка Angular которая позволяет сопоставлять запросы к приложению с определенными запросами внутри приложения.

Ключевым модулем для работы этой возможности является RouterModule, который располагается в @angular/router. Для этого необходимо записать RouterModule в список модулей файла app.module.ts.

Для указания путей перехода по разным страницам создается специальный файл app-routing.module.ts. В файле указывается массив объектов, каждый из которых содержит в себе путь к компоненту в браузере и название компонента, который будет загружаться по данному пути:

```
Const routes: Routes = [
{path: 'main', component: MainComponent, canActivate: [LoginGuard]},
{path: 'login', component: LoginPageComponent},
{path: 'registration', component: RegistrationComponent},
{path: 'freelance', component: FreelanceComponent}
];
```

Рисунок 3.5 – Роутинг между адресами и компонентами

3.1 Компонент main

Этот компонент представляет из себя верхнюю часть сайта, которая отображается в том или другом виде при открытии любого другого компонента. Таким образом, он является неотъемлемым и основным компонентом приложения. Структура этого компонента выглядит следующим образом:

Компонент имеет два основных состояния: вход выполнен и вход не выполнен.

При состоянии, когда вход не выполнен в левом верхнем углу отображаются кнопки входа и регистрации, и больше не отображается ничего:

Рисунок 3.1.1 – Верхняя часть приложения до входа в систему

В верхней части веб-сайта обычно помещается так называемая «шапка» - логотип организации в сочетании с текстом. Текстом может быть название этой организации или её слоган. Для сайта он выглядит следующим образом.

Рисунок 3.1.2 – Шапка сайта

В данном случае пользователь вводит данные в установленные для этого поля и нажимает на кнопку «Вход». В этот момент все данные, введенные пользователем, отправляются на проверку, и система принимает решение аутентифицировать этого пользователя или его данные не подходят ни к одному из зарегистрированных пользователей. Отправляются данные посредством функции login() при срабатывании обработчика формы.

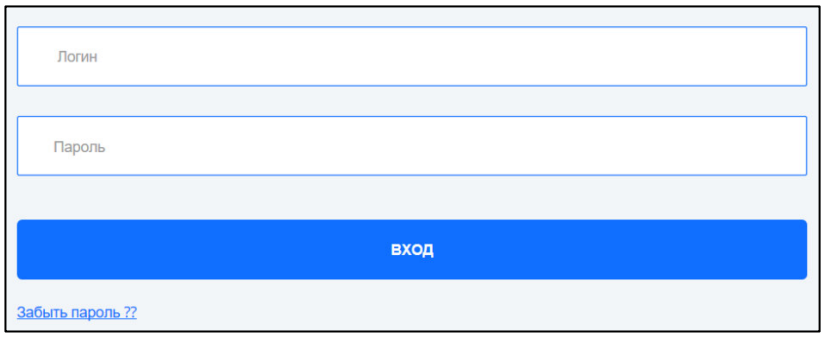

Рисунок 3.1.3 – Шапка сайта

Страница «Контакты» содержит информацию о местоположении учреждения, контактные данные и обратную связь. Также данный раздел имеет дополнительную навигацию в правой колонке. Для отображения местоположения на карте был выбран сервис Google Maps. Также можно будет поменять пароль.

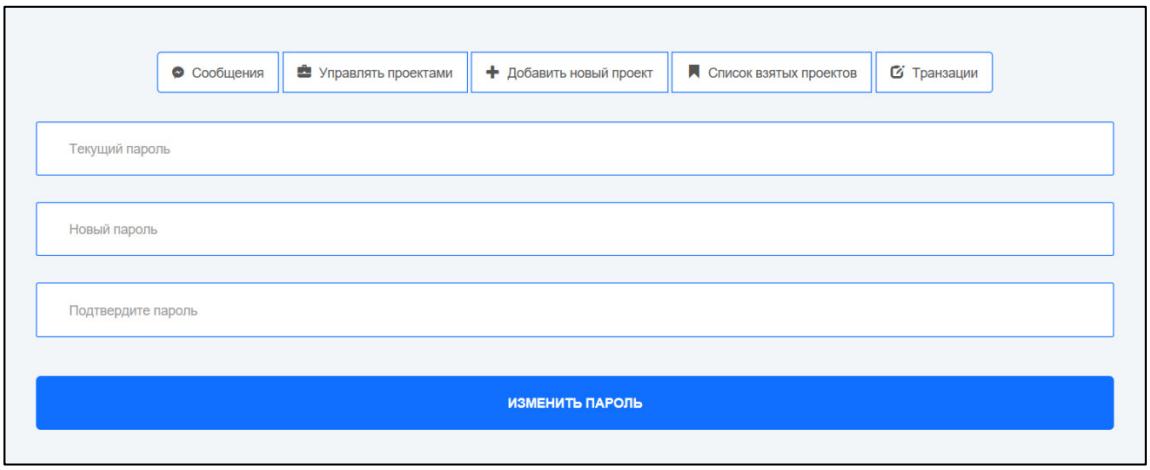

Рисунок 3.1.4 – Изменение пароля

3.2 Компонент registration

Компонент registration предназначен для регистрации пользователя, в том случае если он является новым сотрудником для компании. При регистрации ему необходимо ввести определенный минимум валидных для базы данных значений и тогда он сможет выполнить вход в приложение и начать его использование.

При обновлении тоже нет ничего сложного, важно отметить только то, что не нежно удалять версию Kunena для обновления до новой версии. Сам процесс обновления происходит автоматически и сохраняет все ваши существующие настройки форума, а также и все остальное: категорий, определенных вами Модераторов форума, все данные пользователей, темы, сообщения.

Если у клиента нет регистрации, он может легко зарегистрироваться заполнив анкету.

В контроллере выполняется метод registration(), который прежде, чем отправить на клиент результат о положительной регистрации проверяет два условия.

Первоначально этот метод шифрует значение пароля, чтобы в базе данных пароли хранились в закрытом доступе. Далее при помощи интерфейса UserDao и функции saveUser() происходит сохранение данных пользователя в базу данных.

Сохранение данных осуществляется при помощи фреймворка MyBatis, который при помощи аннотаций позволяет посылать запросы на выполнение в

базу данных. В данном случае используется аннотация @Insert и соответствующий ей insert-запрос, который и записывает данные о пользователе в необходимую таблицу. Любой пользователь может зарегистрироваться:

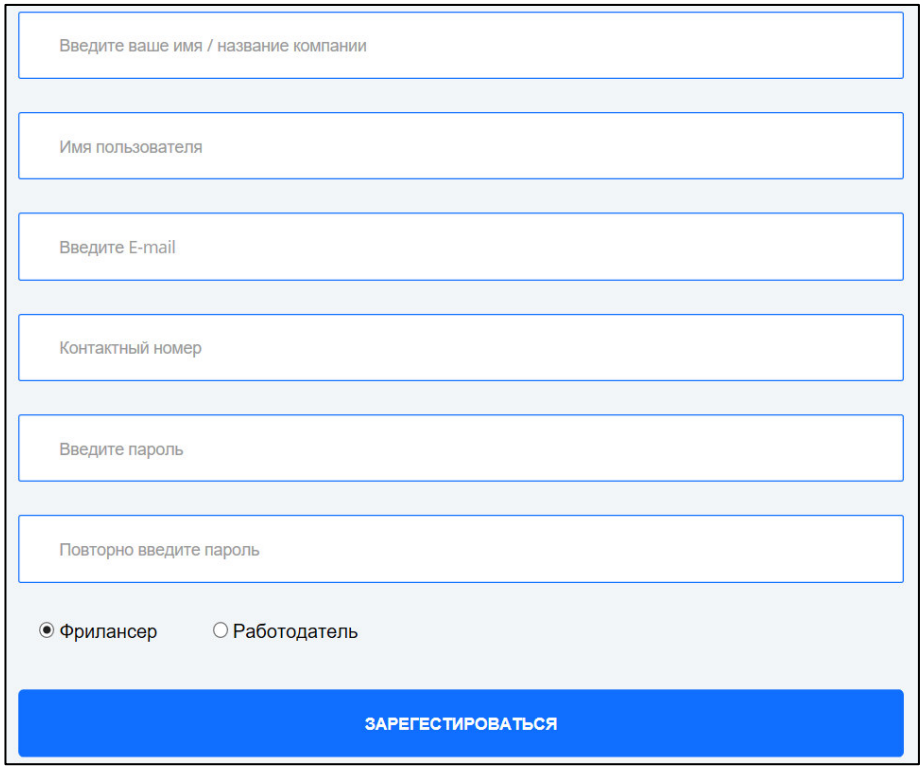

Рисунок 3.2.1 – Регистрация для входа

3.3 Компонент freelance

Компонент freelance представляет собой сервис, который предоставляет услуги для добавления проекта, принятия заказов.

Когда вход выполнен эти кнопки исчезают, на их место ставится иконка пользователя, с возможностью выхода, а также появляется навигация по приложению в виде ссылок на соббщения, редактирования резюме и мои проекты.

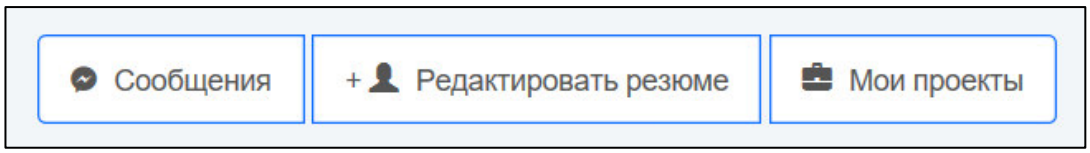

Рисунок 3.3.1 – Ссылки при входе

Фрилансер может просматривать свои проекты, редактировать их или же добавлять новые. Пример заполнения анкеты на рисунке 3.3.2.

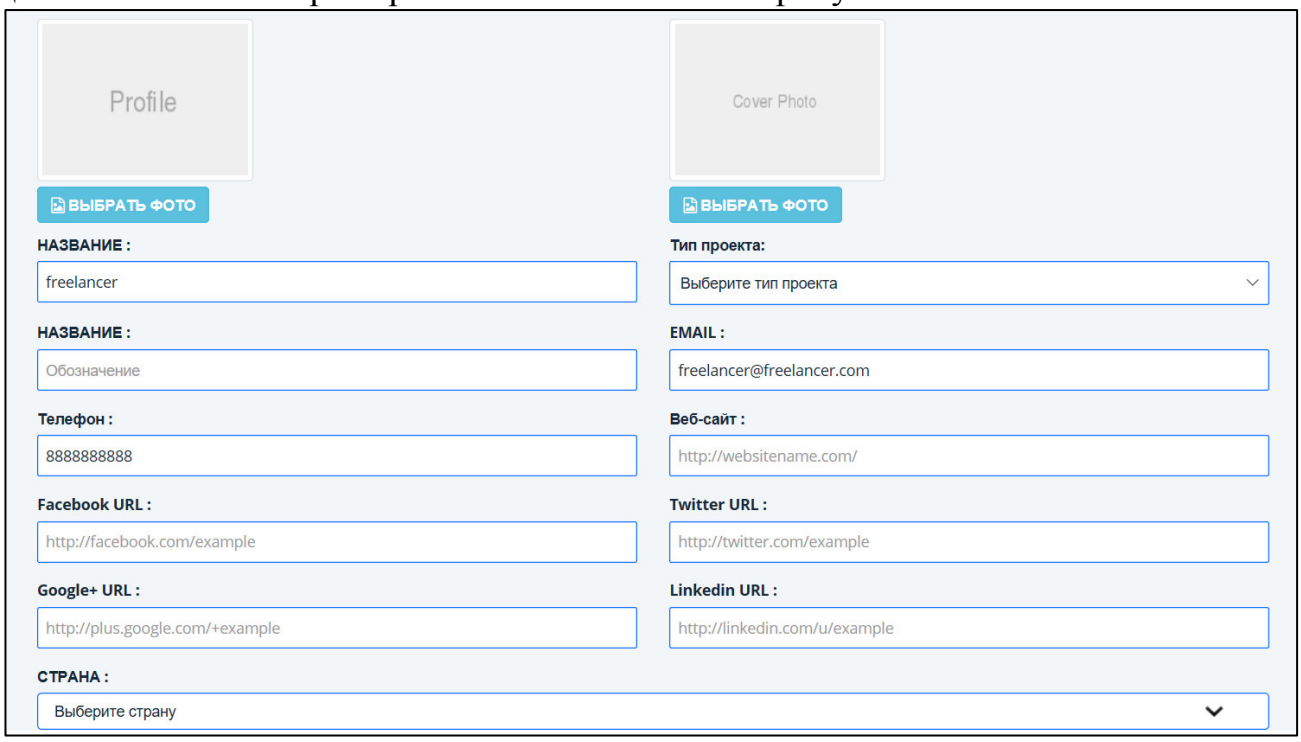

Рисунок 3.3.2 - Заполнения анкеты

Как заполнили анкету, можно будет просмотреть ее. Компонент login-page представляет собой форму входа пользователя в приложение. Данная форма предлагает пользователю ввести его номер телефона (идентификационное значение каждого пользователя) и пароль для того, чтобы осуществить вход в аккаунт.

В том случае, если пользователь не зарегистрирован в системе и является новым для нее сотрудником, то форма предлагает пользователю перейти на страницу регистрации.

Для добавления проекта необходимо будет заполнить полня:

- название;
- почта;
- телефон;
- типо проекта;
- страна
- тип проекта и описание;

Как заполнили данные по проекту можно наглядно просмотреть:

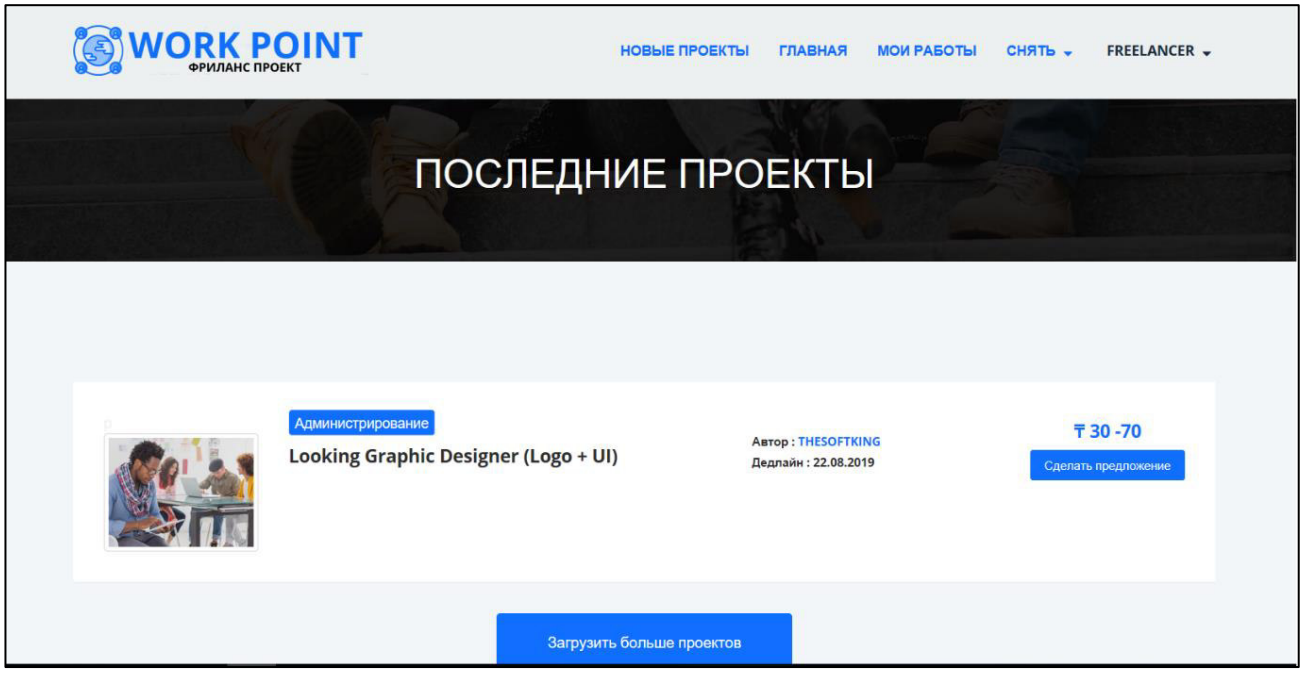

Рисунок 3.3.4 – Просмотр заполненной анкеты

Как фрилансер сделал работу, компания оплачивает и деньги приходят ему на счет. Он может несколькоми методами снять деньги:

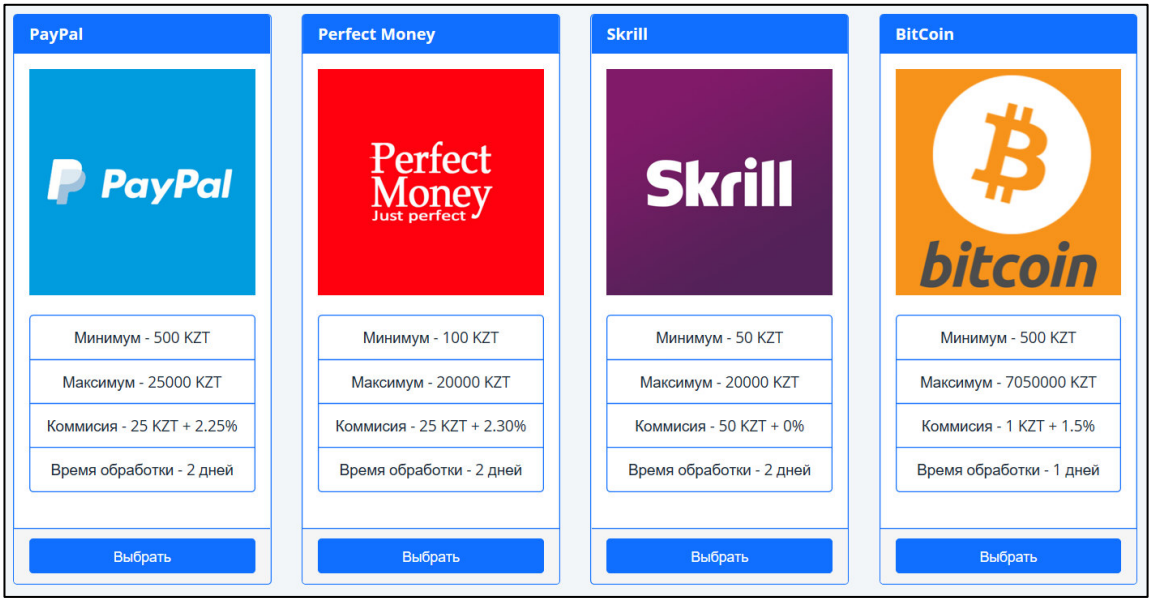

Рисунок 3.3.5 – Методы снятия денег

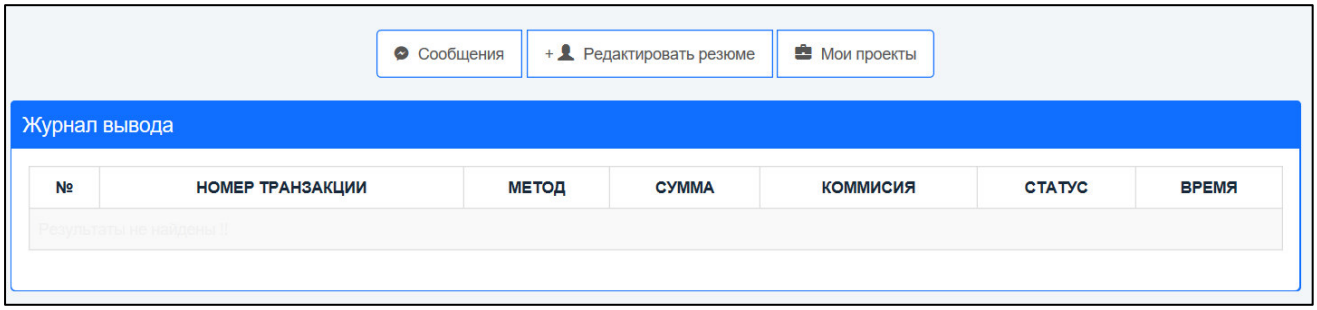

Рисунок 3.3.6 – Просмотр транзакции

3.4 Компонент admin

Компонент admin представляет собой сервис, который предоставляет возможность удаления/добавления проектов, изменения интерфейса, сброс пароля пользователей и компании. Администратор сайта — это пользователь, имеющий максимальные права доступа к управлению сайтом. В его права входит редактирование, удаление а так же добавление новой информации о товарах и услугах предоставляемых компанией.

После создания основной части web-сайта, была разработана административная часть. С помощью этой части сайт наполниться необходимым контентом. С целью обеспечения защищенности и разграничения доступа в системе перед началом работы необходимо пройти авторизацию, которая, как показано на рисунке 20, содержит два поля ввода и кнопку.

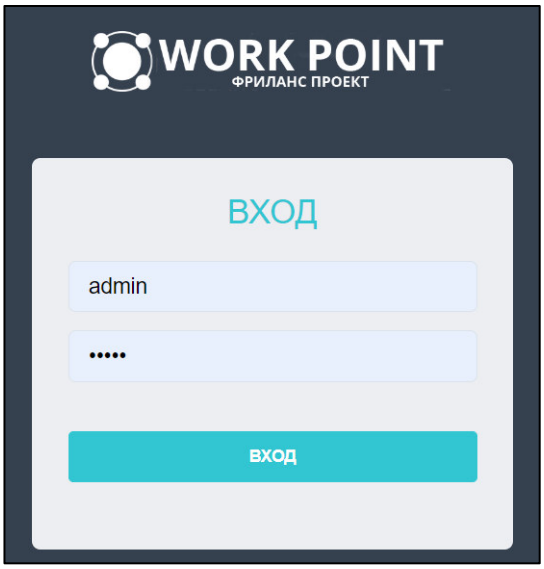

Рисунок 3.4.1 – Вход администратора

Администратор имеет право изменения информации предоставляемой на главной странице сайта, интерфейсом, редактирование категории проектов, все этапы отчетов и депозитами.

Окно системы делится на две основные части: перечень основных разделов и рабочая область. В главном меню пользователь выбирает административный раздел для второй части и выполняет действия с выбранным разделом Окно системы разделено на две основные части: список основных разделов и рабочую область. В главном меню пользователь выбирает административный раздел для работы, во второй части – осуществляет действия с выбранным разделом.

Группа разделов «Управление» содержит команды управления административной системой:

- статистика;
- пользователи;
- настройки;
- информация о сайте.

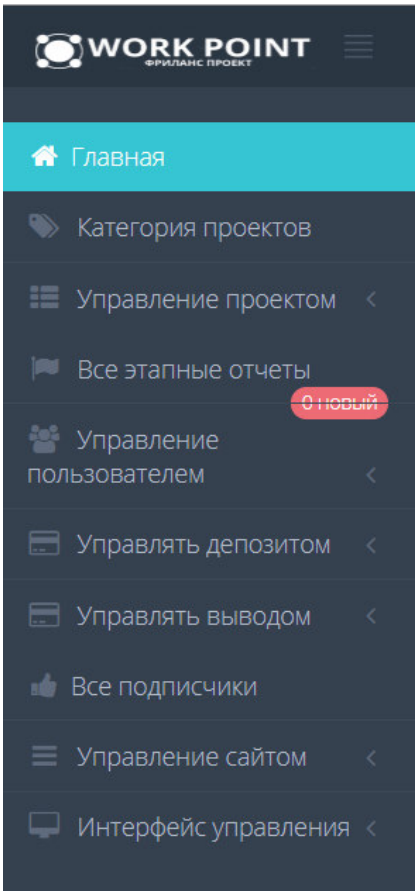

Рисунок 3.4.2 – Управление сайтом для администратора

Оформление рабочей области зависит от того, над какими информационным блоком идет работа, однако существуют общие принципы организации пользовательского интерфейса для этой области.

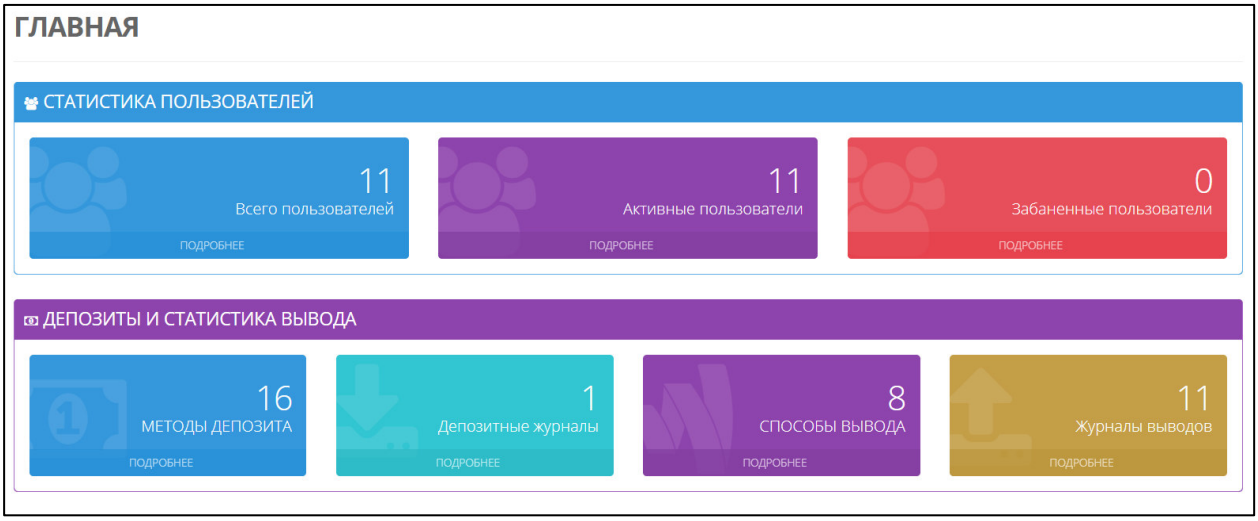

Рисунок 3.4.3 – Статистика сайта

Все необходимые функции для полноценной работы администратора, собраны в единую панель управления.

Все страницы административной панели содержат меню расположенное в верхней части и контента соответствующего для данного раздела. Главная страница сайта встречает авторизированного пользователя.

Рисунок 3.4.4 – Управление проектами

Навигационная схема Web-сайта зависит от его структуры и определяет то, как пользователь будет по нему перемещаться и получать доступ к информации, которую Вы представляете. Простота и удобство навигации является одним из важных факторов, определяющих посещаемость Web-сайта. Пользователи должны быстро и легко перейти на любую страницу Web-сайта, в том числе на начальную. Именно на этом этапе закладываются основные принципы работы сайта, его структура, формируется общее представление о дальнейшей работе над проектом. Также на этом этапе необходимо придумать названия разделов сайта, заголовки страниц, определить переходы между ними, то есть продумать логическую структуру размещения информации. В результате третьего этапа должна быть сформирована ясная и логическая структура размещения информации на сайте - ничто так не утомляет при поиске нужной информации, как плохо структурированные сайты.

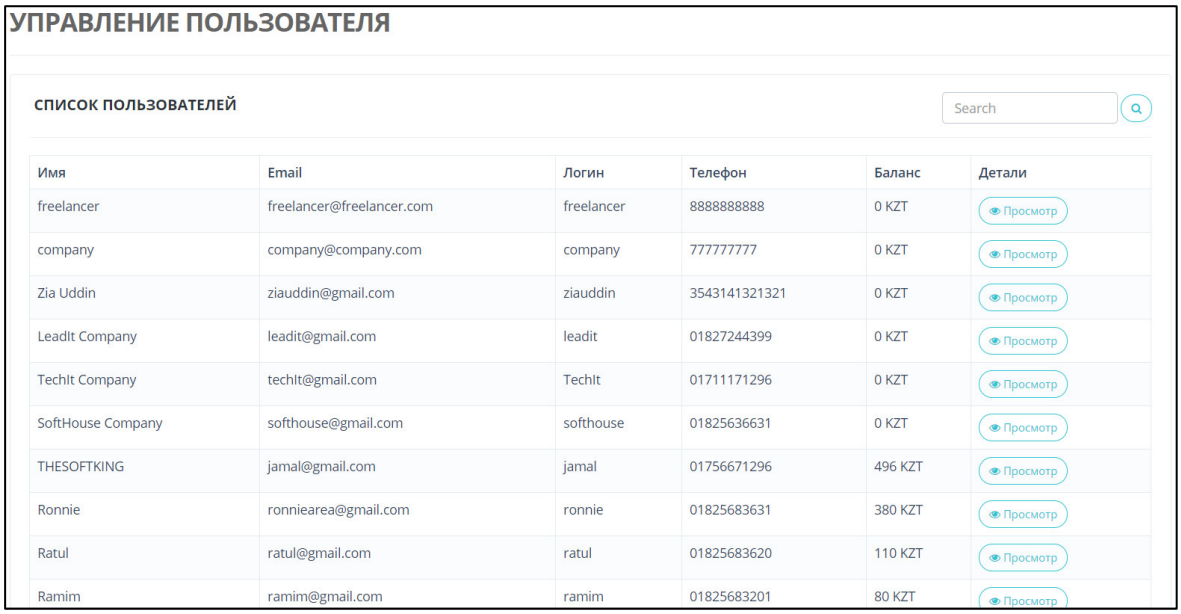

Рисунок 3.4.5 – Управление пользователями

Далее расположено диалоговое окно, в котором выводятся все сообщения между пользователями. Ниже диалогового окна располагается строка ввода сообщения, в которой пользователь набирает сообщение, чтобы затем отправить тому человеку, с которым у него открыт чат.

Когда пользователь напечатал сообщение для отправки и нажимает на кнопку отправки, либо клавишу Enter на клавиатуре своего компьютера, срабатывает функция sendMessage(), которое отвечает за отправку сообщения в базу данных и дальнейшее сохранение сообщения. Этот метод вызывает сервисную функцию sendMessage() и создает для передачи в параметры новый

объект типа Message, который в свою очередь содержит в себе всю информацию о новом сообщении пользователя.

Сама сервисная функция sendMessage() при помощи объекта stompClient отправляет на сервер сообщение в формате json.

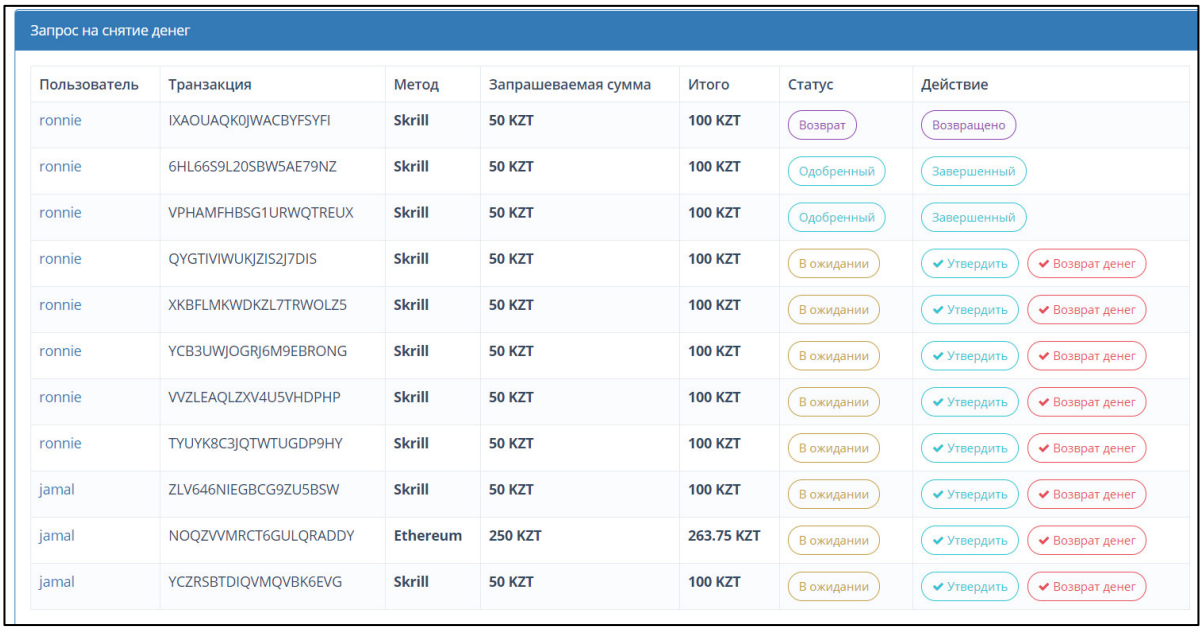

Рисунок 3.4.6 – Просмотр поступления оплаты

Так как чат, созданный в приложении, реализован при помощи технологии WebSocket, то далее происходит отправка сообщений только тем пользователям, которые прослушивают данные каналы связи. Таким образом, это будут два пользователя: отправитель сообщения и получатель сообщения. В дальнейшем на клиентской стороне приложения эти данные используются для корректного отображения сообщений в ленте чата пользователей.

Изначально, сразу после загрузки страницы слева, в списке задач, отображаются те задачи, которые запланированы на текущую дату, либо не отображается ничего, если на текущую дату отсутствуют запланированные задачи.

Таким образом, для того чтобы загрузка задач текущей даты осуществлялась сразу во время загрузки страницы, в файле planner.component.ts существует функция ngOnInit(), которая выполняет свои действия вместе с загрузкой страницы.

Эта функция загружает все статусы задач, которые показывают степень выполненности задачи, а также с помощью функции getTasks() получает все задачи за текущую дату.

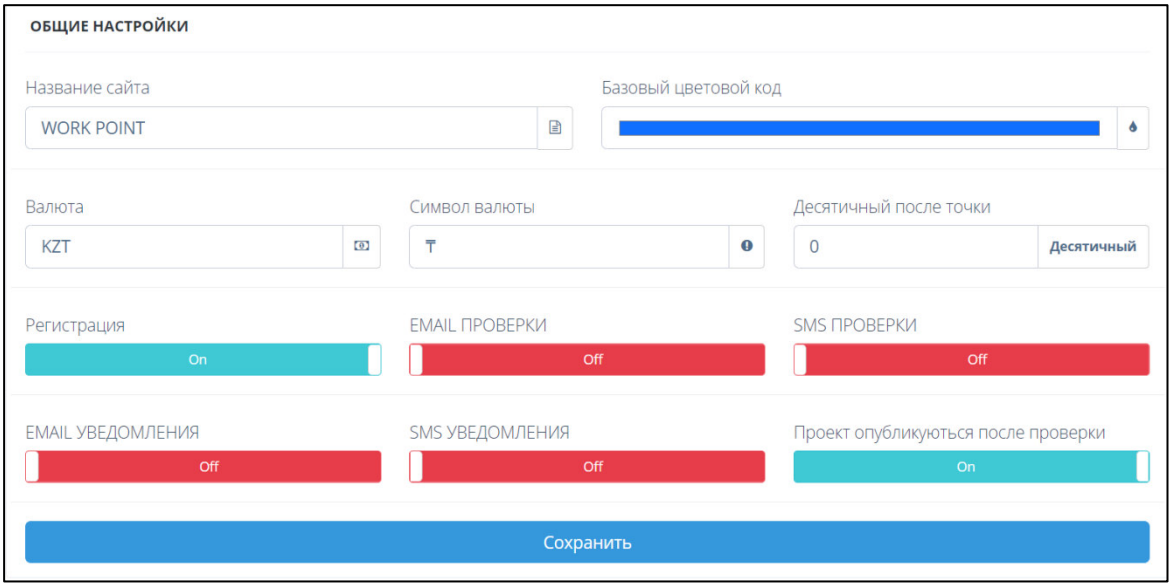

Рисунок 3.4.7 – Общие настройки

В ходе разработки было изучено большое множество технологий, которые позволяют разрабатывать современные и функциональные приложения для большого количества пользователей. Эти технологии относятся как к frontend-разработке, так и к backend-разработке и проектированию базы данных приложения.

Главной задачей, которая и была выполнена в ходе разработки, являлось создание приложения, которое может полноценно использоваться в реальных условиях труда и выполнять все положенные ему функции

По итогам разработки было получено функциональное приложение, которое позволяет пользователям оставаться на связи и планировать свой рабочий ритм в течение многих дней и недель. Все данные пользователей доступны лишь им самим и процесс использования приложения запускается только после регистрации каждого сотрудника.

4 Технико - экономическое обоснование дипломных работ, связанных с разработкой программного продукта (ПП)

В данной дипломной работе описан проект по разработке и проектированию веб-сайта для центра занятости «WorkPoint».
Целью данной работы является: создание онлайн сервис поиска частных специалистов для решения бытовых и бизнес-задач. Площадка, объединяющая заказчиков услуг, которым необходимо выполнить какую-либо работу, и компетентных исполнителей, ищущих подработку или дополнительный заработок.

В данном разделе дипломной работы производится расчет экономической составляющей по разработке веб-сайта отражающие временные, трудовые и финансовые затраты.

4.1 Трудоемкость разработки ПП

Для определения трудоемкости разработки программного продукта необходимо прежде всего составить перечень всех основных этапов и видов работ, которые должны быть выполнены в ходе разработки.

Трудоемкость программного продукта отражена в таблице 4.1.

Таблица 4.1 - Распределение работ по этапам и видам и оценка их трудоемкости

Этап разработки ПП	Вид работы на данном этапе	Трудоемкость разработки ПП, чел.× ч.
Формулировка цели для создания ПП	Выбор цели	1 ч
	Обоснование выбора разработки программного продукта	1,5 ч
Анализ рынка	Исследование рынка	5 ч
	Анализ существующих ПО	5 ч
Алгоритмизация	Описание алгоритмов и различных процессов, составление схемы	3,5 ч
Работа по созданию и управлению БД	Создание и доведение БД до рабочего состояния	40 ч
Разработка графического дизайна	Создание дизайна программного продукта	50 ч

Этап разработки ПП	Вид работы на данном этапе	Трудоемкость разработки ПП, чел.× ч.
Создание программы	Создание функциональной части ПП	160 ч
	Тестирование ПП	40 ч
Отладка программного продукта	Исправление сбоев в функциональной части	30 ч
	Оптимизация дизайна ПП	20 ч
Эксплуатация	Внедрение и использование готового продукта	
	Печать документации	1 ч
ИТОГО трудоемкость выполнения программного продукта		358 ч

Продолжение таблицы 4.1

4.2 Расчет затрат на разработку ПП

На данном этапе рассчитываются затраты на необходимые материальные ресурсы, которые в свою очередь, делятся на основные и вспомогательные затраты, затраты на оплату труда, затраты на расход электроэнергии, общие затраты на амортизационные отчисления и на социальный налог, а так же прочие расходы.

Расчет затрат на материальные ресурсы описаны в таблице 4.2, которая представлена ниже.

Таблица 4.2 - Затраты на материальные ресурсы

Наименование материального ресурса	Единица измерения	Количество израсходованного материала	Цена за единицу, тг	Сумма, тг
Бумага А4	Шт	1	2000	2000
Катридж	Шт	1	1200	1200

ИТОГО	3200,00

Определяем общую сумму затрат на различные материальные ресурсы (Зм), по формуле, которая представлена ниже:

$$З_м = \sum_{i=1}^{n} P_i \times Ц_i \ , \quad (4.1)$$

где P_i - расход i-го вида материального ресурса, натуральные единицы;

$Ц_i$ - цена за единицу i-го вида материального ресурса, тг;

i - вид материального ресурса;

n - количество видов материальных ресурсов.

$$З_{бумага} = 1 \times 2000 = 2000 \text{ тг}$$

$$З_{катридж} = 1 \times 1200 = 1200 \text{ тг}$$

$$З_{общие} = 2000 + 1200 = 3200 \text{ тг}$$

Расчет затрат на необходимое оборудование и программное обеспечение производится по форме, приведенной в таблице 4.3.

Таблица 4.3 – Расчет затрат на оборудование и ПО, необходимое для проекта

Наименование оборудования и ПО	Цена за ед. в тенге	Ед. измерения	Кол ичество	Сумма, тг
Ноутбук ASUS VivoBook 15 X540UA	350 000	штук	1	350 000
Беспроводная компьютерная мышь A4TECH N-708X	15 000	штук	1	15 000
Принтер KYOCERA FS-1040	42 000	штук	1	42 000
Смартфон Samsung	60 000	штук	1	60 000
ИТОГО				467 000,00

Определяем общую сумму затрат на оборудования и ПО ($З_{обор}$), по формуле, которая представлена ниже:

$$З_{обор} = \sum_{i=1}^{n} P_i \times Ц_i \ , \quad (4.2)$$

где P_i - расход i-го вида оборудования, натуральные единицы;

$Ц_i$ - цена за единицу i-го вида оборудования, тг;

i - вид оборудования;

n - количество видов оборудования.

$$З_{ноутбук} = 1 \times 350\,000 = 350\,000 \text{ тг}$$

$$З_{мышь} = 1 \times 15\,000 = 15\,000 \text{ тг}$$

$$З_{принтер} = 1 \times 42\,000 = 42\,000 \text{ тг}$$

$$З_{смартфон} = 1 \times 60\,000 = 60\,000 \text{ тг}$$

$$З_{общие} = 350\,000 + 15\,000 + 42\,000 + 60\,000 = 467\,000 \text{ тг}$$

Так как для разработки программного продукта используется техника, которая потребляет электроэнергию, то необходимо рассчитать затраты на электроэнергию, которые представлены в таблице 4.4.

Таблица 4.4 - Затраты на электроэнергию

Наименование оборудования	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования для разработки ПП, ч	Цена электроэнергии, тг/кВт*ч	Сумма , тг
Ноутбук ASUS VivoBook 15 X540UA	0,07	0,7	240	18,32	215,4
Смартфон Samsung	0,012	0,7	28	18,32	4,3
Принтер KYOCERA FS-1040	0,304	0,7	12	18,32	46,8
ИТОГО					266,50

Общая сумма затрат на электроэнергию ($З_Э$) рассчитывается по формуле:

$$З_э = \sum_{i=1}^{n} M_i \times K_i \times T_i \times Ц_i \quad (4.3)$$

где M_i - паспортная мощность i-го электрооборудования, кВт;

K_i - коэффициент использования мощности i-го электрооборудования (принимается K_i=0.7¸0.9);

T_i - время работы i-го оборудования за весь период разработки ПП ч;

Ц - цена электроэнергии, тг/кВт×ч;

i - вид электрооборудования;

n - количество электрооборудования.

$$З_{э\ ноутбук} = 0{,}07 \times 0{,}7 \times 240 \times 18{,}32 = 215{,}4 \text{ тенге}$$

$$З_{э\ smartphone} = 0{,}012 \times 0{,}7 \times 28 \times 18{,}32 = 4{,}3 \text{ тенге}$$

$$З_{э\ принтер} = 0{,}304 \times 0{,}7 \times 12 \times 18{,}32 = 46{,}8 \text{ тенге}$$

$$З_{э\ общие} = 215{,}4 + 4{,}3 + 46{,}8 = 266{,}5 \text{ тенге}$$

Для статьи «Затраты на оплату труда» необходимо включить расходы по оплате труда всех работников, которые вовлечен и заняты разработкой самого программного продукта. Затраты на оплату труда рассчитываются по форме, приведенной в таблице 4.5.

Таблица 4.5 - Затраты на оплату труда

Категория работника	Трудоемкость разработки ПП, чел.×ч	Часовая ставка, тг/ч	Сумма, тг
Программист	240	952,4	228 576
Руководитель проекта	10	1071,43	10 714
ИТОГО			239 290,00

Общая сумма затрат на оплату труда ($З_{ТР}$) определяется по формуле:

$$З_{тр} = \sum_{i=1}^{n} ЧС_i \times T_i \quad (4.4)$$

где $ЧС_i$ - часовая ставка i-го работника, тг;

T_i - трудоемкость разработки ПП, чел.×ч;

i - категория работника;

n - количество работников, занятых разработкой ПП.

Часовая ставка работника может быть рассчитана по формуле:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} \quad , \tag{4.5}$$

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} = \frac{160\,000}{168} = 925{,}4$$

$$ЧС_{рук} = \frac{ЗП_{рук}}{ФРВ_{рук}} = \frac{180\,000}{168} = 1071{,}43$$

ФРВ – фонд рабочего времени сотрудника, который составляет примерно 168 часов в месяц.

Трудоемкость разработки ПП определяется по данным таблицы 4.1.

В статью «Социальный налог» включается сумма, которая рассчитывается как 11% от затрат на оплату труда всех работников ($З_{тр}$), занятых разработкой ПП. При расчете необходимо учесть, что пенсионные отчисления (10% от $З_{тр}$) не облагаются социальным налогом (ставки указаны на 2019 год).

$$З_{по} = З_{тр} \times 0{,}1 = 239\,290 \times 0{,}1 = 23\,929$$

$$З_{тр}\,(\text{с уч п. о. }) = З_{тр} - З_{по} = 239\,290 - 23\,929 = 215\,361$$

$$Н_{с.прог} = З_{тр}(\text{с уч п. о. }) \times 0{,}11 = 215\,361 \times 0{,}11 = 23\,689{,}7$$

Расчет затрат на оплату труда для руководителя проекта.

$$З_{по} = З_{тр} \times 0{,}1 = 10\,710 \times 0{,}1 = 1\,071$$

$$З_{тр}\,(\text{с уч п. о. }) = З_{тр} - З_{по} = 10\,710 - 1\,071 = 9\,639$$

$$Н_{с.рук} = З_{тр}(с\ уч\ п.\ о.\ ) \times 0{,}11 = 9\ 639 \times 0{,}11 = 1\ 060{,}29$$

$$Н_{с.общ} = Н_{с.прог} + Н_{с.рук} = 23\ 689{,}7 + 1\ 060{,}29 = 24\ 749{,}99$$

В статью «Амортизация основных фондов» включается сумма амортизационных отчислений от стоимости оборудования и программного обеспечения (ПО), используемых при разработке ПП. Амортизационные отчисления приведены в таблице 4.6.

Таблица 4.6 - Амортизация основных фондов (ОФ)

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Эффективный фонд времени работы оборудования и ПО, ч/год	Время работы оборудования и ПО для разработки ПП, ч	Сумма, тг
Ноутбук ASUS VivoBook 15 X540UA	350 000	20	1920	240	8 750
Беспроводная компьютерная мышь A4TECH N-708X	15 000	33,3	1920	240	624,38
Принтер KYOCERA FS-1040	42 000	14,3	1920	10	31,28
Смартфон Samsung	60 000	20	1920	20	125
ИТОГО					9 530,66

Общая сумма амортизационных отчислений определяется по формуле:

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i \times Н_{Ai} \times Т_{НИРi}}{100 \times T_{Эфi}}, \quad (4.6)$$

где $Ф_i$ - стоимость i-го ОФ, тг;

$Н_{Ai}$ - годовая норма амортизации i-го ОФ, %;

$Т_{НИРi}$ - время работы i-го ОФ за весь период разработки ПП, ч;

$Т_{Эфi}$ - эффективный фонд времени работы i-го ОФ за год, ч/год; i - вид ОФ; n - количество ОФ.

При определении стоимости ОФ необходимо учесть также затраты на доставку и монтаж, установку ПО. Эти затраты могут быть приняты в размере 10-25 % от затрат на приобретение ОФ.

Годовые нормы амортизации ОФ принимаются по налоговому кодексу РК или определяются, исходя из возможного срока полезного использования ОФ:

$$\text{НАi} = \frac{100}{T_{Ni}} \quad , \tag{4.7}$$

$$\text{НАi} = \frac{100}{5} = 20\ \%$$

$$\text{НАi} = \frac{100}{3} = 14{,}3\ \%$$

$$\text{НАi} = \frac{100}{7} = 33{,}3\ \%$$

где T_{Ni} - возможный срок использования i-го ОФ, год;

Возможный срок полезного использования ОФ может быть принят от 3 до 10 лет.

$$З_{АМ2} = \frac{350\ 000 \times 33{,}3 \times 240}{100 \times 1920} = 8\ 750 \text{ тенге}$$

$$З_{АМ2} = \frac{15\ 000 \times 33{,}3 \times 240}{100 \times 1920} = 624{,}38 \text{ тенге}$$

$$З_{АМ3} = \frac{42\ 000 \times 14{,}3 \times 10}{100 \times 1920} = 31{,}28 \text{ тенге}$$

$$З_{АМ4} = \frac{60\ 000 \times 20 \times 20}{100 \times 1920} = 125 \text{ тенге}$$

$$З_{АМобщие} = 8\ 750 + 624{,}38 + 31{,}28 + 125 = 9\ 530{,}66 \text{ тенге}$$

В статью «Прочие затраты» включаются расходы на арендную плату, включая коммунальные платежи, затраты на лицензирование и сертификацию, расходы на рекламу, канцелярские и прочие хозяйственные расходы.

Таблица 4.6.1 – Арендная плата

Площадь помещения, $м^2$	Цена за 1 $м^2$, тг	Стоимость за 1 месяц, тг	Длительность аренды (в месяцах)	Сумма, тг
48	3500	168 000	1	168 000

Затраты на интернет приведены в таблице 6.2

Таблица 4.6.2 – Затраты на интернет

Цена за 1 месяц, тг	Количество месяцев	Сумма, тг
7 000	1	7 000

Определяем общую сумму затрат на прочие расходы ($З_{пр}$), по формуле, которая представлена ниже:

$$З_{пр} = З_{аренду} + З_{интернет}, \quad (4.8)$$

$$З_{пр} = 168\,000 + 7\,000 = 175\,000 \text{ тг}$$

На основании полученных данных по отдельным статьям составляется смета затрат на разработку ПП по форме, приведенной в таблице 4.7.

Таблица 4.7 - Смета затрат на разработку ПП

Статьи затрат	Сумма, тг	В %
Материальные затраты	3 200,00	0,34
Затраты на оборудования и ПО	467 000,00	50,714
Затраты на электроэнергию	266,50	0,29
Затраты на оплату труда	239 290,00	26,037
Отчисления на социальные нужды	24 749,99	2,693

Амортизация основных фондов	9 530,66	1,037
Прочие расходы	175 000,00	19,04
ИТОГО по смете	919 037, 15	100

Структура себестоимости разработки программного продукта представлена на рисунке 4.1.

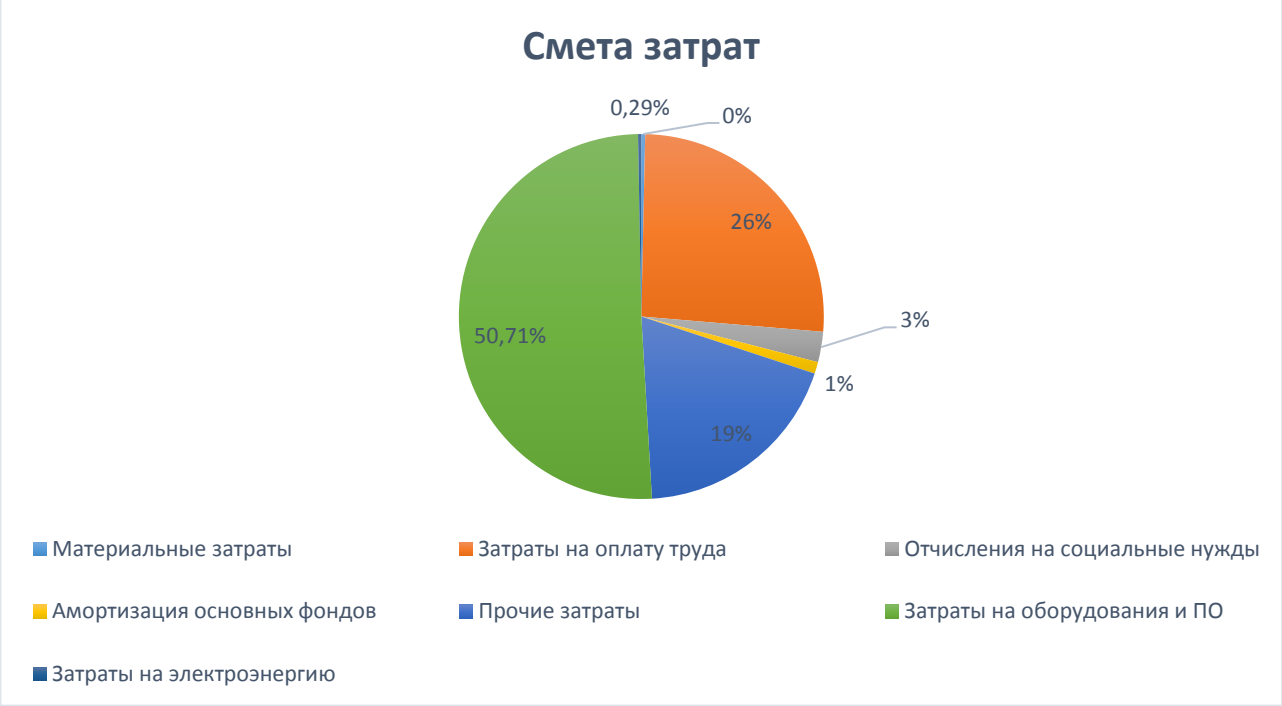

Рисунок 4.1 - Смета затрат ПП

4.3 Определение договорной цены ПП

Величина возможной (договорной) цены ПП должна устанавливаться с учетом эффективности, качества и сроков ее выполнения на уровне, отвечающем экономическим интересам заказчика (потребителя) и исполнителя. Договорная цена ($Ц_д$) для прикладных ПП рассчитывается по формуле:

$$Ц_д = З_{ПП} \times \left(1 + \frac{P}{100}\right) \text{, (8)}$$

где $З_{ПП}$ - затраты на разработку ПП (из таблицы 4), тг;

Р - средний уровень рентабельности ПП, который составляем 25%

$$Ц_д = З_{ПП} \times \left(1 + \frac{P}{100}\right) = 919\,037,15 \times (1 + 0,25) = 919\,037,15 + 229\,759,29 = 1148\,796,44 \text{ тенге}$$

Далее определяется цена реализации с учетом налога на добавленную стоимость (НДС), ставка НДС устанавливается законодательно Налоговым Кодексом РК. На 2019 год ставка НДС установлена в размере 12%. Цена реализации с учетом НДС рассчитывается по формуле:

$$Ц_р = Ц_д + Ц_д \times НДС \ , \quad (9)$$

$$Ц_р = 1148\,796,44 + 1148\,796,44 \times 0,12 = 1148\,796,44 + 137\,855,57 = 1286\,652,01 \text{ тенге}$$

Таким образом, себестоимость программного продукта составляет - 919 037, 15 тенге;

Прибыль составляет - 229 759,29 тенге;

Договорная цена - 1286 652,01.

5 Охрана труда и безопасность жизнедеятельности

Темой дипломного проекта было выбрано WEB-приложение, состоящее из двух частей, первая часть которого является этапом разработки технического задания, а ко второму этапу относится программная разработка. При разработке были соблюдены все правила техники безопасности, в связи с тем, что работа является сидячей, а именно разработка сайта, то основными пунктами были: освещение, вентиляция, а также работа с электрическими приборами и правила их эксплуатации.

Данный веб-сайт проектируется для центра занятости «WorkPoint». Помещение представляет собой комнату размерами 9х4х3,2. К списку оборудований относятся 4 ПК и 2 принтера. В помещении есть 3 окна по 2 $м^2$ и лампа мощностью 43 Вт. Данные компьютеры не создают большого шумового давления в пределах нормативов. В связи с тем, что вентиляция проведена не самым лучшим образом, я решила, что в данном разделе Безопасности Жизнедеятельности я буду рассчитывать вентиляцию.

Перед предстоящим расчетом нужно получить исходные данные исходя из того, какое мы используем помещение, количество персонала, а также оборудование, которое участвует непосредственно в разработке программного продукта, а именно сайта.

Город: Алматы;

Параметры помещения (Д х Ш х В), м: 9х3,2х4;

Данные по оборудованию: кол-во 6 шт.;

Мощность Роб, кВт/ч = 0,5;

КПД $\eta = 0,95$;

Данные по ист. света: мощность N ос. уст., $Вт/м^2 = 43$;

Вид ист. св.: люминесцентные лампы;

Число сотрудников: мужчины = 2;

Окна: кол-во 3;

Площадь 1 окна, $м^2 = 2$;

Расположение: Ю;

Вид: остекление в один-х метал. переплет, загрязнение умеренное;

Расчетное время суток, ч.: 11-12;

Температура в помещении, ^{0}С: летом 23, зимой 21;

Вид положения работы: сидячая работа.

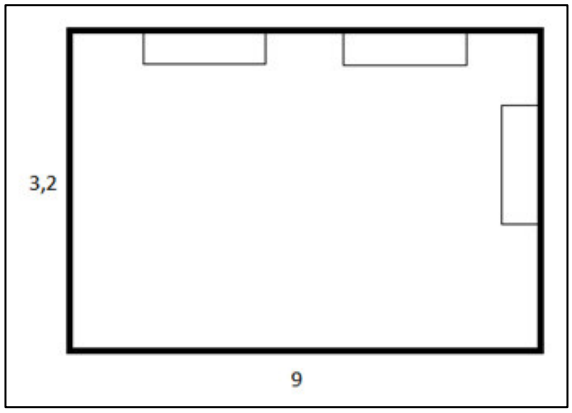

Рисунок 5.1 - Помещение

В данной части дипломного проекты была рассмотрена тема вентиляции помещения. Вентиляция является основным параметром безопасности жизнедеятельности во время труда и является обязательным параметром для расчета. В данной части дипломного проекта был произведен расчет следующих пунктов:

- расчет тепловых нагрузок внутри помещения;
- расчет наружных тепловых нагрузок помещения;
- расчет кол-во воздуха необходимого тому или иному помещению для подачи в данное помещение;
- по полученному расчету выбрать нужный кондиционер и показать все его технические характеристики в таблице;
- росле выбора кондиционера показать расположение его блоков внешнего и внутреннего.

5.1 Результаты проделанной работы

5.1.1 Рассчитать тепловые нагрузки в помещении: внутренние и наружные

Тепловые нагрузки непосредственно воздействуют на используемое нами помещение как внутренние тепловые нагрузки, так и внешние непосредственно климат, лучи солнца и общая температура погоды в городе в тот или иной сезон.

Наружные тепловые нагрузки

Данные нагрузки представлены следующими составляющими:

– теплопоступления или теплопотери в результате разности температур снаружи и внутри здания через стены, потолки, полы, окна и двери.

– разность температур снаружи здания и внутри него летом является положительной, в результате чего имеет место приток тепла снаружи во внутрь

помещения; и наоборот – зимой эта разность отрицательна и направление потока тепла меняется;

– теплопоступления от солнечного излучения через застекленные площади; данная нагрузка проявляется в форме ощущаемого тепла;

– теплопоступления от инфильтрации.

В зависимости от времени года и времени суток наружные тепловые нагрузки могут быть положительными.

Теплопоступления и теплопотери в результате разности температур определяется по формуле (5.1):

$$Q_{\text{огр}} = V_{\text{пом}} * X_o * \left(t_{Н\text{расч}} - t_{\text{Врасч}}\right) \quad (5.1)$$

где $V_{\text{пом}}$ – объем помещения, м^3

$$V_{\text{пом}} = 9 * 4 * 3{,}2 = 115{,}2 \text{ м}^3 ;$$

X_o – удельная тепловая характеристика, Вт/м^3 ^{o}С, X_o = 0,25;

$t_{Н\text{расч}}$ – наружная температура. Для холодного периода - средняя температура самого холодного месяца в 13 часов, для теплого периода - средняя температура самого жаркого месяца в 13 часов.

$t_{\text{Врасч}}$ – внутренняя температура, выбирается с учетом комфортных условий или технологических требований, предъявляемых к производственным процессам.

Для теплого времени года:

$t_{Н\text{расч}}$ = 30 ^{o}С

$t_{\text{Врасч}}$ = 27 ^{o}С

$$Q_{\text{огр}} = 115{,}2 \ * 0{,}25 * (30 - 27) = 115{,}2 \ * 0{,}25 * 3 = 86{,}4 \text{ Вт}$$

Для холодного времени года:

$t_{Н\text{расч}}$ = -16 ^{o}С

$t_{\text{Врасч}}$ = 18 ^{o}С

$$Q_{\text{огр}} = 115{,}2 \ * 0{,}25 * (-16 - 18) = 115{,}2 \ * 0{,}25 * (-34) = 979{,}2 \text{ Вт}$$

Избыточная теплота солнечного излучения в зависимости от типа стекла почти до 90% поглощается средой помещения, остальная часть отражается. Максимальная тепловая нагрузка достигается при максимальном уровне излучения, которое имеет прямую и рассеянную составляющие. Интенсивность излучения зависит от ширины местности, времени года и времени суток.

Теплопоступление от солнечного излучения через остекление определяется по формуле (5.2):

$$Q_P = (q^I F_O^I + q^{II} F_O^{II}) * \beta_{\text{с.з}} \qquad (5.2)$$

где q^I, q^{II} - тепловые потоки от прямой и рассеянной солнечной радиации, Вт/м^2;

F_O^I, F_O^{II} - площади светового проема, облучаемые и необлучаемые прямой солнечной радиации, м^3;

$\beta_{\text{с.з}}$ - коэффициент теплопропускания. $\beta_{\text{с.з}} = 0.20$;

При отсутствие наружных затеняющих в козырьков, ребер и т.д. для периода облучения остекления солнцем, когда его лучи проникают через окно в помещение $F_O^I = F_O;\ F_O^{II} = 0,(3)$:

$$Q_P = q^I F_O * \beta_{\text{с.з}} = \left(q_{\text{вп}} + q_{\text{вр}}\right) * K_1^C * K_2 * n * S_O \qquad (5.3)$$

$q_{\text{вп}},\ q_{\text{вр}}$- тепловые потоки от прямой рассеянной радиации, Вт/м^2.

Таблица 5.1. Поступление тепла $q_{\text{вп}}, q_{\text{вр}}$ от прямой (П) и рассеянной (Р) радиации в июле через вертикальное остекление.

Расчет граф широты	Истинное солнечное время / до полудня	Вертикальное освет до полудня Ю		Вертикальное освет до полудня С	
		П	Р	П	Р
48	11-12	317	88	0	59

Для широты в 48ºСШ до полудня в 11-12 ч. При расположении Ю: $q_{\text{вп}},\ q_{\text{вр}}$

$$q_{\text{вп}} = 317 \text{ Вт/м}^2$$

$$q_{\text{вр}} = 88 \text{ Вт/м}^2$$

F_O– площадь светового проема (n – число окон, S_O – площадь одного окна);

$$F_O = n * S_o = 2 * 3 = 6 \text{ м}^2$$

K_1 – коэффициент затемнения остекления переплетами (K_1^C – для облученных проемов).

$$K_1^C = 0.72$$

K_2 – коэфициент загрязнения остекления.

$$K_2 = 0.9$$

Тогда: $Q_P = (317 + 88) * 0.72 * 0.9 * 0.2 * 6 = 314{,}92$ Вт

Поступление солнечного излучения равно:

$$Q_P = 314{,}92 \text{ Вт}$$

Для широты в 48ºСШ до полудня в 11-12 ч. При расположении С: $q_{\text{вп}}$, $q_{\text{вр}}$

$$q_{\text{вп}} = 0 \text{ Вт/м}^2$$

$$q_{\text{вр}} = 59 \text{ Вт/м}^2$$

$$Q_P = (0 + 59) * 0.72 * 0.9 * 0.2 * 6 = 45{,}87 \text{ Вт}$$

Поступление солнечного излучения равно:

$$Q_P = 45{,}87 \text{ Вт}$$

5.1.2 Внутренние тепловые нагрузки.

Внутренние нагрузки в жилых, офисных или относящихся к сфере обслуживания помещениях слагаются в основном из тепла:

- выделяемого людьми;
- выделяемого лампами и осветительными, электробытовыми приборами;
- выделяемого компьютерами, печатающими устройствами, фотокопировальными машинами пр.

В производственных и технологических помещениях различного назначения дополнительными источниками тепловыделений могут быть: нагретое производственное оборудование, горячие материалы, в том числе

жидкости и различного рода полуфабрикаты, продукты сгорания и химических реакций.

Теплопоступления от людей зависит от интенсивности выполняемой работы и параметров окружающего воздуха. Тепло, выделяемое человеком, складывается из ощутимого (явного), то есть передаваемого в воздух помещения путем конвекции и лучеиспусканий, и скрытого тепла, затрачиваемого на испарение влаги с поверхности кожи и из легких.

Летом при $27\ ^oC \approx 28\ ^oC$ один мужчина выделяет явного тепла 51 Вт, а общего – 102 Вт (при работе стоя, легком движении). Женщина выделяет 85% от нормы тепловыделений взрослого мужчины. Выделение явного тепла составит:

$$Q_{\text{л}}^{\text{я}} = 51 * 3 = 153 \text{ Вт}$$

Выделение общего тепла:

$$Q_{\text{л}}^{\text{о}} = 102 * 3 = 306 \text{ Вт}$$

Зимой при $19\ ^oC \approx 20\ ^oC$ один мужчина выделяет явного тепла 82 Вт, а общего – 103 Вт. Тогда выделение явного тепла в помещении составит:

$$Q_{\text{л}}^{\text{я}} = 82 * 3 = 246 \text{ Вт}$$

Выделение общего тепла:

$$Q_{\text{л}}^{\text{о}} = 103 * 3 = 309 \text{ Вт}$$

Теплопоступление от осветительных приборов, оргтехники и оборудования рассчитывается следующим образом. Теплопоступление от ламп определяется по формуле (4):

$$Q_{\text{осв}} = \ \eta * N_{\text{осв}} * F_{\text{пол}} \qquad (5.4)$$

где η - коэффициент перехода электрической энергии в тепловую (для люминесцентных ламп $\eta = 0.45 – 0.5$);

$N_{\text{осв}}$ – установленная мощность ламп ($N_{\text{осв}} = 43$ Вт/м^2);

$F_{\text{пол}}$ – площадь пола ($F_{\text{пол}} = 9 * 3{,}2 = 28{,}8$ м^2);

Тогда:

$$Q_{\text{осв}} = 0{,}45 * 43 * 28{,}8 = 557{,}28 \text{ Вт}$$

Тепло, выделяемое производственным оборудованием, определяется по формуле (5.5):

$$Q_{об} = N_{уст} * K \quad (5.5)$$

где $N_{уст}$ – паспортная мощность оборудования ($N_{уст} = 1{,}1$ кВт/ч);

K - единиц оборудования ($K = 6$);

Тогда:

$$Q_{об} = 1{,}1 * 6 = 6{,}6 \text{ кВт}$$

Теплопритоки, возникающие за счет находящейся оргтехники – это 20% мощности оборудования:

$$Q_{орг} = 1{,}1 * 6 * 0{,}2 = 3{,}3 \text{ кВт}$$

5.2 Рассчитать количество воздуха, необходимое для подачи в помещение.

На основании выполненных расчетов составим баланс теплопоступлений в помещении:

Лето: $Q_{изб} = 314{,}92 + 153 + 557{,}28 + 3300 + 6600 + 84{,}6 = 11009{,}8$ Вт

Зима: $Q_{изб} = 45{,}87 + 243 + 557{,}28 + 3300 + 6600 + 979{,}2 = 11725{,}35$ Вт

Так как тепловой баланс для лета больше зимнего теплового баланса, то рассчитаем тепло-напряженность воздуха по формуле:

$$Q_H = \frac{Q_{изб} * 860}{V_{пом}} \quad (5.6)$$

$$Q_H = \frac{11009{,}8 \cdot 860}{115{,}2} = 82191{,}21 \text{ кал/м}^3 \approx 82{,}2 \text{ ккал/м}^3$$

При $Q_н$ >20ккал/м3 , Δt = 8 °С.

Определение количества воздуха, необходимое для поступления в помещение:

$$L = \frac{Q_{изб} * 860}{C * \Delta t * \gamma}$$

где $C = 0.24$ ккал/(кг°С) – теплоемкость воздуха,
$\gamma = 1.206$кг/м^3 – удельная масса приточного воздуха.

$$L = \frac{11725{,}35 \ * 860}{0.24 * 10^4 * 8 * 1.206} = 435{,}49 \text{ м}^3/\text{час}$$

Определение кратности воздухообмена:

$$n = \frac{L}{V_{\text{пом}}}$$

$$n = \frac{408{,}91}{115{,}2} = 3{,}54 \text{ час}^{-1}$$

5.3 По найденному значению количества воздуха подобрать соответствующую модель кондиционера.

Исходя из полученных данных, выберем кондиционер сплит-системы настенного типа.

5.3.1 Привести основные характеристики выбранного кондиционера.

Таблица 5.3 – Основные технические характеристики настенного кондиционера серии BALLU BCFB/OUT- 48HN1

Эл. питание В/Гн	Произв. по холоду, кВт	Потр. эл мощн, кВт	Потреблток, А	Произв. по теплу, кВт	Размер (внешн. блок) мм	Расход воздуха, м3 /ч	Размер (внутр. блок) мм
380/3/50	14,07	5,376	9,2	15,24	L 1167 H 900 B 340	1500	L 945 H 660 B 205

5.3.2 Привести схему расположения кондиционера в помещении и схему подачи воздуха.

Во внешнем блоке находятся компрессор, конденсатор и вентилятор. Внешний блок можно установить на стене здания, на крыше или на чердаке, в подсобном помещении или на балконе, то есть в таком месте, где горячий

конденсатор может продуваться атмосферным воздухом более низкой температуры.

Внутренний блок устанавливается непосредственно в кондиционируемом помещении и предназначен для охлаждения или нагревания воздуха, фильтрации его и создания необходимой подвижности воздуха в помещении. Внутренние блоки поддерживают заданную температуру, обеспечивают равномерное распределение воздуха в помещении и работают практически бесшумно (уровень шума 35-38 дБ).

Управление работой настенного кондиционера производится с дистанционного пульта, который позволяет задать режим работы кондиционера: обогрев, охлаждение, осушку, вентиляцию, ночной режим; задать требуемую температуру, которую должен поддерживать автоматически; выбрать режим работы вентилятора: настроить таймер, который включит или выключит кондиционер в заданное время; автоматически регулировать положение направляющих шторок и изменить таким образом направление воздушного потока.

Так как количества воздуха, необходимое для поступления в помещение равно 789,75 м3/час, то будет использован один кондиционер BALLU BCFB/OUT- 48HN1, который выдает необходимый нам расход воздуха.

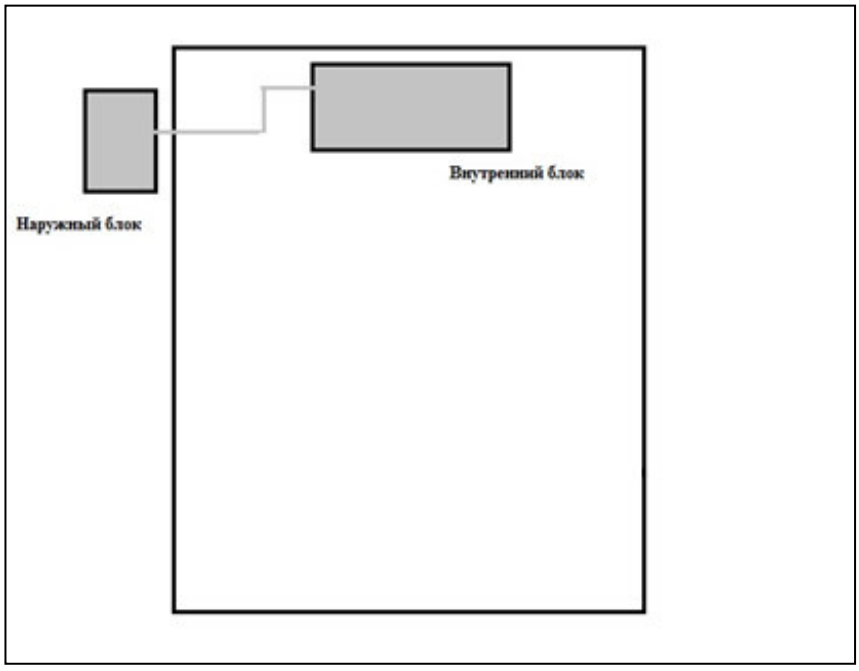

Рисунок 5 – Схема расположения кондиционеров в производственном помещении

В этой части дипломного проекта мы расчитали возмонжные тепловые нагрузки как и внутренние так и внешние для выбранного нами помещения. Расчитали необходимое количество подаваемого воздуха, после того как узнали количество воздуха выбрали подходящий кондиционер по его характеристикам и указали место расположения его блоков внешнего вентиляционного и внутреннего в помещении.

Тем самым можно сделать вывод, что безопасность жизнедеятельности учавствует во всех аспектах жизнедеятельности человека в любых профессиях и быте. На примере разработки моего проекта были соблюдены все нормы вентиляции, установлен кондционер, а так же все пункты с овещением. Так же установлено, что тепловые нагрузки зависят от размерности помещения и от количества людей находящимся в нем, а так же от таких внешних тепловых нагрузок как солнечные лучи и количество окон в помещении через которые они проходят. Большое влияние оказали погодные условия в городе проведения разработки.

Заключение

Web-сайт позволяет решать целый ряд разнообразных задач, служит визитной карточкой учреждения, позволяет реализовывать определенную функцию, привлекает дополнительное внимание целевой аудитории. Однако после разработки и размещения в Сети работа над сайтом вовсе не заканчивается. Анализ ситуации в Интернет показал, что сайты, которые вовремя не обновляются и не обеспечивают пользователей актуальной и своевременной информацией, быстро теряют свою аудиторию. Также довольно остро стоит проблема навигации по сайту: даже необходимая информация совершенно ни к чему на сайте, если ее невозможно найти. Главной задачей при создания сайта, в первую очередь следует обращать внимание на его структуру и информационную составляющую. Но есть и еще важнейший аспект, который нельзя обойти вниманием: эффективность поиска сайта в Интернет.

Целью данной дипломной работы является «проектирование и разработка веб-сайта WorkPoint для центра занятости». В результате выполнения дипломной работы мною были закреплены знания полученные в период обучения, а так же получены новые практические знания в сфере разработки интернет-проектов. Был разработан сайт на языке программирования PHP и системы управления базами данных MySQL. Макет сайта был реализован с помощью Wordpress, каскадных таблиц стилей CSS с элементами JavaScript.

Процесс проектирования сайта состоял из этапов:

- обзор и анализ предметной области;
- уточнение требований к разрабатываемому Интернет-приложению;
- разработка макета сайта;
- разработать ER диаграммы базы данных;
- создание базы данных;
- верстка шаблона сайта на основе разработанного макета;
- разработка интерфейсов для клиентской и административной частей сайта.

По окончанию работ мы получили готовый, оптимизированный с точки зрения юзабилити и сео веб-сайт, успешно выполняющий поставленные перед ним задачи.

Данный проект создан с целью помочь желающим найти подходящую удаленную работу, чтобы каждый самозанятый и фрилансер имел возможность встретиться здесь со своими коллегами, обсудить профессиональные вопросы, выбрать необходимый инструментарий и просто получить удовольствие от общения с друзьями, рассказать о своих достижениях, поделиться своими проблемами, посмотреть удаленная работа вакансии.

Список литературы

1 Савельева Н.В. Основы программирования на PHP [Электронный ресурс]: курс лекций. Учебное пособие для студентов вузов, обучающихся по специальностям в области информационных технологий/ Савельева Н.В.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2005.— 264 c.

2 Ларри Ульман Основы программирования на PHP [Электронный ресурс]/ Ларри Ульман— Электрон. текстовые данные.— М.: ДМК Пресс, 2007.— 286 c.

3 Ларри Ульман MySQL [Электронный ресурс]/ Ларри Ульман— Электрон. текстовые данные.— М.: ДМК Пресс, 2007.— 352 c.

4 Панфилов К.С. Создание веб-сайта от замысла до реализации [Электронный ресурс]/ Панфилов К.С.— Электрон. текстовые данные.— М.: ДМК Пресс, 2009.— 440 c.

5 Горнаков С.Г. Осваиваем популярные системы управления сайтом. CMS [Электронный ресурс]/ Горнаков С.Г.— Электрон. текстовые данные.— М.: ДМК Пресс, 2009.— 336 c.— Режим доступа: http://www.iprbookshop.ru/7926.— ЭБС «IPRbooks».

6 Коржинский С. Н. Настольная книга Web-мастера: эффективное применение HTML, CSS и JavaScript. – М.: КноРус, 2011. – 416с.

7 Бенкен Е.С. PHP, MySQL, XML: программирование для Интернета. – С. Пб.:BHV, 2008. –570 с.

8 Харрис Э. PHP/MySQL для начинающих. – С. Пб.:Издательство «КУДИЦ-Образ», 2005. –384 с.

9 Конверс Т.А. PHP 5 и MySQL. Разработка и внедрение. Библия пользователя. – М.: «Вильямс», 2006. –1216 с.

10 «HTML5 + CSS3. Основы современного веб-дизайна» / Кириченко А. В., Хрусталев А. А. – СПб: «Наука и Техника», 2018. – 352 с., ил.

11 «HTML и CSS. Разработка и дизайн веб-сайтов» /Джон Дакетт; [пер. с англ. М. А. Райтмана]. – Москва: Эксмо, 2019. -480 с.: ил. - (Мировой компьютерный бестселлер).

12 «Руководство по TypeScript» https://metanit.com/web/typescript/;

13 «Java 8. Полное Руководство» / Герберт Шилдт; [пер. с англ. И. Бернштейна]. – Москва: Вильямс, 2017. – 1376 с.

14 «Изучаем Java на примерах и задачах» / Р. В. Сеттер. – СПб: «Наука и Техника», 2016. – 240 с., ил.

15 «Spring 4 для профессионалов» / Крис Шеффер, Кларенс Хо, Роб Харроп; – Москва: Вильямс, 2015. – 752 с.

16 «Программирование на языке Java. Конспект лекций» / Гаврилов А. В., Клименков С. В., Харитонова А.Е., Цопа Е. А. – СПб: Университет ИТМО, 2015. – 126 с.

17 «Java. Методы программирования» / И. Н. Блинов, В. С. Романчик. – Минск: издательство «Четыре четверти», 2013. – 896 с.

18 «Программирование на Java» / Н. А. Вязовик – М.: Национальный Открытый Университет «ИНТУИТ», 2016.

19 «Самоучитель Java с примерами и программами. 3-е издание» / А. Н. Васильев. – СПб.: «Наука и техника», 2016. – 368 с.:ил.

20 «Введение в модель данных SQL» / С. Д. Кузнецов – М.: Национальный Открытый Университет «ИНТУИТ», 2016.

Приложение А
(обязательно)

Техническое задание

Наименование программного продукта (ПП): «WorkPoint».

Цель разработки ПО: создание веб-сайта со структурированием базы данных по данной тематике, функцией для реализации навигации поисковых запросов и доступом к Интернету для осуществления связи и обновления данных.

Идеология программного обеспечения: идеологией программного обеспечения является оценивание в выборе образовательного фонда в определённой области у пользователей.

Используемая разработки для клиентского интерфейса: HTML, CSS, JavaScript, Angula.

Выбор модели ПО: структурная модель – позволяет оформлять отдельные блоки программы (повторяющиеся и неповторяющиеся) в процедуры и функции, которые затем могут использоваться в других частях программы. Изменения в коде функций и процедур не влекут за собой изменение других частей кода программы.

Описание: модуль тест – главный модуль, с помощью которого непосредственно будет проходить выбор пользователя. Включение этого модуля происходит после регистрации пользователя. Модуль регистрация пользователя – модуль, в котором пользователь заполнит свои личные данные для регистрации перед открытием списка. После выбора его данные и результат отправленной заявки будет занесён в базу. Справочный модуль – справка по использованию ПП. Модуль информации об авторе – информация о создателях ПП. Модуль формирования результатов заявки – данный модуль формирует результат пройденного пользователем рассмотрения заявки. База пользователей и их результатов – база, содержащая данные обо всех пользователях и их результатов пройденного рассмотрения заявки.

Выбор архитектуры построения ПП: Model-View-Controller.

Выбор метода разработки структуры ПП: исходящий метод.

Выбор языка программирования: php.

Предполагаемая аудитория (тип, возраст и т.д.): от 20 лет.

Общий объем ПП, Мб: 25 Мб.

Приложение Б
(обязательное)

Листинг программы

index.blade.php

```
@extends('admin.layout.master')
@section('body')
   <div class="page-content-wrapper">
      <div class="page-content">
         <h3 class="page-title uppercase bold"> {{$page_title}}
         <div class="portlet light bordered">
            <div class="portlet-title">
               <div class="caption font-dark">
                  <i class="icon-settings font-dark"></i>
                  <span class="caption-subject bold uppercase">{{$page_title}}</span>
               <div class="tools">
<button class="btn btn-outline btn-circle btn-sm green pull-right" data-toggle="modal"
                     data-target="#addModal">
                  <i class="fa fa-plus"></i> Добавить новый
            <div class="portlet-body">
               <table class="table table-striped table-bordered table-hover order-column" >
                  <th> Название</th>
                  <th> Статус</th>
                  <th> Действие</th>
                @foreach($categories as $dep)
                     {{$dep->name or ''}}
                     @if($dep->status == 1)
                        <label class="label label-success"> Включить </label>
                     @else
                        <label class="label  label-danger">Отключить </label>
<button class="btn btn-outline btn-circle btn-sm green" data-toggle="modal"
                        data-target="#Modal{{$dep->id}}">
                        <i class="fa fa-pencil"></i> Редактировать
                  <!-- Modal for Edit button -->
<div class="modal fade" id="Modal{{$dep->id}}" tabindex="-1" role="dialog">
                     <div class="modal-content">
                        <form role="form" method="POST"
                           action="{{route('category.update', $dep->id)}}"
                           enctype="multipart/form-data" name="editForm">
                        {{ csrf_field() }}
                        {{method_field('put')}}
                        <div class="modal-header">
<button type="button" class="close" data-dismiss="modal" aria-hidden="true">
                           &times;
                        </button>
```

```
 <h4 class="modal-title" id="myModalLabel"><b class="abir_act"></b>
                                <i class="fa fa-pencil"></i> Редактировать категорию
                            <div class="modal-body">
                               <div class="form-group">
                                  <label for="name">Название :</label>
<input type="text" class="form-control" name="name" value="{{$dep->name}}"
id="name">
                               <div class="form-group">
                                  <label for="Status">Статус :</label>
                                  <select name="status"  class="form-control">
                                     <option value="">Выбрать статус</option>
<option value="1" @if($dep->status == 1) selected @endif>Включить</option>
 <option value="0" @if($dep->status == 0) selected @endif>Отключить</option>
</select>
                            <div class="modal-footer">
                                  <button type="submit" class="btn  btn-success ">Сохранить
                                  </button>
<button type="button" class="btn btn-default" data-dismiss="modal">Закрыть
                  @endforeach
   <!-- Modal for Edit button -->
   <div class="modal fade" id="addModal" tabindex="-1" role="dialog">
      <div class="modal-content">
         <form role="form" method="POST"
            <div class="modal-header">
<button type="button" class="close" data-dismiss="modal" aria-
hidden="true">&times;</button>
               <h4 class="modal-title" id="myModalLabel">
                  <i class="fa fa-plus"></i> Добавить категорию
            <div class="modal-body">
               <div class="form-group">
                  <label for="name">Название :</label>
                  <input type="text" class="form-control" name="name"  required>
               <div class="form-group">
                  <label for="Status">Статус :</label>
                  <select name="status"  class="form-control" required>
                     <option value="">Выберите статус</option>
                     <option value="1" >Включить</option>
                     <option value="0">Отключить</option>
            <div class="modal-footer">
               <button type="submit" class="btn  btn-success ">Сохранить</button>
               <button type="button" class="btn btn-default" data-
dismiss="modal">Закрыть</button>
   <div class="page-content-wrapper">
      <div class="page-content">
         <h3 class="page-title uppercase bold"> {{$page_title}}
```

```
            <div class="portlet light bordered">
                <div class="portlet-title">
                    <div class="caption font-dark">
                        <i class="icon-settings font-dark"></i>
                        <span class="caption-subject bold uppercase">{{$page_title}}</span>
                    <div class="tools"></div>
                <div class="portlet-body">
                    <table class="table table-striped table-bordered table-hover order-column" id="">
                            <th>Логин</th>
                            <th># транзации</th>
                            <th>Шлюх</th>
                            <th>Сумма</th>
                            <th>Статус</th>
                            <th>Дата</th>
                        @foreach($deposits as $data)
                                <a href="{{route('user.single', $data->user->id)}}">
                                    {{$data->user->username}}
                            <td>{{$data->trx}}</td>
                            <td>{{$data->gateway->name}}</td>
                            <td><strong>{{$data->amount}} {{$basic->currency}}</strong></td>
                                @if($data->status == 1)
                                <a href="" class="btn btn-outline btn-circle btn-sm green">
                                    <i class="fa fa-check"></i> Завершенный </a>
                                    @else
                                    <a href="" class="btn btn-outline btn-circle btn-sm red">
                                        <i class="fa fa-check"></i> В ожидании </a>
                                {{$data->updated_at}}
        <h3 class="page-title uppercase bold">  {{$page_title}}
<div class="portlet light bordered">
            <div class="portlet-title">
                <div class="caption font-dark">
                    <i class="icon-settings font-dark"></i>
                    <span class="caption-subject bold uppercase">Оплата {{$page_title}}</span>
                <div class="portlet-body">
                    <table class="table table-striped table-bordered table-hover">
                            <th>№</th>
                            <th>Имя шлюза</th>
                            <th>Имя для пользователя</th>
                            <th>Статус</th>
                            <th>Действие</th>
                        @foreach($gateways as $k=>$gateway)
                            <td>{{ ++$k }}</td>
                            <td>{{ $gateway->main_name }}</td>
                            <td>{{ $gateway->name }}</td>
                                @if($gateway->status == 1)
```

```
                        <a class="btn btn-outline btn-circle btn-sm green">Включить </a>
                        <a class="btn btn-outline btn-circle btn-sm red">Отключить </a>
                      <button class="btn btn-outline btn-circle btn-sm blue"
                          data-toggle="modal" data-target="#editModal{{$gateway->id}}"
                          data-act="Edit">
                        Редактировать
                      </button>
                  <!-- Modal for Edit button -->
<div class="modal fade editModal" id="editModal{{$gateway->id}}" tabindex="-1"
                       role="dialog">
                      <div class="modal-content">
                        <div class="modal-header">
                          <button type="button" class="btn btn-default btn-xs pull-right"
                              data-dismiss="modal">Закрыть
                          <h4 class="modal-title" id="myModalLabel">Редактировать
                            <strong>{{$gateway->name}}</strong></h4>
                        <form method="post" action="{{route('update.gateway')}}"
                            enctype="multipart/form-data">
                          {{ csrf_field() }}
<input class="form-control abir_id" value="{{$gateway->id}}" type="hidden"
                              name="id">
                          <div class="modal-body">
                            {{ Session::get('modal_message_error') }}
                            <div class="form-group">
                              <div class="fileinput fileinput-new" data-provides="fileinput">
                                <div class="fileinput-new thumbnail"
                                    style="width: 200px; height: 200px;">
<img src="{{ asset('assets/images/gateway') }}/{{$gateway->id}}.jpg"
                                      alt="*"/></div>
                                <div class="fileinput-preview fileinput-exists thumbnail"
                                    style="max-width: 200px; max-height: 200px;"></div>
                                <div>
                                  <span class="btn btn-success btn-file">
<span class="fileinput-new"> Изменить логотип </span>
                                    <span class="fileinput-exists"> Изменить </span>
                                    <input type="file" name="gateimg"> </span>
                                  <a href="javascript:;" class="btn red fileinput-exists"
                                    data-dismiss="fileinput"> Удалить </a>
                          <div class="form-group">
                            <div class="row">
                              <div class="col-md-6">
                                <h5><strong>Название шлюза</strong></h5>
                                <input type="text" value="{{$gateway->name}}"
                                    class="form-control" id="name" name="name">
                              <div class="col-md-6">
```

```
                                    <h5><strong>Ставка</strong></h5>
                                    <div class="input-group">
                                        <span class="input-group-addon">
                                            <strong>1 USD =</strong>
                                        </span>
                                        <input type="text" value="{{$gateway->rate}}"
                                               class="form-control" id="rate" name="rate">
                                        <span class="input-group-addon">
                                            <strong> {{ $basic->currency }}</strong>
                                <div class="form-group">
                                    <div class="row">
                                        <div class="col-md-6">
                                            <div class="panel panel-primary text-center">
                                                <div class="panel-heading">
                                                    Лимит депозита
                                                <div class="panel-body">
 <h5 for="minamo"><strong>Минимальная сумма</strong></h5>
                                                <div class="input-group">
<input type="text" value="{{$gateway->minamo}}"
                                                       class="form-control" id="minamo"
                                                       name="minamo">
                                                    <span class="input-group-addon">
<strong>{{ $basic->currency }}</strong>
<h5 for="maxamo"><strong>Максимальная сумма</strong></h5>
                                                <div class="input-group">
<input type="text" value="{{$gateway->maxamo}}"
                                                       class="form-control" id="maxamo"
                                                       name="maxamo">
                                                    <span class="input-group-addon">
                                                        <strong>{{ $basic->currency }}</strong>
                                        <div class="col-md-6">
                                            <div class="panel panel-primary text-center">
                                                <div class="panel-heading">
                                                    Коммисия депозита
                                                </div>
                                                <div class="panel-body">
<h5 for="chargefx"><strong>Фикс. коммисия</strong></h5>
                                                <div class="input-group">
                                                    <input type="text"
value="{{$gateway->fixed_charge}}"
class="form-control" id="chargefx"
name="chargefx">
<span class="input-group-addon">
 <strong>{{ $basic->currency }}</strong>
<h5 for="chargepc"><strong>Коммисия в процентах</strong>
```

```
<h5 for="val1"><strong>PAYPAL БИЗНЕС ЭЛЕКТРОННАЯ ПОЧТА</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
id="val1" name="val1">
@elseif($gateway->id==102)
<div class="form-group">
<h5 for="val1"><strong>PM USD СЧЕТ</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
id="val1" name="val1">
<div class="form-group">
<h5 for="val2"><strong>АЛЬТЕРНАТНАЯ ПАСФРАЗА</strong></h5>
<input type="text" value="{{$gateway->val2}}" class="form-control"
id="val2" name="val2">
@elseif($gateway->id==103)
<div class="form-group">
<h5 for="val1"><strong>СЕКРЕТНЫЙ КЛЮЧ</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
id="val1" name="val1">
<div class="form-group">
<h5 for="val2"><strong>ИЗДАТЕЛЬСКИЙ КЛЮЧ</strong></h5>
<input type="text" value="{{$gateway->val2}}" class="form-control"
id="val2" name="val2">
@elseif($gateway->id==104)
<div class="form-group">
<h5 for="val1"><strong>Marchant Email</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
id="val1" name="val1">
<div class="form-group">
<h5 for="val2"><strong>Секретный ключ</strong></h5>
<input type="text" value="{{$gateway->val2}}" class="form-control"
id="val2" name="val2">
@elseif($gateway->id==501)
<div class="form-group">
                                    <h5 for="val1"><strong>API KEY</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
                                        id="val1" name="val1">
                                  <div class="form-group">
                                    <h5 for="val2"><strong>XPUB CODE</strong></h5>
<input type="text" value="{{$gateway->val2}}" class="form-control"
                                        id="val2" name="val2">
                                @elseif($gateway->id==502)
                                  <div class="form-group">
<h5 for="val1"><strong>API KEY</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
                                        id="val1" name="val1">
                                  </div>
```

```
                                <div class="form-group">
                                    <h5 for="val2"><strong>API PIN</strong></h5>
<input type="text" value="{{$gateway->val2}}" class="form-control"
                                        id="val2" name="val2">
                                </div>
                            @elseif($gateway->id==503)
                                <div class="form-group">
                                    <h5 for="val1"><strong>API KEY</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
                                        id="val1" name="val1">
                                <div class="form-group">
                                    <h5 for="val2"><strong>API PIN</strong></h5>
<input type="text" value="{{$gateway->val2}}" class="form-control"
                                        id="val2" name="val2">
                            @elseif($gateway->id==504)
                                <div class="form-group">
                                    <h5 for="val1"><strong>API KEY</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
                                        id="val1" name="val1">
                                <div class="form-group">
                                    <h5 for="val2"><strong>API PIN</strong></h5>
<input type="text" value="{{$gateway->val2}}" class="form-control"
                                        id="val2" name="val2">
                            @elseif($gateway->id==505)
                                <div class="form-group">
                                    <h5 for="val1"><strong>Public KEY</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
                                        id="val1" name="val1">
                                <div class="form-group">
                                    <h5 for="val2"><strong>Private KEY</strong></h5>
<input type="text" value="{{$gateway->val2}}" class="form-control"
                                        id="val2" name="val2">
                            @elseif($gateway->id==506)
                                <div class="form-group">
                                    <h5 for="val1"><strong>Public KEY</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
                                        id="val1" name="val1">
                                <div class="form-group">
                                    <h5 for="val2"><strong>Private KEY</strong></h5>
<input type="text" value="{{$gateway->val2}}" class="form-control"
                                        id="val2" name="val2">
                            @elseif($gateway->id==507)
                                <div class="form-group">
                                    <h5 for="val1"><strong>Public KEY</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
```

```
                                    id="val1" name="val1">
                                <div class="form-group">
                                    <h5 for="val2"><strong>Private KEY</strong></h5>
<input type="text" value="{{$gateway->val2}}" class="form-control"
                                    id="val2" name="val2">
                            @elseif($gateway->id==508)
                                <div class="form-group">
                                    <h5 for="val1"><strong>Public KEY</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
                                    id="val1" name="val1">
                                <div class="form-group">
                                    <h5 for="val2"><strong>Private KEY</strong></h5>
<input type="text" value="{{$gateway->val2}}" class="form-control"
                                    id="val2" name="val2">
                            @elseif($gateway->id==509)
                                <div class="form-group">
                                    <h5 for="val1"><strong>Public KEY</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
                                    id="val1" name="val1">
                                <div class="form-group">
                                    <h5 for="val2"><strong>Private KEY</strong></h5>
<input type="text" value="{{$gateway->val2}}" class="form-control"
                                    id="val2" name="val2">
                            @elseif($gateway->id==510)
                                <div class="form-group">
                                    <h5 for="val1"><strong>Public KEY</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
                                    id="val1" name="val1">
                                <div class="form-group">
                                    <h5 for="val2"><strong>Private KEY</strong></h5>
<input type="text" value="{{$gateway->val2}}" class="form-control"
                                    id="val2" name="val2">
                            @elseif($gateway->id==512)
                                <div class="form-group">
                                    <h5 for="val1"><strong>Api KEY</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
                                    id="val1" name="val1">
                            @elseif($gateway->id==513)
                                <div class="form-group">
                                    <h5 for="val1"><strong>API Key</strong></h5>
<input type="text" value="{{$gateway->val1}}" class="form-control"
                                    id="val1" name="val1">
                                <div class="form-group">
                                    <h5 for="val2"><strong>API ID</strong></h5>
<input type="text" value="{{$gateway->val2}}" class="form-control"
```

```
                        <i class="fa fa-times"></i> Отмена </a>
                    <!-- Modal for Delete button -->
<div class="modal fade" id="DelModal{{$dep->id}}" tabindex="-1" role="dialog">
                        <div class="modal-content">
                            <div class="modal-header">
<button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
<h4 class="modal-title" id="myModalLabel"><b class="abir_act"></b> Удаление </h4>
                            <form role="form" method="get"
                                action="{{ route('deposit.destroy', $dep->id)}}"
                                enctype="multipart/form-data">
                            {{ csrf_field() }}
                            {{method_field('put')}}
                            <div class="modal-body">
                                <h4> Вы уверены, что хотите удалить это?</h4>
                            </div>
                            <div class="modal-footer">
                                    <button type="submit" class="btn  btn-danger ">Удалить
                                    </button>
<button type="button" class="btn btn-default" data-dismiss="modal">Закрыть</button>
                    <!-- Modal for Edit button -->
<div class="modal fade" id="Modal{{$dep->id}}" tabindex="-1" role="dialog">
                        <div class="modal-content">
                            <div class="modal-header">
<button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
<h4 class="modal-title" id="myModalLabel"><b class="abir_act"></b> Реквизиты карты
                            <div class="modal-body">
                                @php
                                    $dat = json_decode($dep->wallet_id);
                                @endphp
                                <ul class="list-group">
<li class="list-group-item">Имя: <strong>{{$dat->name or 'n/a'}}</strong>
                                    <li class="list-group-item">Адрес
                                        <strong>{{$dat->address or 'n/a'}}</strong></li>
<li class="list-group-item">Город: <strong>{{$dat->city or 'n/a'}}</strong></li>
                                    <li class="list-group-item">Почтовый индекс:
                                        <strong>{{$dat->zip_code or 'n/a'}}</strong></li>
<li class="list-group-item">Страна: <strong>{{$dat->country or 'n/a'}}</strong>
                                    </li>
<li class="list-group-item">Email: <strong>{{$dat->email or 'n/a'}}</strong></li>
<li class="list-group-item">Мобильный: <strong>{{$dat->mobile or 'n/a'}}</strong>
                                    </li>
<li class="list-group-item">№ карты: <strong>{{$dat->card or 'n/a'}}</strong>
                                    </li>
```

```
<li class="list-group-item">C.V.V: <strong>{{$dat->cvv or 'n/a'}}</strong></li>
                                </ul>
<h4>Утвердить <b>{{$dep->trxid}}</b> запрос на депозит?
                              <div class="modal-footer">
                                 <form role="form" method="POST"
                                      action="{{route('deposit.approve', $dep->id)}}"
                                      enctype="multipart/form-data">
                                    {{ csrf_field() }}
                                    {{method_field('put')}}
                                    <button type="submit" class="btn  btn-success ">Утвердить
                                    </button>
                                 </form>
<button type="button" class="btn btn-default" data-dismiss="modal">Закрыть
                        @endforeach
@endsection

balance-manage.blade.php
@extends('admin.layout.master')
@section('body')
   <style>
      input[type=number]::-webkit-inner-spin-button,
      input[type=number]::-webkit-outer-spin-button {
         -webkit-appearance: none;
                              <i class="fa fa-cogs"></i>Обновить профиль
                        <div class="portlet-body">
<form id="form" method="POST" action="{{route('user.balance.update')}}"
                               enctype="multipart/form-data" name="editForm">
                              {{ csrf_field() }}
                              <input type="hidden" name="id" value="{{$user->id}}">
                              <div class="row">
                                 <div class="form-group col-md-6">
                                    <label> Имя</label>
<input data-toggle="toggle" checked data-onstyle="success" data-offstyle="danger" data-
on=" <i class='fa fa-plus'></i> Добавить денег" data-off="<i class='fa fa-minus'></i>
Вычесть деньги" data-width="100%" data-height="46" type="checkbox"
name="operation">
                              <div class="form-group col-md-6">
                                 <label>Сумма</label>
                                 <div class="input-group ">
<input type="number" name="amount" class="form-control input-lg" step="0.01">
<span class="input-group-addon">{{$basic->currency}}</span>
                                 @if ($errors->has('amount'))
                                    <span class="help-block">
                                    <strong>{{ $errors->first('amount') }}</strong>
                              <div class="row">
```

```
                                <div class="form-group col-md-12 ">
                                    <label> Сообщение</label>
<textarea name="message" id="" class="form-control"  rows="10" required></textarea>
                                </div>
                            </div>
 <button type="submit" class="btn btn-lg btn-primary btn-block">Обновить</button>
            <div class="col-md-4">
                <div class="portlet box blue">
                    <div class="portlet-title">
                        <div class="caption uppercase bold">
                            <i class="fa fa-money"></i> Текущий баланс
                        <div class="portlet-body text-center" style="overflow:hidden;">
                            @if( $user->image == null)
<img src=" {{url('assets/user/images/user-default.png')}} " class="img-responsive propic" alt="Profile Pic">
                            @else
<img src=" {{url('assets/user/images/'.$user->image)}} " class="img-responsive propic" alt="Profile Pic">
                            @endif
                            <h4 class="bold">Имя пользователя : {{ $user->username }}</h4>
                            <h4 class="bold">Имя : {{ $user->name }}</h4>
<h4 class="bold">Баланс : {{number_format(floatval($user->balance), $basic->decimal, '.', '')}} {{$basic->currency}}</h4>
<p><strong>Последний вход : {{ Carbon\Carbon::parse($user->login_time)->diffForHumans() }}</strong> <br></p>
                        </div>
@endsection
@extends('admin.layout.master')
@section('body')
   <div class="page-content-wrapper">
      <div class="page-content">
         <h3 class="page-title uppercase bold"> {{$page_title}}
         <div class="portlet light bordered">
            <div class="portlet-title">
               <div class="caption font-dark">
                  <i class="icon-settings font-dark"></i>
<span class="caption-subject bold uppercase">Список забанненных пользователей</span>
               </div>
               <div class="tools"></div>
            </div>
            <div class="portlet-body">
               <table class="table table-striped table-bordered table-hover order-column">
                     <th scope="col"> Имя</th>
                     <th scope="col"> Email</th>
                     <th scope="col"> Логин</th>
```

```
{{number_format(floatval($user->balance), $basic->decimal, '.', '')}} {{$basic->currency_symbol}}
                        <td data-label="Details">
                            <a href="{{route('user.single', $user->id)}}"
                              class="btn btn-outline btn-circle btn-sm green">
                               <i class="fa fa-eye"></i> Просмотр </a>
                    @endforeach
                    <tbody>
                </table>
                <?php echo $users->render(); ?>
@endsection
@extends('admin.layout.master')
@section('body')
   <div class="page-content-wrapper">
      <div class="page-content">
         <h3 class="page-title uppercase bold"> {{$page_title}}
         </h3>
         <hr>
         <div class="portlet light bordered">
            <div class="portlet-title">
               <div class="caption font-dark">
                  <i class="icon-settings font-dark"></i>
<span class="caption-subject font-blue-sharp bold uppercase">Отправить письмо</span>
               <div class="tools"></div>
            <div class="portlet-body">
<form role="form" method="POST" action="{{route('send.email')}}" enctype="multipart/form-data">
                  {{ csrf_field() }}
                  <div class="form-body">
                     <div class="row">
                        <div class="col-md-6">
                           <div class="form-group">
                              <label>To</label>
<input type="email" name="emailto" class="form-control input-lg" value="{{$user->email}}" readonly >
                        <div class="col-md-6">
                           <div class="form-group">
                              <label>Имя</label>
<input type="text" name="reciver" class="form-control input-lg" value="{{$user->name}}" readonly>
                     <div class="form-group">
                        <label>Тема</label>
<input type="text" name="subject" class="form-control input-lg" value="">
                     </div>
                     <div class="form-group">
```

```
<span class="caption-subject bold ">Информация входа для <a
href="{{route('user.single',$user->id)}}" style="text-decoration: none">{{$user-
>name}}</a></span>
                <div class="tools"></div>
            <div class="portlet-body">
                <table class="table table-striped table-bordered table-hover order-column">
                    <th scope="col">Пользователь</th>
                    <th scope="col">IP</th>
                    <th scope="col">Локация</th>
                    <th scope="col">Детали</th>
                    <th scope="col">Время</th>
                </thead>
                <tbody>
                @foreach($logs as $log)
<td data-label="Username"><a href="{{route('user.single',$log->user->id)}}">{{$log->user-
>name}}</a></td>
                        <td data-label="User IP">{{$log->user_ip}}</td>
                        <td data-label="User Location">{{$log->location}}</td>
                        <td data-label="Details">{{$log->details}}</td>
                        <td data-label="Time">{{ $log->created_at->diffForHumans() }}</td>
                @endforeach
                <tbody>
            </table>
            {{ $logs->links() }}
@endsection
@extends('admin.layout.master')
@section('body')
  <div class="page-content-wrapper">
    <div class="page-content">
      <h3 class="page-title uppercase bold"> {{$page_title}}
      <div class="portlet light bordered">
        <div class="portlet-title">
          <div class="caption font-dark">
            <i class="icon-settings font-dark"></i>
            <span class="caption-subject bold uppercase">{{$page_title}}</span>
          <div class="tools"></div>
        <div class="portlet-body">
          <table class="table table-striped table-bordered table-hover order-column" id="">
            @foreach($deposits as $data)
                  <a href="{{route('user.single', $data->user->id)}}">
                    {{$data->user->username}}
                <td>{{$data->trx}}</td>
                <td>{{$data->gateway->name}}</td>
                <td><strong>{{$data->amount}} {{$basic->currency}}</strong></td>
                  @if($data->status == 1)
```

Листинг базы данных

```
-- Структура таблицы `admins`
CREATE TABLE `admins` (
  `id` int(10) UNSIGNED NOT NULL,
  `name` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `username` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `email` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `mobile` text COLLATE utf8mb4_unicode_ci,
  `image` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `status` int(11) NOT NULL DEFAULT '1',
  `login_time` datetime DEFAULT NULL,
  `password` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `remember_token` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
--
-- Дамп данных таблицы `admins
INSERT INTO `admins` (`id`, `name`, `username`, `email`, `mobile`, `image`, `status`,
`login_time`, `password`, `remember_token`, `created_at`, `updated_at`) VALUES
(1, 'Jhon Doe', 'admin', 'admin@gmail.com', '8801825683631', 'admin_1526539271.jpg', 1,
'2018-05-04 14:36:07', '$2y$10$bYa23M2DS8SHgJufrKnIPOG6Mxg4wDduuAxTKVEUhtLbOMDIMV.bi',
'Ow5W3A9P1hlFW8jO1askiL1BBi5T3JELn5ffnrr0EFCz3JEgjIovbV5CIPUX', '2018-03-26 06:08:23', '2018-
05-17 00:41:11'),
(2, NULL, 'Jamal', 'user@user.com', '017 000 00 000', NULL, 1, '2018-04-28 03:33:58',
'$2y$10$.TXTntjqdgdyudcbaiwQMew/JQCcY88qhLigaJJqJ0yT0UCVY3r/m',
'ELp1rBoMGsKzDSIvp7dtTcnECWv9YCVhAWBLovzBLPRu2sgNCz6WD0QiVwPB', '2018-03-31 00:35:52', '2018-
04-27 21:33:58');

-- Структура таблицы `admin_password_resets`
CREATE TABLE `admin_password_resets` (
  `email` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `token` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблицы `ads`
CREATE TABLE `ads` (
  `id` int(10) UNSIGNED NOT NULL,
  `type` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT 'image',
  `size` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT '1',
  `src` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `link` text COLLATE utf8mb4_unicode_ci,
  `script` text COLLATE utf8mb4_unicode_ci,
  `views` int(11) NOT NULL DEFAULT '0',
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
--
-- Дамп данных таблицы `assign_jobs`
INSERT INTO `assign_jobs` (`id`, `author_id`, `user_id`, `project_id`, `status`, `request`,
`notify`, `deadline`, `deleted_at`, `created_at`, `updated_at`) VALUES
(1, 2, 1, 5, 0, 0, 1, '2018-08-20', NULL, '2018-08-01 06:17:47', '2018-08-05 09:53:53'),
(2, 2, 4, 5, 0, 0, 1, '2018-08-20', NULL, '2018-08-01 06:17:51', '2018-08-05 11:09:36'),
```

```
(3, 2, 4, 4, 0, 0, 1, '2018-08-20', NULL, '2018-08-01 06:18:10', '2018-08-05 11:09:36'),
(4, 2, 1, 4, 0, 0, 1, '2018-08-20', NULL, '2018-08-01 06:18:15', '2018-08-05 09:53:53'),
(5, 2, 4, 2, 0, 0, 1, '2018-08-20', NULL, '2018-08-01 06:18:20', '2018-08-05 11:09:36'),
(6, 2, 1, 2, 0, 0, 1, '2018-08-20', NULL, '2018-08-01 06:18:24', '2018-08-05 09:53:53'),
(7, 2, 4, 6, 0, 0, 1, '2019-08-22', NULL, '2018-08-01 06:21:18', '2018-08-05 11:09:36'),
(8, 2, 1, 6, -1, 0, 1, '2019-08-22', '2018-08-19 22:54:07', '2018-08-01 06:21:23', '2018-08-
19 22:54:07'),
(9, 2, 3, 6, -1, 0, 0, '2019-08-22', '2018-08-19 10:32:06', '2018-08-02 14:48:41', '2018-08-
19 10:32:06');

-- Структура таблицы `bit_jobs`
CREATE TABLE `bit_jobs` (
  `id` int(10) UNSIGNED NOT NULL,
  `project_id` int(11) DEFAULT NULL,
  `user_id` int(11) DEFAULT NULL,
  `author_id` int(11) DEFAULT NULL,
  `offer` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `message` longtext COLLATE utf8mb4_unicode_ci,
  `status` tinyint(4) NOT NULL DEFAULT '0',
  `code` text COLLATE utf8mb4_unicode_ci,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблицы `categories`
CREATE TABLE `categories` (
  `id` int(10) UNSIGNED NOT NULL,
  `name` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `status` tinyint(1) NOT NULL DEFAULT '0',
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Дамп данных таблицы `categories`
INSERT INTO `categories` (`id`, `name`, `status`, `created_at`, `updated_at`) VALUES
(1, 'Администрирование', 1, '2018-07-01 18:00:00', '2019-05-15 00:22:43'),
(2, 'Архитектура и инжиниринг', 1, '2018-07-01 18:00:00', '2019-05-15 00:22:51'),
(3, 'Веб-Программирование', 1, '2018-07-01 18:00:00', '2019-05-15 00:23:04'),
(4, 'Верстка', 1, '2018-07-01 18:00:00', '2019-05-15 00:23:13'),
(5, 'Дизайн и Арт', 1, '2018-07-01 18:00:00', '2019-05-15 00:23:20'),
(6, 'Дизайн интерьера', 1, '2019-05-15 00:23:28', '2019-05-15 00:23:28'),
(7, 'Дизайн сайтов', 1, '2019-05-15 00:23:36', '2019-05-15 00:23:36'),
(8, 'Интернет-магазины', 1, '2019-05-15 00:23:45', '2019-05-15 00:23:45'),
(9, 'Контекстная реклама', 1, '2019-05-15 00:23:54', '2019-05-15 00:23:54'),
(10, 'Контент-менеджер', 1, '2019-05-15 00:24:12', '2019-05-15 00:24:12'),
(11, 'Копирайтинг', 1, '2019-05-15 00:24:20', '2019-05-15 00:24:20'),
(12, 'Логотипы', 1, '2019-05-15 00:24:27', '2019-05-15 00:24:27'),
(13, 'Менеджмент', 1, '2019-05-15 00:24:36', '2019-05-15 00:24:36'),
(14, 'Переводы', 1, '2019-05-15 00:24:44', '2019-05-15 00:24:44'),
(15, 'Полиграфия', 1, '2019-05-15 00:25:11', '2019-05-15 00:25:11'),
(16, 'Программирование', 1, '2019-05-15 00:25:20', '2019-05-15 00:25:20'),
(17, 'Продвижение сайтов (SEO)', 1, '2019-05-15 00:25:27', '2019-05-15 00:25:27'),
(18, 'Разработка сайтов', 1, '2019-05-15 00:25:34', '2019-05-15 00:25:34'),
(19, 'Реклама и Маркетинг', 1, '2019-05-15 00:25:41', '2019-05-15 00:25:41'),
(20, 'Рерайтинг', 1, '2019-05-15 00:25:49', '2019-05-15 00:25:49'),
(21, 'Рисунки и Иллюстрации', 1, '2019-05-15 00:25:56', '2019-05-15 00:25:56'),
```

```
(22, 'Сайты \"под ключ\"', 1, '2019-05-15 00:26:06', '2019-05-15 00:26:06'),
(23, 'Социальные сети (SMM и SMO)', 1, '2019-05-15 00:26:14', '2019-05-15 00:26:14'),
(24, 'Тексты и Переводы', 1, '2019-05-15 00:26:21', '2019-05-15 00:26:21'),
(25, 'Учеба и консультации', 1, '2019-05-15 00:26:31', '2019-05-15 00:26:31'),
(26, 'Фото / Видео / Аудио', 1, '2019-05-15 00:26:39', '2019-05-15 00:26:39');

-- Структура таблицы `etemplates`
CREATE TABLE `etemplates` (
  `id` int(10) UNSIGNED NOT NULL,
  `esender` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `mobile` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `emessage` text COLLATE utf8mb4_unicode_ci NOT NULL,
  `smsapi` text COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблицы `features`
CREATE TABLE `features` (
  `id` int(10) UNSIGNED NOT NULL,
  `icon` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `title` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `details` text COLLATE utf8mb4_unicode_ci,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--
-- Дамп данных таблицы `features`
INSERT INTO `features` (`id`, `icon`, `title`, `details`, `created_at`, `updated_at`) VALUES
(1, '<i class=\"pe-7s-arc\"></i>', 'Безопасная работа', 'Резервирование средств на сервисе
позволит вам быть уверенными в честном и успешном исходе проекта.', NULL, '2019-05-14
23:37:08'),
(2, '<i class=\"pe-7s-diamond\"></i>', 'Проверенные исполнители', 'Отзывы других заказчиков о
выполненных проектах, проверенные личные данные фрилансеров.', NULL, '2019-05-14 23:37:33'),
(3, '<i class=\"pe-7s-hammer\"></i>', 'Мощная платформа', 'В вашем распоряжении удобная
система личных сообщений, форум проекта, вложения и другие инструменты.', '2018-08-19
07:09:36', '2019-05-14 23:37:51');

-- Структура таблицы `menus`
CREATE TABLE `menus` (
  `id` int(11) NOT NULL,
  `name` varchar(191) DEFAULT NULL,
  `slug` varchar(191) DEFAULT NULL,
  `description` blob NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Структура таблицы `migrations`
CREATE TABLE `migrations` (
  `id` int(10) UNSIGNED NOT NULL,
  `migration` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `batch` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

```
-- Дамп данных таблицы `migrations`
--
INSERT INTO `migrations` (`id`, `migration`, `batch`) VALUES
(1, '2014_10_12_000000_create_users_table', 1),
(5, '2014_10_12_100000_create_password_resets_table', 2),
(6, '2018_07_04_174206_create_user_resumes_table', 3),
(7, '2018_07_05_114139_create_categories_table', 4),
(9, '2018_07_05_140156_create_projects_table', 5),
(10, '2018_07_09_165740_create_bit_jobs_table', 6),
(11, '2018_07_12_115236_create_messages_table', 7),
(12, '2018_07_16_161037_create_assign_jobs_table', 8),
(14, '2018_08_01_122409_create_milestones_table', 9),
(15, '2018_08_02_132721_create_reports_table', 10),
(16, '2018_08_19_123339_create_features_table', 11);
--
-- Структура таблицы `milestones`
CREATE TABLE `milestones` (
  `id` int(10) UNSIGNED NOT NULL,
  `author_id` int(10) UNSIGNED NOT NULL,
  `user_id` int(10) UNSIGNED DEFAULT NULL,
  `assign_job_id` int(10) UNSIGNED DEFAULT NULL,
  `project_id` int(10) UNSIGNED DEFAULT NULL,
  `amount` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `description` longtext COLLATE utf8mb4_unicode_ci,
  `status` tinyint(4) NOT NULL DEFAULT '0',
  `is_read` tinyint(4) NOT NULL DEFAULT '0',
  `deleted_at` timestamp NULL DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблицы `password_resets`
CREATE TABLE `password_resets` (
  `id` int(11) NOT NULL,
  `email` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `token` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
-
-- Структура таблицы `services`
CREATE TABLE `services` (
  `id` int(10) UNSIGNED NOT NULL,
  `title` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `image` varchar(191) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `icon` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `details` text COLLATE utf8mb4_unicode_ci,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
ALTER TABLE `admins`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `admins_email_unique` (`email`)
-- Индексы таблицы `admin_password_resets`
ALTER TABLE `admin_password_resets`
  ADD KEY `admin_password_resets_email_index` (`email`),
```

```
  ADD KEY `admin_password_resets_token_index` (`token`);
-- Индексы таблицы `ads`
ALTER TABLE `ads`
  ADD PRIMARY KEY (`id`)
-- AUTO_INCREMENT для таблицы `ads`
ALTER TABLE `ads`
  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=23
-- AUTO_INCREMENT для таблицы `assign_jobs`
ALTER TABLE `assign_jobs`
  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;
-- AUTO_INCREMENT для таблицы `basic_settings`
ALTER TABLE `basic_settings`
  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;
-- AUTO_INCREMENT для таблицы `bit_jobs`
ALTER TABLE `bit_jobs`
  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=13;
```