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: Разработка Website на основе клиент-серверного приложения

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

Выполнил: Хиониди Г.Г. Группа:ВТ-15-2
Научный руководитель:д.т.н., профессор Казиев Г.З.

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

Рецензент: д.т.н., профессор_____У.А. Тукеев
«____» _____ 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 дипломногопроекта

Студенту Хиониди Георгию Георгиевичу

Тема проекта: Разработка Website на основе клиент-серверного приложения

Утверждена приказом по университету №124 от «26» октября2018 г.

Срок сдачи законченного проекта «21» 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 дипломногопроекта:

- 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 9 таблиц, 69 иллюстрации.

Основная рекомендуемая литература:

1 Дронов, Владимир JavaScript и AJAX в Web-дизайне / Владимир Дронов. - М.: "БХВ-Петербург", 2012. - 736 с.

2 Хэррон, Дэвид Node.js Разработка серверных веб-приложений на JavaScript / Дэвид Хэррон. - М.: ДМК Пресс, 2014. - 144 с.

3 Сухов, Кирилл Node.js. Путеводитель по технологии / Кирилл Сухов. - М.: ДМК Пресс, 2015. - 416 с.

4 Крокфорд Д.А. JavaScript. Сильные стороны. – СПб.: БХВ – Петербург, 2019. – 656 с.

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.	12.03.2019 – 14.05.2019	
Программное обеспечение	Майкотов М.Н.	08.04 – 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	05.11.2018 – 21.12.2018	
Проектная часть	07.01.2019 – 31.01.2019	
Экспериментальная часть	04.02.2019 – 12.04.2019	

Дата выдачи задания «25» 10 2018г.

Заведующий кафедрой_____Т.С. Картбаев

Научный руководительпроекта_____Г.З. Казиев

Задание принял к
исполнению студент_____Г.Г. Хиониди

Аңдатпа

Дипломдық жобаның тақырыбы: «Клиент-серверлік қосымша негізінде Website әзірлеу ».

Клиентті іске асыру үшін келесі технологиялар: HTML, CSS, SCSS, JavaScript, jQuery оқу және пайдалану қажет болды.

Сервер тарапынан: Node.js, Express, ejs, SQL, JavaScript, XAMMP, phpMyAdmin.

Жоба 5 негізгі тарауларды, қорытындыларды, әдебиеттер тізімі мен міндетті қосымшаларды қамтиды.

Кіріспе осы веб-қосымшаның мәнін, оның дамуының өзектілігін сипаттады. Осы қолданбаның ауқымын кеңейтеді. Негізгі мақсаттар мен міндеттер анықталды.

Бірінші тарау аналитикалық болып табылады және бағдарламалық өнімнің сипаттамасын және оны іске асыру үшін технологияларды таңдауды қамтиды.

Екінші тарау модельдер мен құрылымдық кестелер арқылы клиент бөлігі мен дерекқор құрылымының дизайнын көрсетеді.

Үшінші бөлім практикалық және бағдарламалық өнімнің іске асырылу сипаттамасын, әрбір веб-беттің жарылған талдауын қамтиды.

Төртінші бөлік - бұл жобаның экономикалық тұрғыдан талдаудан өткен техникалық-экономикалық негіздемесі.

Бесінші тарауда негізгі кемшіліктер анықталатын еңбек жағдайларын қарастыру кіреді, ол кейінгі әрекеттерде жойылады.

Қорытындылай келе, жасалған жұмыстың нәтижелері, дайын бағдарламалық өнімге әкелген мақсаттар мен міндеттерге қол жеткізу сипатталған.

Аннотация

Тема дипломного проекта: «Разработка Website на основе клиент-серверного приложения».

Для реализации клиентской части потребовалось изучить и использовать следующие технологии: HTML, CSS, SCSS, JavaScript, jQuery.

Для серверной части: Node.js, Express, ejs, SQL, JavaScript, XAMMP, СУБД phpMyAdmin.

В дипломный проект входит 5 основных глав, заключение, список литературы и обязательные приложения.

Во введение была описана суть данного веб-приложения, актуальность его разработки. Раскрывается область применения данного приложения. Определены основные цели и задачи.

Первая глава является аналитической и в нее входит описание программного продукта и выбор технологий для его реализации.

Вторая глава посредством моделей и структурных таблиц, показывает проектирование структуры клиентской части и базы данных.

Третья часть является практической и включает в себя описание реализации программного продукта, подетальный разбор каждой составляемой веб-страницы.

Четвертая часть является технико-экономическим обоснованием, где пошагово была разобрана реализация данного проекта с экономической точки зрения.

Пятая глава включает в себя рассмотрения условий труда, где определяется главный недостаток, который устраняется в последующих действиях.

В заключении описаны результаты проделанной работы, достижения целей и выполнения задач, которые привели к получению готового программного продукта.

Annotation

Theme of the graduation project: «Website development based on client-server application».

To implement the client side, it was necessary to study and use the following technologies: HTML, CSS, SCSS, JavaScript, jQuery.

For the server side: Node.js, Express, ejs, SQL, JavaScript, XAMMP, phpMyAdmin DBMS.

The thesis project includes 5 main chapters, conclusion, list of references and mandatory applications.

The introduction described the essence of this web application, the relevance of its development. Expands the scope of this application. The main goals and objectives are defined.

The first chapter is analytical and includes a description of the software product and the choice of technologies for its implementation.

The second chapter by means of models and structural tables shows the design of the structure of the client part and the database.

The third part is practical and includes a description of the implementation of the software product, an exploded analysis of each web page.

The fourth part is a feasibility study, where the implementation of this project was analyzed from an economic point of view.

The fifth chapter includes consideration of working conditions, where the main drawback is determined, which is eliminated in subsequent actions.

In conclusion, the results of the work done, the achievement of goals and tasks that led to the finished software product is described.

Содержание

Введение

Наше время — это время компьютерных технологий, компьютерных сетей и информационных систем в целом. Каждый человек в том или ином смысле задействован в компьютерных технологиях и зависим от них, так как в современном мире почти каждый аспект человеческой жизни автоматизирован и упрощен для пользования. Как показывает статистика использования WEB-приложений в интернете стоит на первом месте, WEB-приложения это всем известные сайты, которые в современных кондициях. Они представляют собой огромный спектр функциональных особенностей, а также большое количество хранимой информации, технологии, которые используются для создания WEB-приложений, постоянно обновляются и становятся удобнее для разработчиков, что в конечном итоге дает идеальный продукт для клиента данного сайта. Исходя из изложенной информации, можно сделать следующий вывод, что разработка WEB-приложений является очень актуальной профессией и очень интересной.

Около 95% пользователей компьютерных сетей используют WEB-приложение для достижения своей цели, это могут быть покупки в интернет-магазинах, бронирование авиабилетов, получение нужных документов, т. е почти все аспекты жизнедеятельности человека можно найти и решить в той или иной мере с помощью интернета, а интернет представляет собой огромное количество WEB - страниц. Так же в современное время все сайты являются кроссбраузерными и адаптивными, т. е возможность использование сайтов на мобильных устройствах делает их уникальными и тем самым уходит потребность для создания программного приложений, которые пользователь скачивает непосредственно на свое мобильное устройство.

В данном диплом проекте было разработано WEB-приложение (сайт) на тему “Конфигуратор технологии”, т.е это проект, который содержит в себе конфигуратор той или иной технологии и в качестве среды изучения была выбрана технология информационных систем - Персональный компьютер. Сайт представляет собой интернет-магазин со встроенным конфигуратором. Так же данный конфигуратор является универсальным, и его предназначение зависит от того какая база данных используется для выполнения его целей, поэтому его можно смело обобщать и называть - Конфигуратор технологий. Если перечислить все возможности данного WEB-приложения, не углубляясь, то можно сказать следующее: удобный интерфейс (меню) для навигации по сайту, регистрация и вход в учетную запись клиента, возможность собрать компьютер с нуля использую все необходимые детали, предложенные конфигуратором. Возможность изменить нужные детали на необходимые конкретно вам у уже готовых компьютеров, приобрести все необходимые товары, приобрести уже готовый персональный компьютер, представлен огромный спектр аксессуаров, которые можно докупить к персональному компьютеру, очень удобный интерфейс оформления заказа и покупки и многое другое.

Актуальность данного WEB-приложения заключается в том, что оно универсальное, и сочетает в себе как интернет-магазин, так и функциональный конфигуратор, который позволяет в кротчайшие сроки собрать персональный компьютер, используя только совместимые элементы компьютера, которые готовы в базе данных. Большое количество людей ошибается при покупке готово персонально компьютера с огромной наценкой, данный сайт ликвидирует наценку тем, что предоставляет каждый элемент персонального компьютера по отдельной цене, тем самым в результате получится рабочий компьютер с полностью работающими между собой элементами за общую цену составляющих этого компьютера.

В качестве целей я обозначил следующие пункты:

- создание удобного и понятного интерфейса использования;
- простота использования конфигуратора при сборке компьютера;
- изменения конфигурации уже готового продукта;
- интернет-магазин с большим количеством продукции;
- большое количество дополнительных аксессуаров;
- удобное оформление заказа и запросов на помощь на сайте;
- быстродействие WEB-страницы по все современным нормам;
- использование передовых технологий при создании сайта.

1 Описание программного продукта

Конфигуратор – это механизированное программное средство, задачами которого является выбор того или иного объекта в списке объектов по определенному принципу или правилу, которое прописано в программном коде. В случае конфигуратора, который был создан для сборки компьютеров, он имеет умную функцию подбора нужных деталей, которые будут взаимодействовать без ошибок, так как они заранее подготовлены в базе данных. Конфигуратор работает независимо от товара, выбранного перед конфигурированием, т. е клиент может выбрать дополнительную периферию и после этого собрать свой собственный компьютер с нужными параметрами для той или иной цели либо же может изменить детали уже собранных персональных компьютеров [3].

Удобство данного конфигуратора заключается в том, что он удобно выдает всю информацию о той или иной детали компьютера посредством запросов в базу данных относительно того, какая деталь, выбрана и рассматривается клиентом в качестве выбранной. Данной WEB-приложения является интернет-магазином со встроенным конфигуратором. Все это разрабатывалось в качестве одного проекта. С помощью данного сервиса люди могут собрать себе компьютер, из современных деталей узнав о них самую свежую и правдивую информацию. Особенностями данного конфигуратора является то, что он выдает только совместимые детали компьютеров и избегает погрешности в работоспособности. Данный сервис не имеет себе аналогов по информативности и функциональности [4]. Функциональные особенности данного сайта имеют высокую актуальность и внедряемость во все аспекты с возможностью выбора деталей до самых мелочей.

Перед созданием данного WEB-приложения были поставлены определенные задачи:

- изучение различного рода литературы по программированию;
- разделить разработку на 3 части;
- для каждой части разработки найти оптимальный метод выполнения;
- выбор среды разработки;
- создать WEB-приложение.

Функциональные возможности приложения:

- возможность регистрации и авторизации клиентов;
- покупка ПК или дополнительных аксессуаров;
- изменение конфигураций готовых ПК;
- создание новой конфигурации использую конфигуратор;
- оформление заказа с уведомлением на электронную почту;
- заказ звонка;
- возможность навигации по листингу;
- информативность данных о конфигураторе;
- гибкая база данных с множеством таблиц;

– безопасность хранения личной информации пользователя.

В данном случае WEB-приложение представляет собой не только страницу с конфигуратором, а также весь спектр услуг интернет-магазина и предоставляет всю необходимую информацию, и его смело можно называть сайтом, необходимым для сборки компьютеров и дополнительной периферии.

1.1 Назначение программного продукта

Разработанный программный продукт предназначен для автоматизации продажи компьютеров через интернет, т. е компания, которая занимается продажей и сборкой компьютеров может быть крупной или средней в зависимости от потребностей. Данный сервер должен облегчить процесс выбора компьютера из-за своей информативности, а также дает возможность клиенту лично собрать компьютер с совместимыми деталями с подробным описанием каждой из них.

WEB-приложение состоит из 7 частей:

– регистрация и вход;

Данный процесс предназначен для создания клиентской базы компьютерной компании, т. е данный процесс регистрации является важным в данном сервисе, потому что только после регистрации и дальнейшей авторизации клиент получает возможность оформления покупок.

Но следует сделать вывод, что для ликвидации информационных нужд клиента, а именно просмотр цены и пробный вариант сборки компьютера или, же изменения уже готовой продукции на данном сервисе возможны и без авторизации. При регистрации того или иного пользователя, он заносится в базу данных, а точнее в клиентскую таблицу. Маркетинговый момент в данной системе заключается в том, что клиент при регистрации вносит свою электронную почту, что после регистрации дает сайту возможность рассылки новостей о компании на личную почту зарегистрированного пользователя. После оформления покупки клиент заносится в клиентскую базу и не является клиентом, который впервые произвел заявку на покупку компьютера.

Регистрация позволит каждому клиенту индивидуально пользоваться услугами конфигуратора и его информационной части, каждая авторизация создает отдельно принятую сессию, и она является безопасной до тех пор, пока клиент сам ее не покинет или по времени. Сессия каждого пользователя является анонимной информацией, как и данные пользователя кроме электронной почты. Данные отправляются менеджеру, подтверждающему заказ на сборку или на покупку, на личную почту тем самым уменьшает возможность ошибок или изменение данных пользователя. Пользователь, не покинувший предыдущую сессию, не сможет войти от другого имени, чтобы покинуть текущую сессию недостаточно перезагрузить страницу, нужно нажать на кнопку выйти либо же выйти со страницы дожидаясь самостоятельной перезагрузки сервера. Как правило на авто выход

пользователя с текущей сессии, если он покинул сайт достаточно 10 минут, после чего клиент может зайти на сайт и пройти регистрацию повторно.

– модельный ряд;

Модельный ряд представляет собой большое количество компьютеров двух типов, первый это небольшое количество базовых компьютеров с возможностью изменить их конфигурацию. Можем приобрести компьютеры данного типа, либо можем перейти к конфигурации данного компьютера, просмотрев все его детали и прочитать о них всю необходимую информация, а также поменять на нужные нам детали [20].

– корзина и оформление заказа;

После нажатия на кнопку купить готовый товар, либо же при нажатии кнопки купить у конфигуратора, на сайте автоматически открывается корзина. Корзина дает клиенту возможность увидеть, какой товар вы собираетесь оформить на покупку или же на сборку. Корзина является адаптивной единицей сайта и так же имеет логику, имеет параметры удаления элементов. Так же в конце всех товаров, записанных в корзину, существует функция оформления заказа, которая представляет собой форму, которую необходимо заполнить всем пользователям веб-приложения и в дальнейшем отправить на сервер. После отправки сообщения проходит обработки на сервере и присылает заявку на личную почту консультанта данного сервиса, после чего происходит перезвон клиенту для подтверждения и уточнения деталей [17].

– конфигуратор;

Является основным элементов WEB-приложения, включает в себя порядка 15 баз данных с полностью защищенным соединением. Имеет функциональную базу на основе выборе элементов той или иной группы. Имеет встроенные информационные панели, показывающие увидеть все характеристики, выбранной клиентом детали.

Конфигуратор предоставляет изначально только совместимые и современные элементы конфигурации. Имеет порядка пятнадцати элементов на выбор той или иной категории имеет функцию переноса всех выбранных деталей в корзину, так же существует возможность приобрести необходимые детали по отдельности или в единичном количестве. Имеет встроенные категории покупки дополнительной периферии, которые передаются с каталога аксессуаров.

Конфигуратор оснащен панелью навигации по конфигурируемым категориям тем самым создает удобный интерфейс при переходе или при выборе того или иного товара. При выборе функции конфигуратор у собранного компьютера, имеющего такую функцию в модельном ряду, будет произведен переход в конфигуратор. Где программа автоматически определит все детали данного компьютера и выберет их в конфигураторе, после чего клиент получает полную информацию о каждой детали данного компьютера с возможностью изменить любую совместимую деталь на деталь похожего типа. Клиент имеет возможность для любого компьютера докупить дополнительные аксессуары, такие как мышка, клавиатура, коврик для

мышки, монитор, наушники. Можно с уверенностью сказать, что конфигуратор дает большие возможности, пользователям данного сервиса предоставляя самим людям читать информацию и собирать свои компьютеры. Одной из основных целей создания данного сервиса было создать удобный интерфейс с огромной информативностью и функциональностью что и получилось в конечном результате.

– каталог аксессуаров;

Все аксессуары, предоставленные в конфигураторе, имеют отдельный каталог. Каталог аксессуаров работает независимо от конфигуратора и является продолжением каталога модельного ряда. Данный каталог предоставляет товары дополнительной периферии и предоставляет изображения всех товарных единиц, а также модель и цену для наглядности. Так же возможность покупки и переход непосредственно в корзину, а дальше в оформления заказа. Каталог предоставляет удобный и с красивым дизайном интерфейс выбора категории аксессуаров.

– разработчикам;

Разработанный сайт предоставляет уникальную возможность и для программистов, которые желают внедрить разработанный конфигуратор на свой сайт. Созданный конфигуратор является универсальным и его возможности зависят от того какая база данных подключена и по какому принципу выбираются конфигурируемые элементы настройка конфигуратора простая и универсальная. Исходи из этого, было принято решение поместить данный конфигуратор в сеть, перейти на который может только пользователь данного проекта. В современном мире все больше и больше интернет сервисов предоставляет свои программы с открытым исходным кодом и данный проект не исключение, потому как вторая по важности цель дипломного проекта, это создать самое современное WEB-приложение.

– обратная связь;

Последней, но не маловажной частью WEB-приложения является то, что существует функция обратной связи, с помощью которой существует возможность ввести ваш номер телефона и получить обратный звонок от менеджера или консультанта работающее в данном сервисе. После отправки вашего номера консультант опять же получает уведомление в личной почте и сразу же организовывает обратную связь для консультации.

В итоге можно сделать следующий вывод, что данное WEB-приложение является очень актуальным и нужным в процессе сбора вычислительной техники, а использование всех современных норм и языков программирования позволило создать удобный интерфейс совместно с сильнодействующими функциональными особенностями. Данный сервис ликвидировал недостатки всех похожих сайтов и предлагает исходный код, что в современном мире очень помогает разработчикам.

Назначение данного сайта является как информационным, так и интернет-магазином, что делает его оптимизированным на просторах интернета. Внедрить конфигуратор, который используется на данном веб-

приложении не составит труда в любую отрасль, где он необходим посредством подключения необходимой базы данных.

1.2 Планировка создания WEB-приложения

Перед выбором сред и средств разработки, был выбран метод разработки сайта, был выбран метод разделения разработки на 4 части: первая часть – разработка макета сайта, т.е. непосредственный дизайн; вторая часть – разработка клиентской части WEB - приложения; третья часть – разработка сервера; четвертая часть – объединение и в результате приложение с готовым клиент-серверным взаимодействием [1].

1.2.1 Обоснование необходимости дизайна

Дизайн сайта является неотъемлемой частью всей разработки, так как от того, как выглядит сайт, зависит количество клиентов, которые на нем работают, в современном мире, в интернете огромное количество сайтов (WEB-приложений), поэтому хорошо продуманный дизайн с элементами цветовых контрастов, будет выделять именно ваш сайт среди других. Как известно на данный момент инструменты для создания сайта очень сильно прогрессируют и дают возможность создать интерфейс любого вида, поэтому сложность шаблона не является проблемой для верстки сайта.

Разработка шаблона в общих понятиях делится на 2 этапа. Первый этап представляет собой выбор цветовой гаммы, а также создание слоев заднего фона, на что в последующем будет находиться интерфейс и блочные элементы. Второй этап — это создание визуального функционала WEB-приложение тем самым показать интерфейс в макете для будущей верстки и для создания непосредственно функционала. Для создания дизайна требуется техническое задание, которое разрабатывается тоже на стадии планировки действий, после чего готовое техническое задание получает внешние стили и преобразуется в макет (шаблон). Полученный шаблон имеет большое количество слоев, которые взаимодействую между собой, и в целом преобразуют наше техническое задание в нефункциональную красивую картинку, которая в дальнейшем преобразуется в сайт. Так же при создании шаблона очень важным моментом является выбор подходящих шрифтов их размерность, цвет, межстрочное расстояние, вертикальное расстояние и наложение на него стилей, например, тени для шрифта.

Данное WEB-приложение представляет собой одностраничный сайт, т. е листинг, такое решение было принято, так как на данный момент самым популярным вариантом сайтов является именно листинг, потому что он нейтрализует переход на другие страницы, тем самым загружаясь и тратя время на загрузку компонентов другой страницы. Листинг представляет собой длинную страницу, на котором находится вся необходимая информация. Навигация обычно спускает сайт в нужное место при нажатии на нужный пункт меню и соответственно поднимает. Соответственно если мы выбрал

формат сайта в виде листинга, то и дизайн будет соответствующего типа. Такого рода дизайн не должен быть 1 файловым, так как создание дизайна сайта подразумевает, и создание дизайна для мобильных устройств, планшетов телевизоров, которые имеют возможность использовать WEB-браузер и непосредственно сам ПК. После этого все три файла (шаблона) для трех устройств подвергаются верстки и на стадии верстки возможно изменение дизайна в некоторых мелочах, так как при верстке нужно соблюдать кроссбраузерность, а современные дизайны могут иметь элементы дизайна, которые не поддерживаются старыми браузерами, но которые до сих пор в ходу адаптивные страницы (см. рисунок 1.2.1).

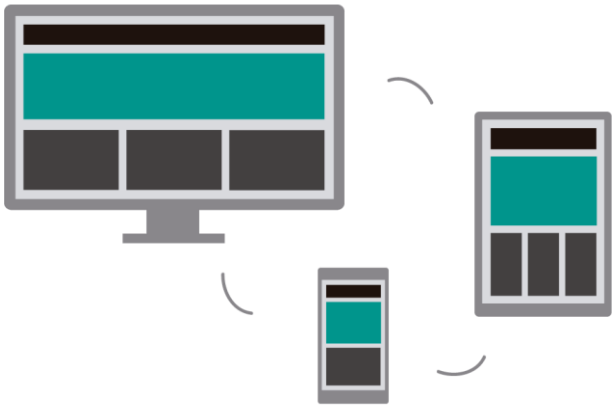

Рисунок 1.2.1 – Адаптивность WEB-приложения

Так же стоит отметить факт, что 85% всех приложений в интернете и не только сайтов, а приложений в целом, все они имеют свой индивидуальный дизайн, тем самым отличаясь от других, и в мире технологий красота имеет очень важную роль. Очень важный факт, что для данного дипломного проекта, дизайн является главной частью особенности для улучшения клиентской базы компьютерного магазина, который в теории может использовать данный сайт, это можно обосновать тем, что сайт выглядит современно и тем самым внушает доверие и показывает качество разработки и отношение к клиенту в целом [4].

1.2.2 Описание клиентской части

Клиентская часть – это основная часть сайта, на которой отображается всё содержимое сайта, которое предназначено для взаимодействия с сервером.

Данная часть сайта является связывающим звеном между сервером и интерфейсом, который использует клиент при использовании того или иного функционального окна. Т.е. можно смело говорить, что клиентская часть сайта, это то, что видит пользователь первым делом, когда заходит на сайт, поэтому создание данной части занимает большое количество времени и

труда, так как если создавать идеальную визуальную часть нужно работать очень усердно и внимательно для соблюдения всех мелочей [14].

Что касается разработки, следует понимать, что создание данного этапа работы, делится на много частей, рассмотрим несколько из них [6]:

– верстка сайта по утвержденному макету;
– создание клиентского функционала, не зависящего от сервера;
– создание запросов к серверу.

Верстка сайта – неотъемлемая часть разработки WEB-страницы, при которой разработчик использует языки разметки, язык запросов и язык определения стилей, как показывает статистика, 90% верстальщиков выполняют одну и ту же рутинную работу тем самым данная профессия – верстальщик, является не востребованной и довольно-таки простой [12]. Но в современном мире все больше и больше набирает оборот использование шаблонов, которые предназначены для упрощения и быстродействие клиент-серверной части сайта. Так же существует разработка по WEB-приложения с помощью компонентов это очень удобно при создании большого проекта, где все контролируется и обновляется через GitHub. И опять же основной целью верстки данного сайта, это создать адаптивную, гибкую и удобную верстку. Адаптивная верстка, кроссбраузерность являются основными атрибутами W3C validates [2].

При создании клиентской части данного диплома была изучена литература, касающаяся SEO оптимизации. SEO оптимизация имеет свои минимальные требования, которые отвечают за положения сайта в поисковой системе браузера и именно от них зависит популярность сайта и восприятия вашего сайта разными браузерами, тем самым SEO оптимизация играет ключевую роль в данном проекте, потому что положение в поисковой системе напрямую зависит с прибылью на сайте, с количеством заказов, а так же с посещаемостью [11]. Стоит отметить тот факт, что существует внутренняя оптимизация и внешняя. Внешняя оптимизация производится разработчиком при разработке, и проверятся на стадии тестирования [4]. Внутренняя это оптимизация самого браузера (см. рисунок 1.2.2).

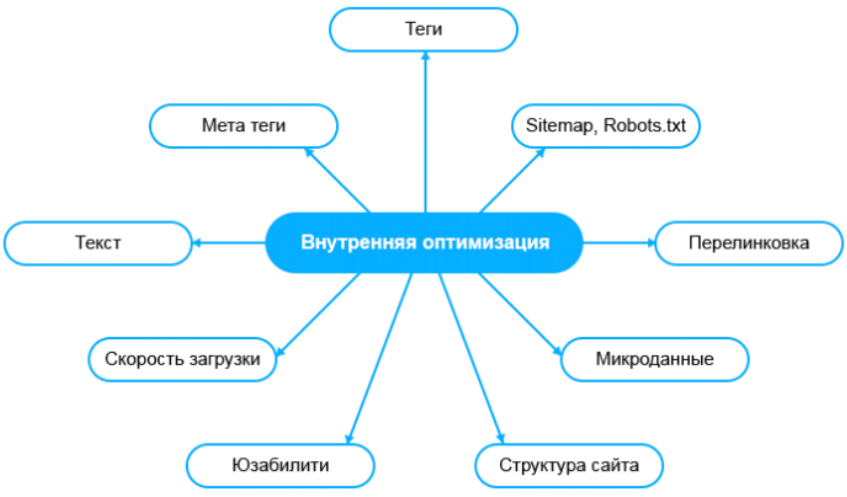

Рисунок 1.2.2 – Используемые параметры SEO продвижения

В конце концов, хочется сказать, что основная часть времени на разработку уходит именно на создания pixel perfect верстки с элементами SEO оптимизации, так как эти вещи имеют очень мелкие детали, к которым нужно относиться очень серьезно [13].

Для чего же нужен функционал на клиентской части, он нужен для того, чтобы пользователь, который использует WEB-страницу, мог подавать запросы, например при нажатии кнопок при заполнении форм, таблиц и т.д. Без клиентского функционала работа страницы сайта была бы невозможна и не имела бы никакого смысла [11]. Так же следует отметить, что все функции, написанные для клиентской части WEB-приложения, были созданы именно так, чтобы пользователю было удобно и просто [8]. При создании данных функций обычно используют имена функций, связанные непосредственно с именами форм и таблиц, за которые они отвечают.

Запросы к серверной части сайта создаются в том же формате, что и язык программирования на котором пишется вся клиентская часть. Как показывает статистика, 79% разработчиков использует Ajax для создания запросов к серверу. Путь от клиентского запроса к базе данных (см. рисунок 1.2.3).

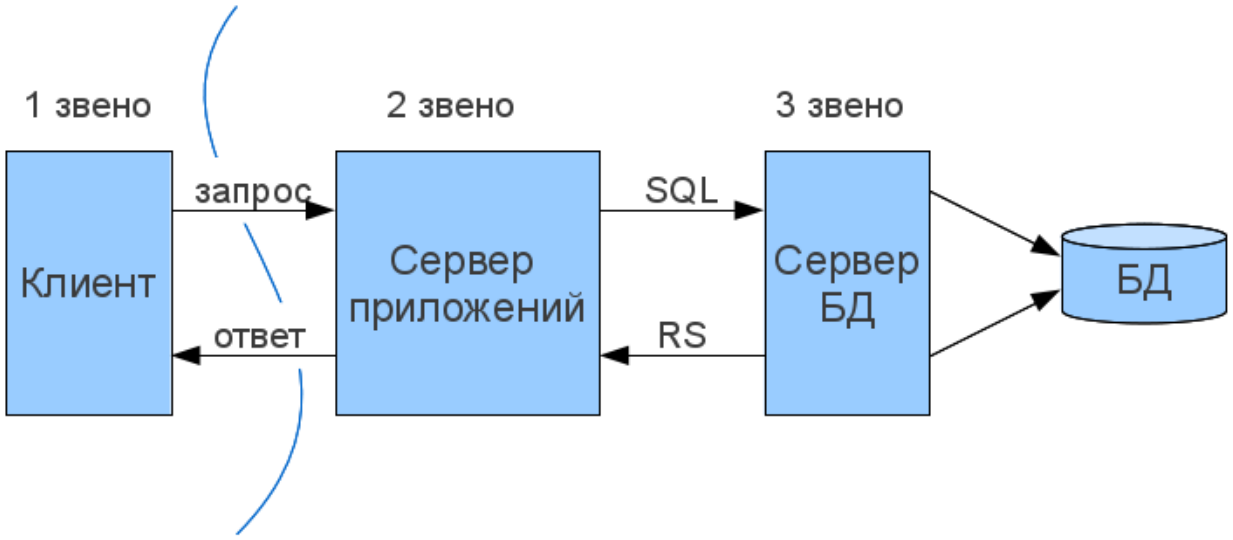

Рисунок 1.2.3 – Путь от клиентского запроса к базе данных

В конце описания клиентской части, можно сказать, что пользователь взаимодействует с сервером именно через эту часть проекта, и она состоит из 3-х основных моментов, в последующих главах будет определена среда разработки, и языки программирования для реализации данной части дипломного проекта [3].

1.2.3 Описание серверной части

Серверная часть WEB-приложения – это одна из основных частей при создании функционирующего сайта, серверная часть отвечает за базу данных и за выдачу данных на клиентскую часть. Если рассмотреть на примере интернет-магазин, который был сделан в данной работе, то можно увидеть огромное количество таблиц связанных между собой разного рода связями и имеющими ключи. То с помощью чего браузер может контактировать с сервером, называется гипертекстовый транспортный протокол, который известен всем как HTTP [11]. Т.е. при нажатии на какую-нибудь ссылку либо же при отправке формы отправляется HTTP запрос на сервер [11].

Например, когда заходишь в адресную строку Google.com и запрашиваешь какой-либо сайт, на сервер Google отправляется HTTP-запрос. Центровой сервер Google получает запрос и должен понять, как его интерпретировать. Сервер Google отправляет обратно HTTP-ответ, содержащий информацию, которую получает мой веб-браузер. Затем он отображает то, что вы запрашивали на странице в браузере.

Веб-серверы часто входят в состав пакета программ, в которых в том числе есть все не обходимые программы для пользования, например Filezila. Все эти программы вместе образуют хост или локальный хост, которым можно пользоваться очень легко и удобно и всем с ним связанным с интернетом и интрасетью, для обслуживания почты, загрузки запросов на файлы протокола передачи FTP с помощью непосредственно Filezila, а также для создания и публикации веб-страницы. При выборе веб-сервера следует учитывать, насколько совместимо он взаимодействует с операционной системой и другими серверами и портами, его способность обрабатывать программирование на стороне сервера, характеристики безопасности и конкретные инструменты публикации, поиска и создания сайтов, которые поставляются с ним. Т.е. при запросе в базу данных отправляется HTTP-запрос, созданный на клиентской части, после этого запрос обрабатывается на серверном языке программирования и в свою очередь создает запрос в базу данных [9]. После чего база отправляет данные на сервер, а сервер создает HTTP-ответ, который идет в клиентскую часть и передает данные с базы данных. После чего разрабатывается вывод полученных данных на пользовательский экран, в том же запросе только создается разметка HTML и в эту разметку помещаются наши полученные данные и выводятся непосредственно на экран пользователя. Быстродействие серверной части во много зависит от выбранного веб-сервера, поэтому установку сервера следует проводить очень внимательно и лично разработчику серверной части WEB-приложения, для этого стоит выбрать язык программирования и совместимый с ним сервер, так же желательно загружать все необходимые модули вручную, чтобы избежать перегруженности сервера и ускорить его быстродействие [12].

Как показывается статистика лучше хранить все данные в базе данных таких как, например MongoDB [19]. Они являются облачными хранилищами и не загружают ваш локальный хост, если таковой имеется, для максимального

ускорения работы серверной части следует размещать сайт в интернет на платной основе, купив при этом хост и домен, но об этом в следующих главах. Безопасность присутствует в любой СУБД, как например СУБД phpMyAdmin.

База данных – сокращенно называется БД, это огромная и организованная структура данных, которая предназначается для хранения данных для образования взаимосвязанной информации и в дальнейшем ее изменение и улучшение. База данных является защищенной частью серверного программирования, подключится к ней, может только человек, занимающийся ее разработкой и серверной частью в целом, для входа в учетную запись обязательно стоит защита паролем во избежание внешнего контакта на хранимую информацию, и на серверной части должна быть 2-йная проверка на подключение. Все базы данных очень удобны в использование каким бы сервером не пользовался разработчик в интерфейсе есть параметры экспорта и импорта для переноса базы на другой компьютер если таковая не находится в общем облаке. Так же стоит отметить, что базы данных имеют огромный спектр настроек. Базы данных используются практически во всех динамических сайтах с большим объемом данных, например интернет-магазины и в обычных информационных сайтах. Тем самым можно сделать следующий вывод, что выполнение клиентской и серверной разработки используя базы данных, то вывод данных производится налету и поэтому данное WEB-приложение можно назвать динамичным сайтом.

Система управления базами данных это одна из самых важных частей при организации базы для взаимодействия с сервером и в дальнейшем с клиентом. Система управления базами данных представляет собой интерфейс, по которому можно легко ориентироваться, по таблицам, по связям между ними все это можно очень просто изменить и создать новые. Как известно база данных состоит из огромного количества таблиц, в которые заносится информация непосредственно с клиентской части, например при регистрации, так же можно заносить данные с помощью системы управления базами данных использую пункт интерфейса. Для примера можно использовать phpMyAdmin для управления базой данных MySQL. Основные особенности системы управления базами данных [18]:

– проходной доступ к БД;

– удобная и обще-централизованная обработка запросов;

– максимальный уровень доступности, безопасности;

– удобное расположение СУБД и базы данных на сервере.

При создании базы данных тоже существует особая методика, при которой нужно выполнять все этапы постепенно, это делается для того, чтобы в результате получить максимально продуманную и логическую базу данных. Все этапы создания базы данных представлены в стандартах W3C, но они не являются обязательными, а просто помогают разработчику безошибочно продумать структуру создаваемой базы данных. Основные этапы создания базы данных по стандартам W3C представлены в виде диаграммы последовательности создания и эксплуатации базы данных, как известно по

стандартам, что бы база данных была логически правильно собрана и имела правильные связи и работала правильно. Реляционная база данных предназначена для просмотра атрибутов таблицы (см. рисунок 1.2.4).

Рисунок 1.2.4 – Этапы создания базы данных WEB-приложения

Так же стоит отметить, что сервер является посредником между более высокоуровневыми технологиями, допустим между почтовыми рассылками и пользователем, который ввел свою почту и подтвердил соглашение на рассылку. После всего сказанного про серверную часть можно сделать вывод, что в современном мире существует большое количество сервером и модулей, которые можно подключить и использовать для взаимодействия между клиентом и базой данных, между клиентом и другими высокоуровневыми технологиями [4].

В итоге можно сказать, что планировка создания WEB-приложения облегчила всю работу в целом, выбор метода разработки позволил разобраться с чего начать выполнение сайта, были определены этапы выполнения, после чего была выбрана среда разработки, а также языки программирования для всех частей описанных выше. Клиент-серверное взаимодействие на страницах является самым безопасным в сети и развивается в непрерывном темпе.

1.3 Выбор средств разработки

Среда разработки – это среда, состоящая из множества программных средств, которые используются разработчиками для создания программного обеспечения в зависимости того, для чего используется среда разработки их разделяют по назначению. Интегрированная среда разработки включает в себя следующие элементы:

– отладчик;

– текстовый редактор;
– компилятор и интерпретатор;
– средства сборки проекта и его использования.

При создании WEB-приложения (сайта) использования компилятора теряет необходимость, потому как мы пересылаем наш код непосредственно через локальный хост непосредственно в браузер. Основными параметрами, по которым было принято решения использовать среду разработки: это браузер классов, удобная сборка проекта, прямой доступ ко всем элементам проекта из самой среды разработки. Подсказки при неполном написании кода, если указан язык программирования, встроенные настройки обновления браузера, терминал в котором можно догрузить необходимые моду, а также встроенные W3C стандарты, для написания правильного и современного кода.

Все параметры, описанные выше, оказались в среде разработки под названием WebStorm. Среда разработки WebStorm – это интегрированная среда разработки для WEB-программирования, сделанная компанией JetBrains. Данная среда разработки написана на языке Java и на основе общей среды разработки IntelliJ IDIA. Она оказалась очень удобной и еще в одном направлении, так как в качестве серверной части разработки была выбрана платформа Node.js, WebStorm дал возможность загрузить все необходимые модули очень удобно и быстро через встроенный терминал [1].

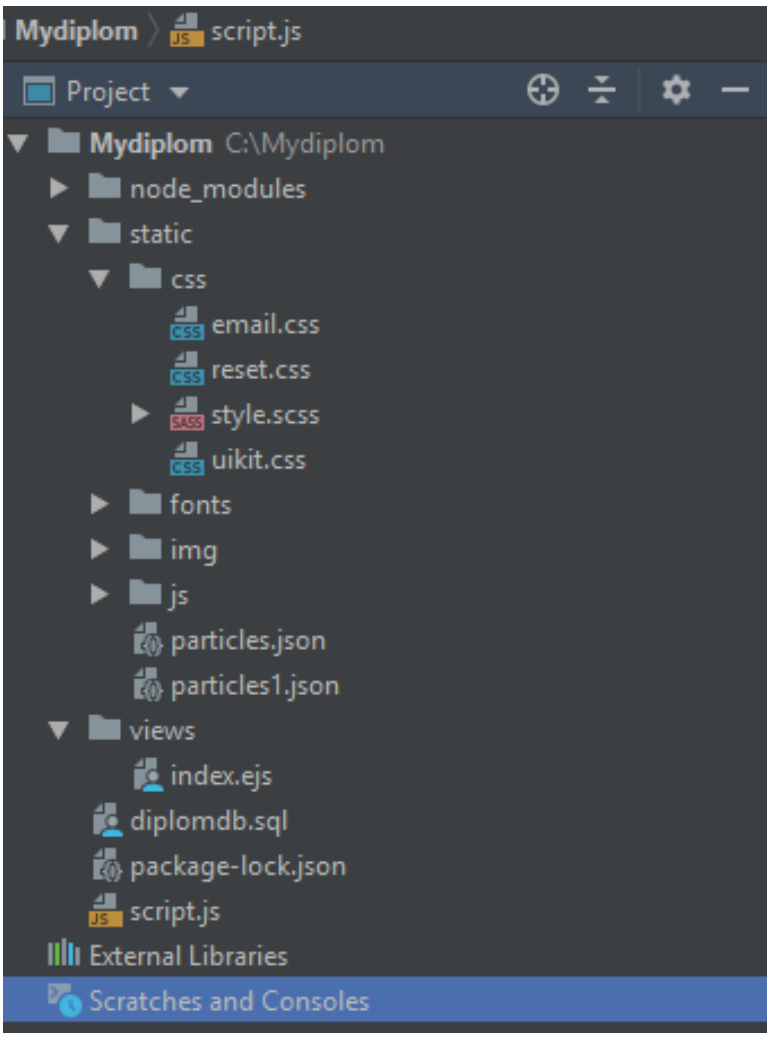

Рисунок 1.3.1 - Сборка проекта в WebStorm

На первоначальном этапе разработке был использован Sublime text 3 в качестве текстового редактора, потому что он имеет самую удобную систему управлением интернет протоколами для улучшения быстродействия и на стадии организации всего проекта и проверки подключения нужных модулей с помощью терминала.

1.3.2 Выбор технологий разработки

Для создания дизайна WEB-приложения на тему конфигуратор технологии использовался Adobe Photoshop в качестве графического редактора и Inkscape Project в качестве векторного графического редактора. Они были выбраны, так как являются самыми современными редакторами при создании дизайна, как показывает статистика, 95% всех разработчиков создают дизайн в Adobe Photoshop [20].

В качестве технологии для создания клиентской части сайта был выбран стек, состоящий из следующих пунктов:

– html – язык разметки;
– css – язык описания внешнего вида;

– scss – препроцессор css;
– bootstrap 4 – набор инструментов для создания сайтов;
– javascript – функциональный язык программирования;
– jQuery – библиотека JavaScript (фреймворк);
– ejs – шаблонизатор;
– ajax – библиотека создания пользовательских интерфейсов.

Все выбранные технологии разработки, в частности, шаблонизатор ejs очень хорошо работают совместно с Node.js тем самым исключает ошибки взаимодействия или внезапные ошибки между сервером и клиентов. В качестве технологий для сервера были выбраны следующие технологии:

– node.js – программная платформа;
– mySQL – в качестве базы данных;
– phpMyAdmin – СУБД для MySQL;
– xampp – локальный сервер;
– apache – HTTP сервер (свободный веб-сервер).

Данный стек для серверной части работает бесперебойно и является самым популярным на 2019 год [11]. Как показывает статистика, сайты использующие данные технологии в 10% ломались, либо на них была утечка данных, т. е хакерские атаки. Конфигурация данных технологий является самой актуально для разработки клиентской и серверной части сайта, т.к браузеры все более и более располагаются к современным технологиям, а старые технологии становятся не актуальными и процесс их разработки сводится к нулю [1]. Для реализации безопасного подключения существуют отдельные модули, которые можно активировать через командную строку NodeJS. Все модули NodeJS используются через командную строку для упрощения установки и эксплуатации.

1.4 Обоснования выбора технологий клиентской части

В данной главе будет описаны все основные технологии, выбранные в качестве используемых технологий для реализации клиентской функционала веб-приложения, так же создания полной клиентской части для пользователя. Все выбранные технологии являются современными и взаимосвязанные собой, т.е имеют максимальную совместимость и быстродействие.

1.4.1 HTML

HTML – это основной язык разметки предназначенный для построения скелета сайта, HTML разметка так же называется версткой и является отдельной профессией, первым делом HTML отвечает за блочную структуру и за адаптивность WEB-приложения [2]. Для облегчения написания на языке HTML существуют множество препроцессоров, которые не требуют закрывающихся тегов, не требуют создания дерева по закрывающимся тегам, язык препроцессоров позволяет в короткие сроки написать разметку ваших блоков в нужном вам виде, не зная тонкостей.

HTML разметка делится на множество видов например: блочная верстка, структурная, семантическая, адаптивная, резиновая. В настоящее время уже существует HTML5, по стандартам данной версии HTML работают все веб-специалисты и в том или ином роде соблюдают данные стандарты W3C. Последняя версия очень удобна тем, что были добавлены очень удобные теги для управления аудио и видео фрагментами разметки, что очень упростило верстку в целом. Все файлы, написанные на языке HTML, используют расширение html или же htm. Так как современные и старые версии Windows допускают разное расширение и понимание файлов. Основная задача HTML – это построить логическую цепочку блоков, используя стандарты адаптивности. Если использовать только язык HTML, то в результате мы получим страничку формата html, на которой будет некрасивый не современный вид, которым очень неудобно пользоваться для добавления стилей и динамики сайтов используются CSS3 и JavaScript, а также их библиотеки и препроцессоры. HTML – является очень старым языком разметки и не считается за язык программирования как например JavaScript, но не стоит забывать что он лежит в основе всех сайтов старых и современных не зависимо от того как они выглядят и функционируют [5]. Данный дипломный проект не обошелся без разметки, так как он является сайтом, при разработке были соблюдены все правила W3C при помощи встроенного валидатора в среду разработки. Код, сделанный по правильной структуре, и по правильной разметки является SEO оптимизированным и тем самым воспринимается браузером лучше и выдает веб-страницу выше других [2].

Как показывает статистика, 100% сайтов используют технологию HTML, если не напрямую, то используют препроцессоры или же создают разметку в JavaScript посредство создания дерева взаимодействия, но в итоге все добавляется в единый файл типа HTML. Согласно стандартам использования HTML существует специальная разметка, которой следует придерживаться при создании файла типа html, пример указан. Стандартная структура разметки это лишь то, что дается для упрощения разработки, но если вы используете какие-либо шаблонизаторы либо используется Bootstrap, тогда необходимо использовать структуру именно этих технологий как, например ejs имеет свою структуру для реализации компонентов [2].

Рисунок 1.4.1 – общепринятая структура HTML документа

Так же не следует забывать, что существует программный язык разметки XML, который основан на языке HTML, в отличии от стандартного языка разметки он имеет тип расширения .xml, а также является программным языком разметки для создания интерфейсов программных продуктов, а не для браузерных приложений. То, как браузер понимает ваш HTML, отвечает сам браузер, к примеру, Internet Explorer, который пользуется очень мало. Как показывает статистика, 9% всех интернет пользователей, использующих Internet Explorer, не поддерживают многие современные теги, которые разработчик внедряет, соблюдая стандарты W3C.

Для решения этой проблемы, связанной не только с HTML5, но и с другими современными версиями стека для программирования существует кроссбраузерная верстка, сутью которой является создать веб-страницу, которая будет поддерживаться всеми браузерами на просторах интернета. Для этого существуют множество вариантов написания кода, как например, для создания сайта для всех браузеров в CSS существуют дополнительные префиксы или псевдоклассы, которые обращаются к тому, или иному браузеру.

Ваш браузер отображает код с помощью механизма рендеринга, который обрабатывает и представляет интерпретацию кода веб-сайта на вашем экране. Internet Explorer, Mozilla и Google Chrome используют разные механизмы рендеринга. Интерпретация является ключевым словом здесь. В частности, CSS и HTML (коды, обычно используемые для форматирования веб-сайтов) не отображаются одинаково во всех браузерах. В результате веб-сайт может показаться слегка отключенным, а иногда и вовсе отключенным при просмотре в разных браузерах [20]. Именно поэтому во избежание проблемы с работой сайта на разных браузерах было принято решение

верстать проект для всех веб-браузеров одновременно с использованием префиксов и псевдоклассов, которые в свою очередь определяют код к определенной системе вывода информации [2].

Существуют способы кодирования, которые делают ваш дизайн похожим в разных браузерах, когда это происходит. Поэтому, если что-то отображается неправильно, ведет себя смешно или не в нужном месте на вашем сайте, вы можете настроить код так, чтобы он правильно отображался в браузере, в котором происходит ошибка.

1.4.2 CSS

CSS – это язык стилей, от которого зависит внешний вид вашего сайта вплоть до всплывающих окон. Без CSS создания верстки по шаблону было бы невозможным, а все сайты были бы однотипными и не воспринимались бы как WEB-приложения, не смотря на функциональность [3]. CSS никак не связан с HTML, но контактируют они с помощью: классов, идентификаторов, или же в целом с тегом. В системе присваивания стилей существует своя иерархия по важности на первом месте стоит включения стилей в самом HTML коде, либо же в самом теге в качестве атрибута style. Далее идет подключение стилей через идентификатор и на последнем месте стоит подключение через классы. Самым распространенным способом присвоения стилей является назначения класса в HTML и после этого по этому классу прописывать стили в файле формата .CSS. Для написания стилей существует множество препроцессоров, которые могут объединять языки программирования, в таблицу стилей компилируя это все в итоге в обычный всем понятный формат [3].

Обычно создание стилей занимает большое количество времени, но при использовании классов как основных селекторов можно создавать классы определенного типа и обращаться к ним разными блоками. Т.е один класс может служить стилями для огромного числа блоков или другого типа контейнеров. Так же через классы можно обращаться к дочерним элементам разметки, допустим, если блок содержит другой блок, мы можем обратиться к нему и задать стили через класс внешнего блока и селектором, который находится внутри внешнего блока, допустим текстовым контейнером. Существует огромное количество стилей, которые применяются по своему назначению, как например блочные элементы они принимают только классы, которые применимы к блочным элементам, а строчные соответственно только строчные классы. Язык CSS имеет возможность преобразовать ваш элемент в элемент другого типа с помощью свойства display. Все что описывает класс, называется свойством, а сам класс может в свою очередь является селектором, по которому классы назначаются. Позиционирование блоков напрямую зависит от свойств, которые мы задаем в стилях, например расположение блока по центру. Позиционирование блоков является основным элементом создания резиновой верстки или как ее называют в простом языке сделать

сайт адаптивным. Адаптивный сайт подразумевает плавное позиционирование блоков, не вылезая за контуры главного экрана на всех устройствах и приборах, поддерживающих веб-браузеры. Так же стили содержат такие псевдоклассы, которые отвечают за динамичность сайта, например при наведении, или при выделении с помощью данных псевдоклассов можно применять любые классы, т. е некоторые псевдоклассы CSS работают как события в JavaScript.

Анимация CSS3 включает в свой спектр большое количество функций, которые отвечают за динамичность сайта, по последней статистике W3C известно, что 95% всей анимации создается на CSS3 либо через JavaScript с использованием библиотек, содержащих CSS стили. Использование CSS в современном мире очень неудобная вещь для создания адаптивных сайтов, а короткое время для этого используются препроцессоры как, например SCSS.

1.4.3 SCSS

SCSS – препроцессор CSS, является одной из интерпретации языка стилей SASS, предназначенный для упрощения работы с CSS. Имеет огромное количество функций, с помощью которых можно присвоить какое-либо свойство переменно и вызывать его в нужном месте, так же имеется наследование классов от родительского элемента, и импорт внешних ресурсов. Самой основной задачей SCSS является преобразовать код на простом языке в обычный CSS с помощью встроенного компилятора, который переносит наш код в сжатый. Сжимание кода является главной в разработке сайта, потому как обычно создание стилей занимает порядка 10 тысяч строк кода. Проходит путем импорта огромного количества файлов в один, который является основным и подключен к HTML разметке. При создании верстки нужно учитывать следующие факторы это .psd макет, который включает в себя шрифт, картинки, и все необходимые детали для создания идеального шаблона по макету. Препроцессор SCSS имеет параметры объединения всех файлов макета с PS для удобной работы в одном интерфейсе без постоянных скачиваний и препарирований макета. Можно смело сказать, что HTML и CSS являются самыми необходимыми языками разметки и стилей, заменить которые нельзя. Мы можем изменить язык написания клиентский функций или использовать какую-нибудь библиотеку, но в основе каждого сайта лежит дерево разметки и огромное количество стилей подключенных к ним с помощью селекторов или же с использованием прямых атрибутов для переназначения того или иного класса. Все эти моменты открыты в общем доступе, каждый может исследовать любую страничку в образовательных целях. Следует отметить тот факт, что препроцессоры CSS и HTML имеют открытый код доступа, с помощью которого любой пользователь или разработчик может с легкостью изменить конфигурации компилятора [3].

1.4.4 Bootstrap

Bootstrap – библиотека, предназначенная для упрощения создания стилей и разметки, так же содержит основные реализации функций JavaScript. На данный момент Bootstrap является самой используемой библиотекой, которая содержит в себе огромное количество реализованных классов с возможностью изменить любой фрагмент класса, сделав его основным и использовать как шаблон.

Веб-дизайнеры обычно проходят этап, называемый каркасным, который включает рисование того, каким должен быть макет или структура веб-страницы. Это помогает определить, как представлена информация и как взаимодействуют различные биты информации. Поскольку эта запутанность информации является центром каркаса, здесь не уделяется внимания шрифтам, цветам или графике [15].

Короче говоря, создание каркаса — это способ заставить веб-автора ответить на вопросы относительно того, что он хочет, чтобы его веб-сайт делал. Что они хотят, чтобы их пользователи делали. Какую информацию они хотят раскрыть.

Сетки помогают организовать визуальную информацию. Если элементы на странице (включая веб-страницу) не совпадают, страница может быть запутанной и утомительной для чтения [20]. В Bootstrap, когда div имеет класс row, он запускает систему сетки Bootstrap, которая генерирует сетку шириной в двенадцать столбцов, что позволяет нам распределять эти столбцы по своему усмотрению.

Большая часть сетки Bootstrap заключается в том, что она меняет свой размер в зависимости от ширины браузера. Например, то, что может быть двумя столбцами на ноутбуке или планшете, может выглядеть как один столбец (с двумя столбцами, расположенными друг над другом) на смартфоне [3].

Каждый столбец в Bootstrap обычно также является div с классом, который использует синтаксис col-sz-n. Здесь sz относится к точке останова, то есть, к какому размеру столбцы начинают распространяться [2]. Точками останова могут быть sm (телефоны в горизонтальном режиме), md (планшеты), lg (рабочие столы) и xl (большие рабочие столы), с другой стороны, представляет собой целое число от 1 до 12, которое описывает, сколько столбцов сетки должно быть шириной div.

Можно сделать следующие выводы, что использование сетки Bootstrap является аналогом flex, который является основным моментом адаптивной верстки под все устройства в CSS. В данном дипломном проекте Bootstrap использовался исключительно в качестве библиотеки иконок, которые включены в стандартный набор, а также в виде CDN подключения некоторых компонентов [20].

1.4.5 JavaScript

Язык программирования JavaScript – является самым прогрессивным языком программирования для создания WEB-приложений, имеет огромное количество библиотек, очень просто синтаксис и очень удобное использование функций [13]. За последние 3 года JavaScript стал основным языком программирования на клиентской части сайтов, и как показывает статистика Google, он является самым удобным языком программирования для браузера. На данном языке можно сделать все что угодно на вашем сайте, так как он имеет в себе и стилизацию и функциональность, так же по JavaScript есть огромное количество литературы и справочников, что очень удобно при изучении [8]. Функциональный язык программирования JavaScript оказался очень удобным для создания конфигуратора, так как создание его функционала и переходов полностью написаны на этом языке программирования с использованием библиотеке jQuery [13].

1.4.6 jQuery

jQuery – библиотека JavaScript, является самой используемой и самой простой библиотекой, содержащей все функции основного языка программирования клиентской части WEB-приложения. На данном языке написаны 90% программ, которые можно установить на сайт. Особенностями jQuery является то, что он имеет очень простой синтаксис, и краткое название функций, которые в JavaScript могут представлять огромное количество строк кода. Эти строки кода содержат все события JavaScript, если говорить простым языком jQuery, то же самое, что и JavaScript с простым синтаксисом и поэтому разработчики приходят к тому, что jQuery очень удобна и проста [13]. С помощью jQuery были написаны все основные функции для конфигуратора, тем самым количество написанного кода было сокращено в два раза.

Вызов jQuery событий или же вызов для обработки при действии отличается от JavaScript основной версии только тем, что обработчик событий данной библиотеки является универсальным [13]. Не требует для своего обращения полной структуры обращения функций к документу по классу или идентификатору, что в свою очередь упрощает обработку событий примерно в 2 раза. Данная библиотека имеет огромное количество модулей, активация которых занимает не больше секунды, если вы умеете пользоваться npm.

1.5 Обоснования выбора технологий серверной части

В данной главе будет описаны все основные технологии, выбранные в качестве используемых технологий для реализации серверной части функционала веб-приложения, для безопасной передачи и записи данных, а также организованность запросов и быстродействие. Все выбранные технологии являются современными и взаимосвязанные собой, т.е имеют максимальную совместимость и быстродействие.

1.5.1 Node.js

В качестве серверного языка программирования была выбрана платформа Node.js, которая на данный момент является самой популярной платформой для серверного программирования, серди разработчиков.

NodeJS — это исходный код, основанный на платформе Javascript V8, он используется для создания веб-приложений, таких как страницы видеоклипов, форумы и особенно узкие социальные сети [1]. NodeJS — это открытый исходный код, широко используемый тысячами разработчиков по всему миру. NodeJS может работать на многих различных платформах ОС от WIndows до Linux, OS X, так что это также является преимуществом. NodeJS предоставляет богатые библиотеки в форме Javascript. Различные модули упрощают программирование и сокращают время на самом низком уровне [11].

Когда дело доходит до NodeJS, вы должны думать о реальном времени.

Например, когда вы просматриваете Facebook, каждый раз, когда вы комментируете или ставите лайк в определенной теме, сразу же владелец темы и комментаторы получат уведомление, которое вы прокомментировали. Если вы думаете, что Facebook использует Ajax, то вы ошибаетесь, если они используют Ajax, то сервер немедленно затеряется и не сможет дать ответ по запросу, представленному в количестве тонн. Т. е можно сделать следующий вывод, что Node.js отвечает за огромную работу, например, достаточно привести в пример socket, что бы стало понятно, что серверная часть работает и обрабатывает запросы в режиме реального времени как приведено в примере сверху с Facebook. Т.е сервер с помощью технологии socket обрабатывает и в тот же момент присылает уведомление. Серверный язык программирования Node.js имеет огромное количество модулей, перечислить половину из них невозможно. В данном проекте использовался закрытый модуль передачи электронного сообщения при запросе на его отправку [11].

Так же стоит отметить тот факт, что Node.js загружает все модули через npm в командной строке для чего достаточно просто создать переменные и присвоить им имена нужных нам модулей. После чего прописать в командной строке npm install и ваши модули, прописанные на сервере, загрузятся и распакуются, и будут полностью готовы к использованию. В дипломном проекте вся серверная часть написана на языке программирования Node.js все сессии, передача данных и запросы обрабатывались на сервере [1]. Так же показана вся простота ответа на запросы клиента, что является самым важным моментом при создании серверной части дипломного проекта.

Все действия, описанные в Node.js можно просмотреть в консоли GitHub или же в стандартной консоли, предоставляемой Windows.

1.5.2 MySQL

Разработка базы данных проводилась с использованием СУБД phpMyAdmin, посредством него мы, с легкостью используя стандартный интерфейс, создали базу данных и выбрали ей кодировку для чтения данных из таблиц и тем самым обеспечили внос в базу данных строк, который подходят данной кодировке, что очень удобно. Что бы использовать phpMyAdmin нам понадобился локальный сервер или сборка, содержащая локальный сервер. После чего мы получаем доступ к базе данных, которую создали ранее, по тому паролю и логину, который указываем в подключении на серверной части реализации дипломного проекта.

Локальные серверы работают очень быстро и без ошибок, если не использовать порты для сервера и для еще чего-либо. После создания базы данных начинается создание таблиц и связей между ними перед непосредственно созданием таблиц и связей необходимо проектирования базы данных в специальных приложениях, где можно продумать все связи и поля. Все поля в базе имеют атрибуты, которые отвечаю, какого типа будет поля текстовым или целочисленным [18]. Сама база данных использует язык программирования SQL, который используется при программировании всех баз данных, отвечает за выдачу прав, а также за безопасность хранения данных.

В данном проекте было создано большое количество таблиц, имеющих связь между собой, все они были протестированы и проверены на безопасность тем самым можно сказать, что лучшим решением при использовании Node.js был выбор MySQL, так как создание базы данных очень простое и невозможно совершить какие-либо ошибки. В то же время база данных была использована для того, чтобы загружать в нее информацию непосредственно через регистрацию, это очень тонкий момент, потому что база данных не должна принимать информацию всю подряд для этого нужно создавать двойную валидацию со стороны клиента, а также со стороны сервера. Эти параметры обеспечат базе данных компактность и высокую работоспособность [17].

1.5.3 XAMPP

Является на данный момент самым современным набором, для создания локального приложения, он имеет самый удобный интерфейс использования и управления конфигурациями посредством языка PHP. В его конфигурацию входят огромное количество вспомогательных программ для организации и тестирования сайта на локальном уровне и в дальнейшем загрузки сайта на купленный домен. С помощью удобного переключателя активности можно управлять базой данных, полная отдача и принятие данных с базы зависит от локального сервиса, который предоставляется XAMPP, чем же он отличается от своих аналогов, в современно мире существует более пятидесяти аналогов подобной сборки, но, как известно не все они работают хорошо. Самой распространенной ошибкой при использовании других открытых сборок для

создания локального сервера является занятость портов, порты могут быть заняты другим программным обеспечением. В сборке XAMPP существует автоматизированная настройка портов согласно их эксплуатации, что облегчает работу системы и помогает устранить все ненужные конфликты и задержки на стадии установки и подключения базы данных через phpMyAdmin.

Для успешной работы в данной конфигурации программ не потребовалась настройка портов и настройка открытия локального сервера через URL, тем самым можно открыто сказать [17], что XAMPP является самым лучшим выбором среди всех локальных сборок, предоставляемых в сети интернет.

Интерфейс XAMPP имеет встроенный терминал событий, который реагирует на каждый запрос, связанный с локальным сервисом в той или иной степени. Интерфейс так же содержит возможности настройки конфигурации под разработчика, что очень удобно и практично. Если существует какая-либо ошибка, связанная именно с портами или с подключением разработчика к локальной системе терминал, немедленно выдает код ошибки [19]. С помощью данного кода можно посмотреть суть ошибки и понять в чем проблема если в порте, то нужно изменить его на 1 больше, чем он стоял до этого, обычно это мелки ошибки (см. рисунок 1.5.3).

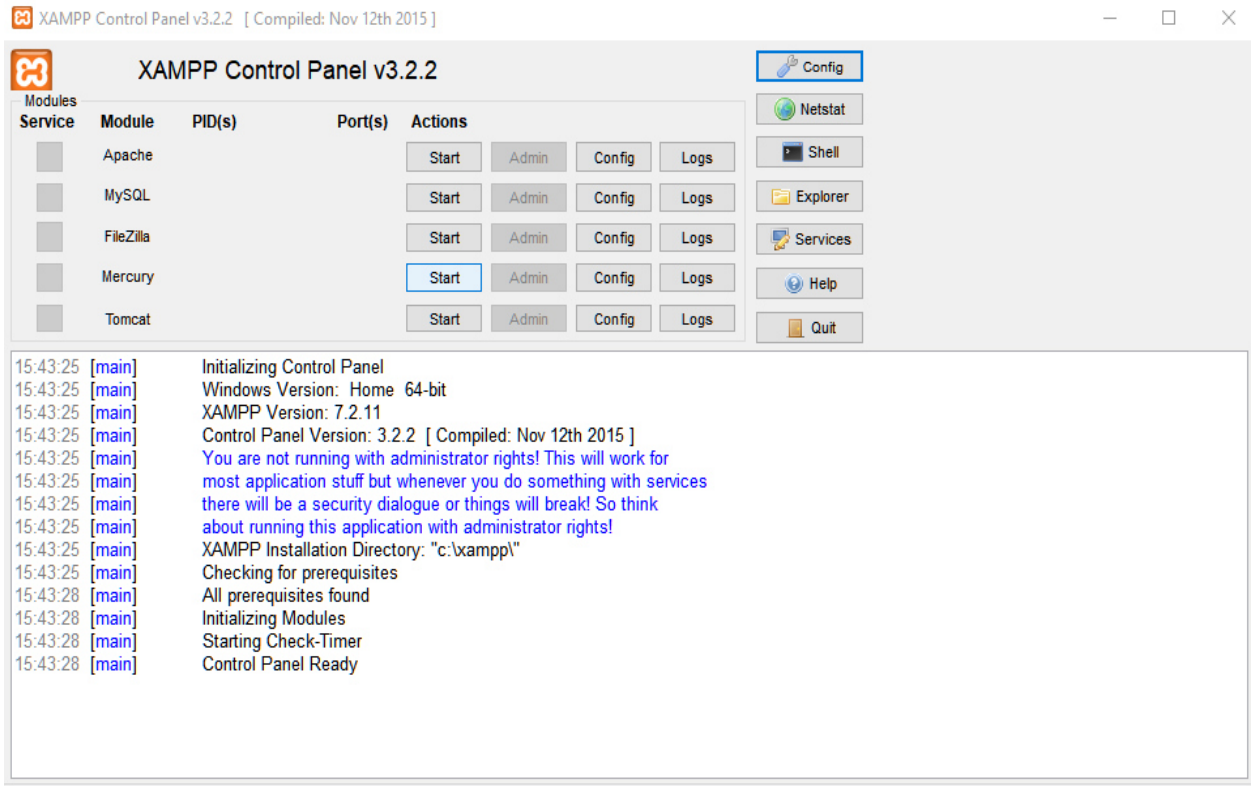

Рисунок 1.5.3 – Панель управления XAMPP v3.2.2

Все событие произошедшие в какой-либо технологии XAMPP сразу же приходит в лог событий, что позволяет очень быстро найти ошибку.

Можно сделать следующие выводы, что использование XAMPP автоматизирует работу всей конфигурации, или пакета программ, которые мы используем непосредственно на локальном сервере [17].

1.5.4 Apache

Данная технология является, как и HTTP протоколом, который отвечает на запросы, так же является HTTP протоколом, который отвечает за вывод информации, т. е за ответ, поэтому данную технологию можно открыто назвать HTTP-сервисом. Стоит понимать, что данная технология является основной в работе клиент-серверной части проекта, и отвечает за безопасность, за выдачу или за внос той или иной информации в базу данных посредством запросов. Так как мой проект имеет огромное количество запросов в базу данных и постоянно отправляет их в режиме настоящего времени, можно сделать вывод, что эта технология является одной из основных для реализации данного дипломного проекта. Последняя версия Apache всегда автоматически устанавливается, что очень удобно, это возможно благодаря тому, что мы используем XAMPP.

Правильная настройка веб-сервера Apache может быть чрезвычайно важной, поскольку иногда она может предотвратить определенные атаки веб-приложений, даже если в веб-приложении есть уязвимость. В этом части будет описано, как настроить configure apache для отправки заголовков HTTP, связанных с безопасностью, в своем ответе и скрыть конфиденциальную информацию от заголовков ответа сервера [18].

В следующей части будет обоснованная настройка виртуальных хостов с активированным протоколом SSL, переписывание запросов и перенаправление. Если вы тщательно проверяете HTTP-ответы, поступающие с веб-серверов с Google, Facebook [19].

2 Проектирование программного продукта

Основная задача при проектировании приложения, имеющего базу данных, является создать модель базы данных и непосредственно связей между ней ее заполнение, после чего идет процесс реализации базы, и подключение к серверной части разработки. Так как темой дипломного проекта является WEB-приложение, следует отметить, что оно работает по принципу клиент-сервер. При проектировании следует учесть 2 вещи проектирование клиентской части т.е структура сайта и его возможности, и серверной части, под проектирование серверной части подразумевается проектирование базы данных посредством СУБД.

Проектирование функциональной части WEB-приложения представляет собой две модели эксплуатации сайта, на которых показаны все возможности. Посредством данных моделей можно определить какие функциональные

возможности будет иметь сайт при реализации клиентской части запросов, основанных на функциональной зависимости между клиентом и сервером.

2.1 Проектирование структуры клиентской части

Проектирование структуры клиентской части сайта, это проектирование сайта в виде структуры имеющихся на сайте блоков, т. е можно сказать, что проектирование клиентской части — это создание технического задания того, что должно быть на главной страницы, так как основная часть сайта является листингом, на структуре представлены блоки, которые входят в главную страницу. После проектирования того, что должно быть представлено главной страницы WEB-приложения, начинается этап создания общего макета, состоящего из блоков и примерной структуры сайта. После создания примерного вида общего макета начинается этап создания макета сайта в Photoshop и после начинается проектная реализация.

В части будет рассмотрено 2 структуры WEB-приложения, первое это создание структуры клиентской части путем создание примерных информационных блоков, и функциональная структура WEB-приложения.

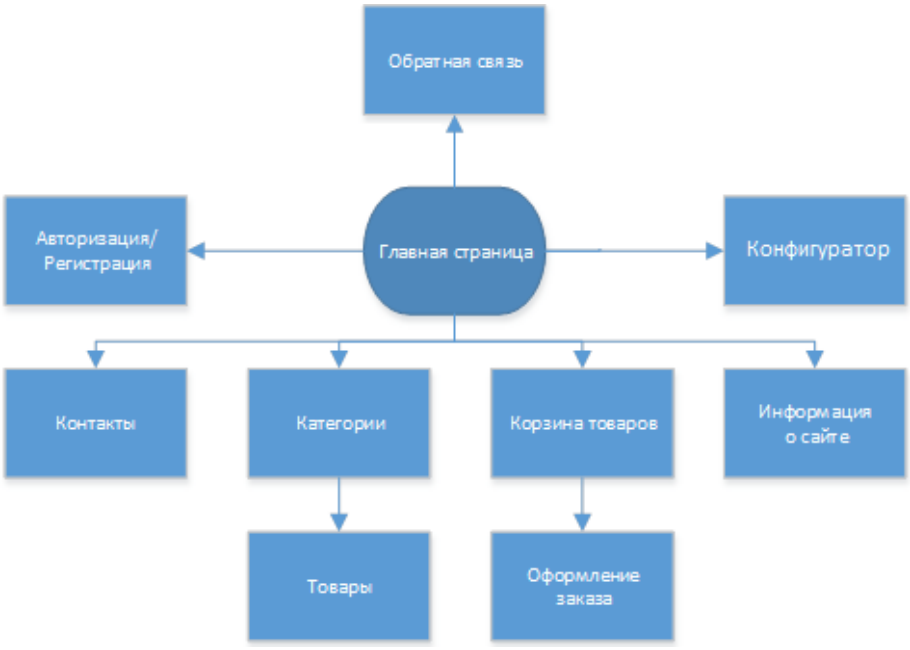

Рисунок 2.1.1 – Блочная структура веб-приложения

Проектирование функциональной структуры WEB-приложения стало следующей задачей после того, как стало понятно какие блоки потребуются для создания современного листинга. Исходя из того, что регистрация является необязательным фактором для сборки компьютера, но обязательным для его покупки и оформления заказа, было спроектировано две функциональные структуры WEB-приложения.

Первая функциональная структура представляет структуру с параметрами входа, то есть регистрация, если пользователя не существует, и

вход в сессию, либо же сразу вход, если пользователь регистрировался ранее. При входе появляется возможность, оформления заказа, и отправки заказа на почту менеджеру или консультанту данного сервиса. Обратная связь имеет доступ на запрос аналогично оформлению заказа (см. рисунок 2.1.2).

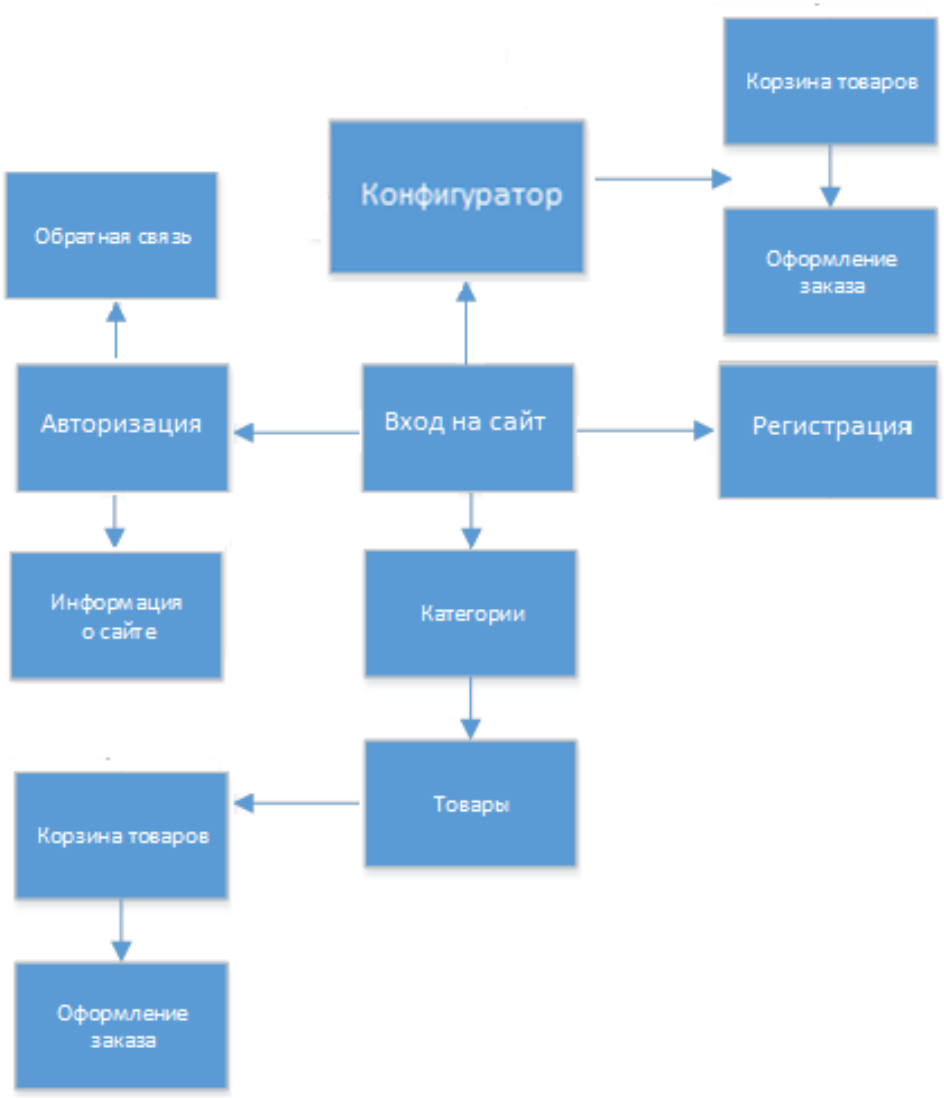

Рисунок 2.1.2 – Функциональная структура с учетом авторизации

Вторая функциональная структура представлена, если пользователь не прошел авторизацию и является незарегистрированным и его сессия проходит в качестве гостя. Некоторые функции будут недоступными, пока клиент не пройдет этап авторизации. Т.е. можно сделать вывод, что вторая схема будет представлять собой интерфейс, который будет сводиться к регистрации и авторизации пользователя, так как это действие является обязательным для выполнения некоторых функций веб-страницы. Но стоит отметить что большая часть WEB-приложения является общедоступным и не зависит от того зарегистрирован ли пользователь или нет [19].

Создание данных моделей стало возможно благодаря сервису WEB-model, который позволяет создать любую структуру для сайта.

Рисунок 2.1.3 – Функциональная структура без учета авторизации

Как можно увидеть функциональная структура без учета авторизации не имеет параметров оформления заказа, и при нажатии оформить заказ в корзине система возвращает обратную связь, которая не переход на меню отправки почты, а перекидывает неавторизированного пользователя к форме регистрации [18]. Обратная связь будет перекидывать пользователя к форме регистрации до тех пор, пока пользователь сервиса не пройдет регистрацию, либо авторизацию.

Посредством данных функциональных структур появилось представления, как взаимодействовать между клиентом и сервером, как создавать запросы в базу данных, при каких событиях выводить данные и при каких делать исключения. После проектирования клиентской части можно преступать к реализации клиентской части, а именно можно начинать рисовать макет в Photoshop, после чего начинается этап реализации, как уже было описано адаптивная верстка [2]. После создания клиентской части на определенные события пишутся запросы в JavaScript и эти запросы начинают обрабатываться на сервере, но сервер должен контактировать с базой данных,

которую нужно создать и сделать логические связи между табличками для этого была выполнена часть проектирования базы данных [6].

2.2 Проектирование базы данных

Если речь идет об интернет-магазине или о каком-то количестве товара, который нужно выбрать в той или иной степени, то без подключения базы данных не обойтись. Данный проект использовал одну базу данных, состоящую более чем из 20 таблиц каждая из которых, имеет свои имена и атрибуты. Огромное количество данных было подключено только к конфигуратору персональных компьютеров. Правильность работы базы данных зависит от того какие запросы обрабатывает сервер и в какие таблицы заносятся данные и наоборот, когда данные выводятся в браузер пользователю [16].

Проектирование базы данных является очень важным этапом при разработке клиент-серверного приложения. База данных данного веб-приложения имеет название diplomdb, она была разработана посредством СУБД phpMyAdmin через сборку локального сервера XAMPP. Для каждой категории товаров была создана таблица [17]. В итоге получилось порядка 21 таблицы. Все таблицы заполнены данными о том или ином элементе конфигурации вручную или же такая таблица как userdata является таблицей, в которую заносятся все данные зарегистрированного пользователя. Данная таблица имеет двухэтапную защиту и кодировку для скрытия данных зарегистрированных пользователей [19].

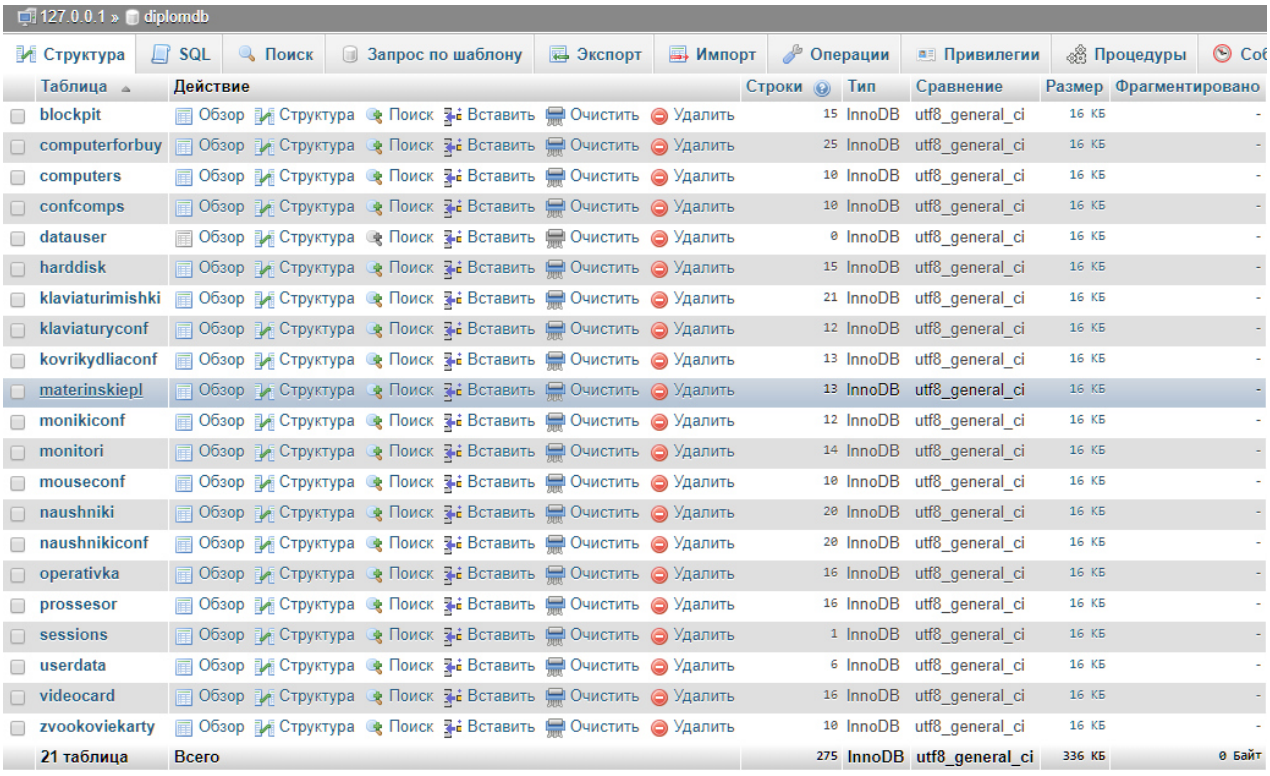

Рисунок 2.2.1 – База данных diplomdb

Главной особенностью базы данных является кодировка для поддержки русского и английского языка используется общепринятая кодировка utf8_general_ci, которая является принятым стандартом в русскоговорящем сегменте. База данных имеет логические соединения для вывода правильной информации один ко многим или один к одному [12]. Все атрибуты таблиц имеют определенные параметры и кодировку.

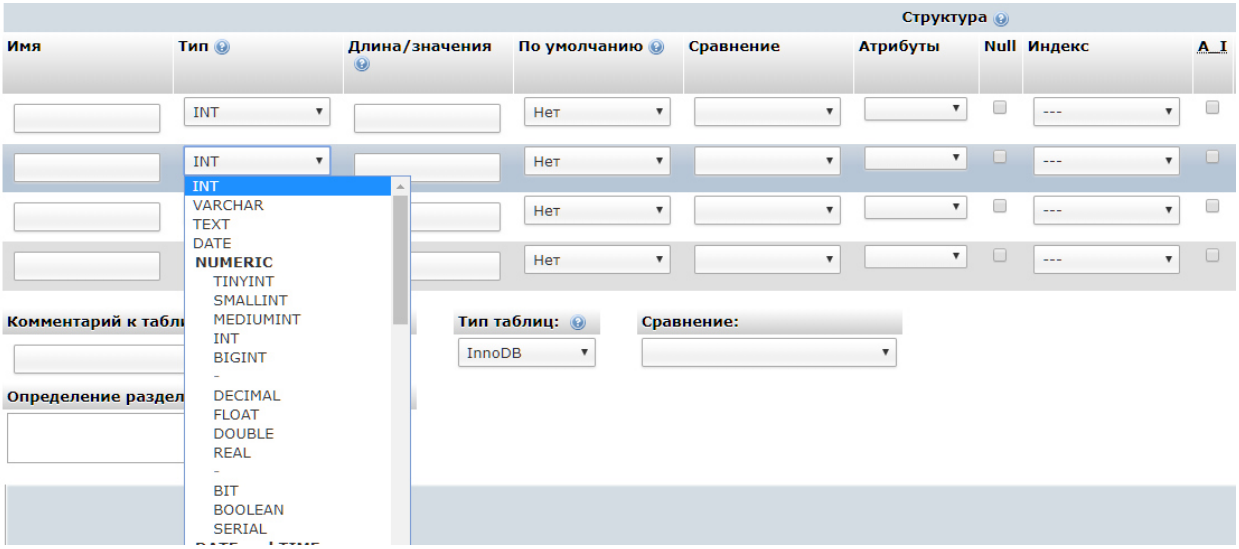

Рисунок 2.2.2 – База данных diplomdb

3 Разработка программного продукта

Для создания WEB-приложение, как было указано выше, разработка разделилась на два этапа первый это создание клиентской части приложения и второй этап создания серверной части приложения.

В первую очередь нужно создать проект, структура состоит из множества папок, содержащих модули и папок отображение, существуют статические папки, в которых находятся наши стили, и папки отображение, где находится разметка HTML. Все серверные команды и функции реализованы в script.js.

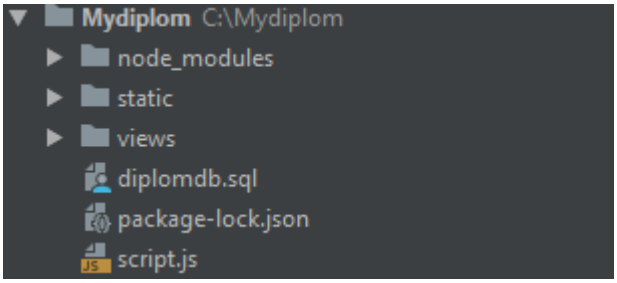

Рисунок 3.1 – Структура проекта Mydiplom

Файлы, которые подключаются к нашей разметке, а именно к шаблонизатору ejs находятся в папке static.

Что бы сервер создался, мы должны так же отдельно установить Express, который позволяет серверной части создаться и работать без каких-либо ошибок, после чего можно приступать к настройке самого сервера и клиентской части разработки дипломного проекта. Перед установкой, каких-либо модулей обязательно провести установку самой программы Node.js. Данный модуль Express отвечает за все приходящие запросы и служит заменителем Apache, соответственно тоже является HTTP – сервером. Поэтому установка Express проходит отдельно от всех модулей и проводится напрямую через командную строку (см. рисунок 3.2).

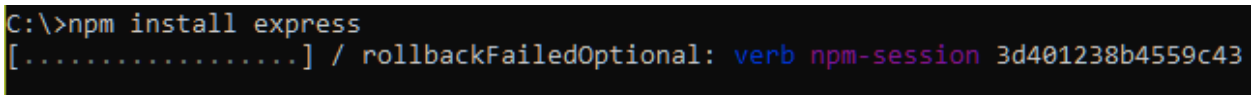

Рисунок 3.2 – Структура проекта Mydiplom

В папке static находятся директория стилей css содержащая наши стили формата css и scss. В директории fonts содержатся все шрифты, которые были выбраны в качестве основных для разработки приложения. Директория img содержит все фотографии, которые мы используем в базе данных и те фотографии, которые используем непосредственно в самой разметке.

Директория JS содержит файл формата js, который содержит все функции и реализацию клиентской части веб-приложения (см. рисунок 3.3).

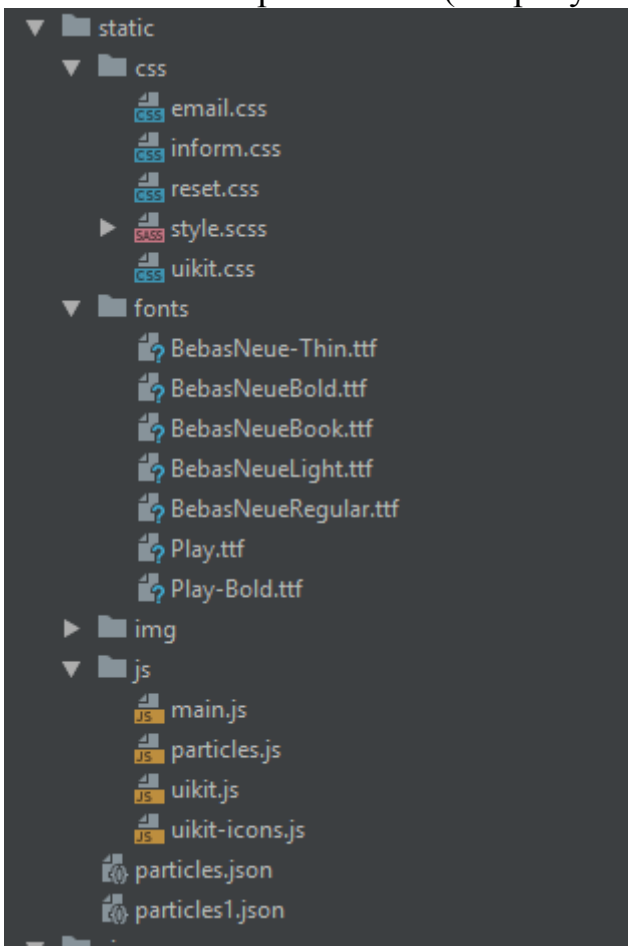

Рисунок 3.3 – Структура директории static

Сама разметка находится в директории отображения шаблона ejs, который содержит разметку нашей страницы, разметка написана на стандарт ном языке построение структуры сайта HTML. Формат ejs шаблонизатора ejs понимает язык разметки HTML и конвертирует его, при помощи встроенных компиляторов (см. рисунок 3.4).

Рисунок 3.4 – Структура директории views

3.1 Настройка сервера

Для успешной работы нашего проекта необходимо загрузить модули, которые отвечают за сборку проекта данного типа. Для этого необходимо прописать переменные и присвоить им название необходимых модулей на

серверной части реализации. Прописанные переменные примут параметры модулей, и при загрузке будут ориентироваться на них (см. рисунок 3.1.1).

```
var express = require('express');
var mysql = require('mysql');
var bodyParser = require('body-parser');
var SessionStore = require('express-mysql-session');
var connect = require('connect');
var router = express.Router();
var cookie = require('cookie');
var cookieParser = require('cookie-parser');
var session = require('express-session');
var app = express();
var http = require('http');
var server = http.createServer(app);
var rand,mailOptions,host,link,auth;
var nodemailer = require('nodemailer');
var fs = require('fs');
```

Рисунок 3.1.1 – Присвоение к переменным модулей

После чего необходимо прописать команду npm install и перечислить все созданные, переменны, после данных манипуляция все необходимые модули загрузятся в директорию node_modules.

Для успешной работы сервера необходимо создать и подключить базу данных, используя локальный сервер, нам необходимо включить все необходимые программы, используя интерфейс XAMPP, отвечающие за работы базы данных (см. рисунок 3.1.2).

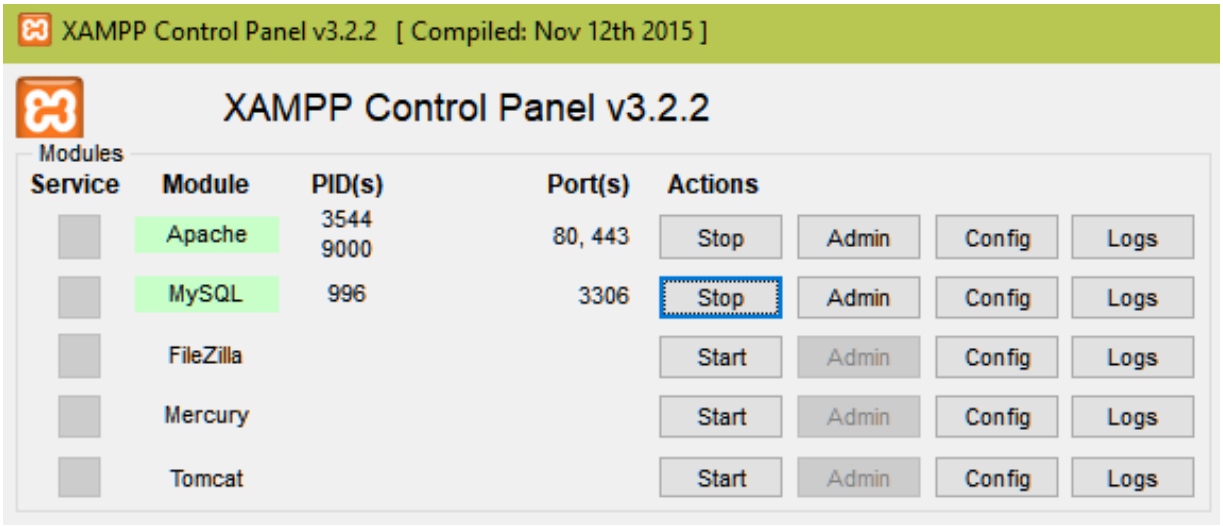

Рисунок 3.1.2 – Интерфейс управления XAMPP

После чего необходимо создать базу данных, используя СУБД phpMyAdmin, для этого переходим в административную панель, нажав на

admin в панели управления XAMPP. После создания базы данных без таблиц нам необходимо подключить базу к серверу, для этого используем node.js и подключим базу (см. рисунок 3.1.3).

```
app.set('view engine', 'ejs');
app.use('/static', express.static('./static'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended:false}));
var options = {
host: 'localhost',
user: 'root',
password: '',
database: 'diplomdb'
}
var sessionStore = new SessionStore(options)
var cfg;
app.use(session(cfg ={
key : 'session_name',
secret : 'session_secret',
resave : true,
store: sessionStore,
saveUninitialized: true ,
cookie: {
}
}));
var connection = mysql.createConnection({
host: 'localhost',
user: 'root',
password: '',
database: 'diplomdb'
});

connection.connect(function(error) {
if(error){
console.log('Error');
```

Рисунок 3.1.3 – Подключение базы данных diplomdb

После подключения базы данных можно запустить локальный сервер и приступить к заполнению базы таблицами и к реализации клиентской части дипломного проекта. Для включения всего проекта необходимо открыть командную строку и войти в директорию посредством cd, после чего прописываем консольную команду node script, и проект подключается к базе и локальный сервер может быть открыт в браузере для вывода реализации клиентской части проекта (см. рисунок 3.1.4).

```
C:\>cd mydiplom

C:\Mydiplom>node script
Listining port 1337...
Connected
```

Рисунок 3.1.4 – Общее подключение сервера Node.js

Локальный сервер, который мы используем при запуске модулей NodeJS, занимает порт, NodeJS автоматически подключает свободный порт.

3.2 Организация базы данных

После создания базы данных diplomdb необходимо создать таблицы, которые будут использоваться для вывода информации на веб-страницу и, наоборот, для принятия данных с клиентской части.

В данной главе будут рассмотрены создание и заполнение самых основных таблица базы данных diplomdb.

Таблица computerforbuy содержит все компьютеры, которые находятся в каталоге товаров, как и в модельном ряду, при ее создании было создано 4 атрибута (поля) разных типов. На этом этапе нужно было продумать, какие поля понадобятся для вывода на клиентскую часть, и поэтому было принято решение создать, немного полей, но сделать их информативными.

```
CREATE TABLE IF NOT EXISTS `computerforbuy` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `img` text NOT NULL,
  `name` varchar(25) NOT NULL,
  `cost` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=26 ;
```

Рисунок 3.2.1 – Создание таблицы computerforbuy

Следующим этапом нужно заполнить таблицу компьютерами используя СУБД phpMyAdmin sql панель (см. рисунок 3.2.2).

```
INSERT INTO `computerforbuy` (`id`, `img`, `name`, `cost`) VALUES
(1, 'static/img/6.png', 'ASER PREMIA', 260000),
(2, 'static/img/10.png', 'APEX ASPERAS', 200000),
(3, 'static/img/11.png', 'asus x2001', 600000),
(4, 'static/img/3.png', 'Asus x1001', 300000),
(5, 'static/img/7.png', 'Aser x5001', 1000000),
(6, 'static/img/8.png', 'sartoon', 140000),
(7, 'static/img/9.png', 'salihq1', 320999),
(8, 'static/img/9.png', 'releft', 100000),
(9, 'static/img/5.png', 'windalert', 770000),
(10, 'static/img/6.png', 'Windalert x2', 900000),
(11, 'static/img/3000.png', 'hyperx pc', 430000),
(12, 'static/img/3001.png', 'red line xenon', 600000),
(13, 'static/img/3002.png', 'geforce preview', 840999),
(14, 'static/img/3003.png', 'avp ', 693000),
(15, 'static/img/3004.png', 'rbg gamingpro', 1000000),
(16, 'static/img/3005.png', 'hyper dx', 340000),
(17, 'static/img/3006.png', 'pospition gaming', 500000),
(18, 'static/img/3007.png', 'starcraft gamingceries', 740000),
(19, 'static/img/3008.png', 'starcraft gamingcies x', 790000),
(20, 'static/img/3009.png', 'intel pirplegame', 950000),
(21, 'static/img/3010.png', 'supersteelseries', 340000),
(22, 'static/img/3011.png', 'acer g200', 700000),
(23, 'static/img/3013.png', 'white acer hope', 290000),
(24, 'static/img/3014.png', 'delux ', 500000),
(25, 'static/img/3012.png', 'redgraduate', 970000);
```

Рисунок 3.2.2 – Заполнение таблицы computerforbuy

Данная таблица имеет связь с другой таблицей confcomps, которая отвечает за те компьютеры, которые имеет в себе функцию конфигурации, конфигурацию мы можем изменить посредством конфигуратора. Она соединена связь один ко многим. Таблица confcomps в свое очередь соединена с 10 таблицами всех деталей компьютера как один ко многим т.е. один компьютер к 10 деталям (см. рисунок 3.2.3).

```
CREATE TABLE IF NOT EXISTS `confcomps` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `img` text NOT NULL,
  `name` varchar(25) NOT NULL,
  `cost` int(25) NOT NULL,
  `videocarta` varchar(25) NOT NULL,
  `processor` varchar(25) NOT NULL,
  `materinskayaplata` varchar(25) NOT NULL,
  `opetativnayapamat` varchar(25) NOT NULL,
  `zvookovaia` varchar(25) NOT NULL,
  `harddisk` varchar(25) NOT NULL,
  `blockpit` varchar(25) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=11 ;
```

Рисунок 3.2.3 – Создание таблицы confcomps

Следующим этапом нужно заполнить таблицу компьютерами и соответственно деталями, из которых он состоит, используя СУБД phpMyAdmin sql панель (см. рисунок 3.2.4).

```
INSERT INTO `confcomps` (`id`, `img`, `name`, `cost`, `videocarta`, `processor`, `materinskayaplata`, `opetativnayapamat`, `zvookovaia`, `harddisk`, `blockpit`) VALUES
(1, 'static/img/6.png', 'HYPERPC EARLKASE', 616000, 'GTX 759 TITAN', 'core i7 7500k', 'asusrock x100', 'tridentx', 'xenonaudio el', 'sdd samsund drive', 'kcass1-10000w'),
(2, 'static/img/11.png', 'HYPERPC 1000D', 684000, 'MSI MICRO12', 'razen ironx', 'asusrock gameseriae', 'geil', 'audiocard edition', '5220 pro micron', 'rgb1250w'),
(3, 'static/img/3.png', 'HYPERPC NOCTIS', 191900, 'asus rock xr201', 'cpu xeon gamex2', 'ASUS PROV', 'g.skill tridentz rgb', 'autoend pcx32', 'sdd samsund drive', 'kcas-750 GM'),
(4, 'static/img/7.png', 'HYPERPC CONCEPT 1', 289900, 'asus rrrr121', 'core i7kd 8000', 'gigbyte series home', 'Hyperx ddr4 8 gb', 'ngones audioacuss x64', 'seagate 256gb', 'classic dsx1 power'),
(5, 'static/img/6.png', 'HYPERPC SPEC', 191900, 'Gigabyte X212', 'core duo reborn', 'laga 1151', 'Hyperx ddr4 16 gb', 'rookaudio soundedd', 'toshiba 1000', 'atx 400w'),
(6, 'static/img/7.png', 'HYPERPC EARLKASErx', 117100, 'MSI ORANIC X121', 'core i7 ultimate game', 'asus rokfmgame', 'crusial x200', 'soundcard delux', 'wd 1.0TB', 'fatality 1000w'),
(7, 'static/img/8.png', 'RAZER EXPRESS', 200000, 'MSI MS1', 'core xeon i7 71k2', 'msi z1702', 'Hyperx ddr4/dimm 24000', 'audiocard edition', 'wd 0.5tb', 'swsa 500w'),
(8, 'static/img/9.png', 'RAZER ARHONE', 500000, 'GeForce GTX 1080', 'razen ironx', 'msi z1702 gray', 'tridentx', 'logitechaudio ellon', 'sdd wd 1000', 'kcass1-10000w'),
(9, 'static/img/10.png', 'HYPERX 2000', 400000, 'GeForce GTX 1080TI', 'razen gameedit19921', 'ASUS PROV lux', 'corsair led 2x8gb', 'xenonaudio el', 'harddisk x64', 'firestorm-750w'),
(10, 'static/img/11.png', 'HYPERX 5000', 600000, 'GeForce GTX 950', 'razen homeedit', 'ASUS PROV', 'Hyperx ddr8x4 16 gb', 'soundtruck audioedition', 'wd 500gb', 'corsair gs800');
```

Рисунок 3.2.4 – Заполнение таблицы confcomps

Одна из самых важных таблиц, которая отвечает за хранение данных зарегистрированных пользователей, называется userdata, в ней хранятся все, полня созданные в клиентской форме при заполнении которых они передаются именно в эту таблица в определенные поля.

```
CREATE TABLE IF NOT EXISTS `userdata` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `login` varchar(20) NOT NULL,
  `password` varchar(20) NOT NULL,
  `mail` varchar(50) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=7 ;
```

Рисунок 3.2.5 – Создание таблицы userdata

Как было описано ранее при входе клиента на сайт в качестве зарегистрированного пользователя, создается сессия, которая в свою очередь тоже заносится в базу данных и в режиме реального времени отображает количество активных пользователей это очень удобно для мониторинга активности на веб-странице, которая занимается продажей персональных компьютеров.

```
CREATE TABLE IF NOT EXISTS `sessions` (
  `session_id` varchar(128) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin NOT NULL,
  `expires` int(11) unsigned NOT NULL,
  `data` text CHARACTER SET utf8mb4 COLLATE utf8mb4_bin,
  PRIMARY KEY (`session_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Рисунок 3.2.6 – Создание сессии пользователя

Заполнять данную таблицу не требуется потому, как она заполняется автоматически с помощью отправки формы и соответствующих запросов, которые будут описаны в главе реализации клиентской части дипломного проекта.

Создадим все необходимые таблицы аналогично тем, которые были созданы ранее, заполним их информацией и проведем необходимые связи для корректной работы при запросе на вывод той или иной информации. Информационные блоки, созданные в таблицах конфигурационных деталей очень большие, это сделано для того, чтобы клиент мог полностью увидеть все характеристики компьютера и его деталей по отдельности.

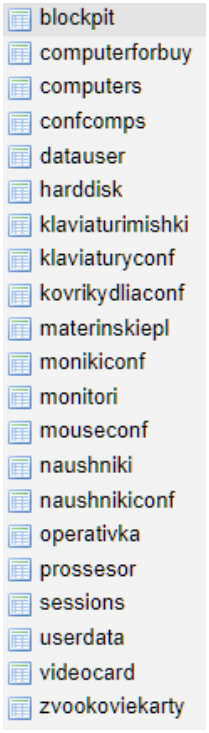

Рисунок 3.2.7 – Все таблицы базы diplomdb

3.3 Подключение необходимых файлов

Реализация клиентской части делится на этапы создания верстки веб-страницы, далее добавления динамичности посредством JavaScript, и создание запросов Ajax исходя из событий, которые происходят при том или ином действии.

Разработка клиентской части началась с верстки веб-страницы, т.е создание разметки HTML и добавления стили CSS. Прежде чем приступить к верстке, нужно было настроить препроцессоры CSS, а именно SCSS для того, чтобы можно было применять свойства объектно-ориентированного программирования при создании стилей, что позволит нам ускорить процесс. Стандартный файл формата .css загружается в программу под названием Koala, после чего файл преобразуется в scss и уже в нем пишется код на css, после чего файл конвертируется обратно в формат .css и подключается к разметке.

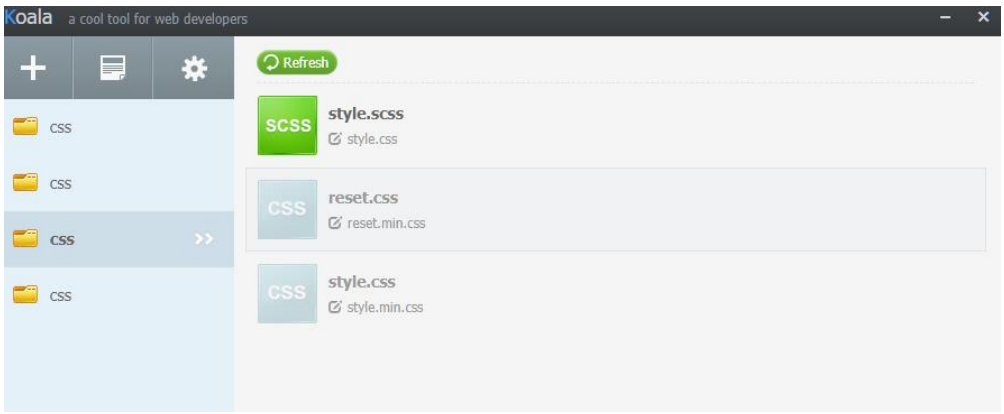

Рисунок 3.3.1 – Конвертация в препроцессорный язык стилей

Далее мы должны подключить все наши файлы стилей и JavaScript, а также библиотеки, которые мы используем jQuery, Ajax, Bootstrap. Все технологии подключаются в одну строчку. Обычно все технологии cdn и т.д. подключаются к разметке в самом начале страницы в теге head, но принято файла типа .js и библиотеки связанные с JavaScript подключать в конце разметки перед закрывающимся тегом body (см. рисунок 3.3.2).

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no
  ">
  <link rel="shortcut icon" href="static/img/iconpreload.png" type="image/x-icon" />
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/
  bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/
  iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
  <link rel="stylesheet" type="text/css" href="static/css/style.css">
  <link rel="stylesheet" type="text/css" href="static/css/reset.css">
<!--    <link rel="stylesheet" type="text/css" href="static/css/uikit.css"> -->
  <title>GEORGE_PC</title>
```

Рисунок 3.3.2 – Подключение style.css, reset.css, bootstrap.css к index.ejs

Использование cdn позволяет использовать библиотеки, не скачивая их в основную директорию нашей программы, тем самым не засорять модули не увеличивать память хостинга, если сайт используется в интернете и имеет собственный домен. Сама директория будет меньше весить и будет иметь большее быстродействие, что увеличит пользование веб-сервисом на слабых устройствах, например на мобильных устройствах или на планшетах.

Если не произвести подключение к сети интернет подключенные библиотеки или технологии через cdn не смогут подключиться, и веб-страница не загрузится, это является основным минусом подключения библиотек посредством технологии cdn.

```
<script src="https://code.jquery.com/jquery-3.3.1.js" crossorigin="anonymous"></
script>
<!--  <script src="static/js/uikit.js"></script>
<script src="static/js/uikit-icons.js"></script> -->
<script src="static/js/particles.js"></script>
<script>
  particlesJS.load('particles-js', 'static/particles.json', function() {
    console.log('callback - particles.js config loaded');
  });
  particlesJS.load('particles1-js', 'static/particles1.json', function() {
    console.log('callback - particles.js config loaded');
  });
</script>
<script src="static/js/main.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/
popper.min.js"divintro integrity="
sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin
="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
 integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
 crossorigin="anonymous"></script>
```

Рисунок 3.3.3 – Подключение main.js, ajax.js(cdn), jQuery(cdn), bootstrap.js

После подключение всех нужных технологий и библиотек можно преступить к реализации клиентской части дипломного проекта, а именно к верстке нашего сайта.

3.4 Реализация header

Сайт представляет собой листинг т.е одностраничный сайт с управляемой навигацией с одним переходом на информационный блок о конфигураторе. Создание сайта подразумевает разделение его на компоненты, что является удобным для соблюдения семантичности кода разметки. Так же стоит отметить тот факт, что использование сервера Node.js позволяет нам в режиме реального времени увидеть все изменения на странице, установив таймер на ее обновление.

Первым компонентов на веб-страницы была шапка сайта, как ее еще называют header. В ее состав входят обязательно меню навигации с одной стороны и с другой стороны иконка корзины, вход для пользователей сайта, обратная связь. Первым делом для создания header была создана разметка, представляющая из себя блок, содержащий дочерние блоки, выстроенные в одну линию, и при изменении ширины браузерного окна происходят переносы, чтобы сохранить адаптивность.

Первая часть header представляет собой меню навигации по листингу путем изменения положения высоты (см. рисунок 3.4.1).

```
<header>
    <div id="particles-js">
      <div class="menu001">
        <div class="menu002">
        <a class="likingpark" href=""><p class="logo1">GEORGE<span class="logo2">PC</span></p></
        a>
        <ul class="menunavigation">
          <li><a href="#modelraid">модельный ряд</a></li>
          <li class="left0ne"><a href="#workd1">конфигуратор</a></li>
          <li class="left1ne"><a href="#worklink3">акссесуары</a></li>
          <li class="left2ne"><a href="#worklink2">Клиентам</a></li>
          <li class="left2ne"><a href="#worklink4">О нас</a></li>
        </ul>
```

Рисунок 3.4.1 – Разметка меню навигации header

Что бы ссылки работали и перемещали на нужные части страниц им необходимо встроить ссылку, тем самым поставить якорь на тот или иной блок, на который переходит данная ссылка, что бы при нажатии на ссылку был плавный переход к якорю необходимо добавить JavaScript функцию. Для создания меню навигации и шапки header в целом серверная честь не понадобилась, так как никаких данных мы пока не обрабатываем и не передаем, тем самым можно сказать, что header реализован, используя только клиентские технологии разработки данного дипломного проекта.

```
$(document).ready(function(){
    $('.menunavigation').on("click","a", function(event){
            event.preventDefault();
            var id = $(this).attr('href'),
            top = $(id).offset().top;
            $('body,html').animate({scrollTop: top}, 1500);

    });
});
```

Рисунок 3.4.2 – Реализация плавной навигации меню header

После чего было добавлено большое количество стилей к данному элементу веб-страницы, и в итоге получилась красивая шапка header с работающими ссылками навигации по странице (см. рисунок 3.4.3).

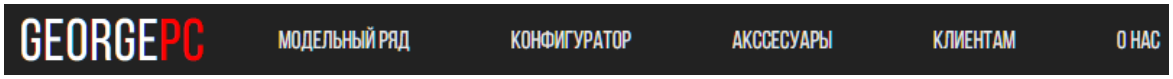

Рисунок 3.4.3 – Реализованное меню навигации header

Правая часть меню представляет собой три пункта, где первый это обратная связь, интерактивный номер телефона, вход на сайт и непосредственно иконка корзины. При нажатии на телефон появляется браузерное меню выбора возможностей для организации связи между клиентом, который использует данный веб-сервис и непосредственно организацией, номер которой указан в header (см. рисунок 3.4.4).

```
<a class="textphone" href="tel:+77019823490"><p >+7 (701) 982-34-90</p></a>
<div class="cartmenuopen">
<svg class="iconlogin1" xmlns="http://www.w3.org/2000/svg" width="23" height="23"
viewBox="0 0 24 24"><path d="M24 3l-.743 2h-1.929l-3.474
12h-13.239l-4.615-11h16.812l-.564 2h-13.24l2.937 7h10.428l3.432-12h4.195zm-15.5
15c-.828 0-1.5.672-1.5 1.5 0 .829.672 1.5 1.5 1.5s1.5-.671
1.5-1.5c0-.828-.672-1.5-1.5-1.5zm6.9-7-1.9 7c-.828 0-1.5.671-1.5 1.5s.672 1.5 1.5 1.5
1.5-.671 1.5-1.5c0-.828-.672-1.5-1.5-1.5z"/></svg>
<div id="testforadaptive" class="cartopened">
  <div class="cartonelover">
    <p class="textcartone">Моя корзина</p>
    <svg class="svgonelovercart" xmlns="http://www.w3.org/2000/svg" width="15" height="
    15" viewBox="0 0 24 24"><path d="M24 3.752l-4.423-3.752-7.771
    9.039-7.647-9.008-4.159 4.278c2.285 2.885 5.284 5.903 8.362 8.708l-8.165 9.447
    1.343 1.487c1.978-1.335 5.981-4.373 10.205-7.958 4.304 3.67 8.306 6.663 10.229
    8.006l1.449-1.278-8.254-9.724c3.287-2.973 6.584-6.354 8.831-9.245z"/></svg>
    </div>
    <div id="cartdivmainidf1" class="redlinecartone">
    </div>
    <div>
```

Рисунок 3.4.4 – Разметка корзины и входа header

На кнопку вход была создана отдельная якорная ссылка при нажатии, на которую пользователь перейдет к формам регистрации и входа в зависимости от того, что он выберет. Иконка корзины представляет собой обычную иконку и при нажатии, на которую существует событие JavaScript. Данное событие привод к реализации функции открытие корзины. Так же открытие корзины существует и при добавлении хотя бы одного товара в нее.

```
$(document).ready(function(){
    $('.svgonelovercart').click(function(){
        $('.cartopened').css('display','none')
    })
})
$(document).ready(function(){
    $('.iconlogin1').click(function(){
        $('.cartopened').css('display','block')
    })
})
```

Рисунок 3.4.5 – Функция открытия корзины при нажатии

Корзина имеет в себе возможность закрытия, и оформления выбранного товара, любой товар, выбранный на веб-странице, передает параметры цены, модели, и фотографию в корзину, где клиент может с легкостью оформить заявку на покупку выбранного товара. Был реализован удобный интерфейс использования корзины для простоты и добавлены только самые нужные функции присущие каждой корзине (см. рисунок 3.4.6).

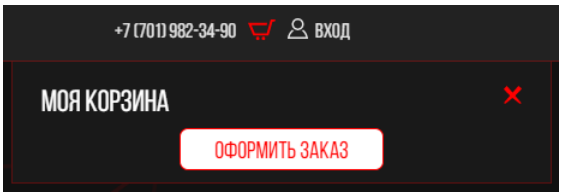

Рисунок 3.4.6 – Реализация корзины, обратной связи header

3.5 Реализация Registrate, login, logout

При нажатии на иконку входа в меню header срабатывает переход к форме регистрации или входа в зависимости от того, что выбирает пользователь, зашедший на сайт.

Разметка формы регистрации представлена в виде формы, состоящей из четырех полей ввода, каждая из которых оснащена валидацией, т.е каждое поля проверяется на правильность написание перед отправкой на сервер и в дальнейшем в базу данных для хранения.

```
<form class="form1" novalidate>
    <div  id="vhodilka2">
    <p class="text01">Регистрация</p>
      <div class="fieres">
    <input id="input1" class="form-control reydu" type="text" name="username" placeholder="
    Введите ваше имя">
    <div class="here3"><img width="35" src="static/img/onetwo.png"></div>
    <div class="here4"><img width="35"  src="static/img/cancel.png"></div>
      </div>
      <div class="fieres1">
    <input id="input2" class="form-control reydu1" type="email" name="email" placeholder="
    Введите вашу почту">
    <div class="here1"><img width="35" src="static/img/onetwo.png"></div>
    <div class="here2"><img width="35"  src="static/img/cancel.png"></div>
  </div>
    <div class="fieres3">
    <input id="input3" class="form-control reydu1" type="password" name="pass" placeholder="
    Придумайте пороль">
    <div class="here5"><img width="35" src="static/img/onetwo.png"></div>
    <div class="here6"><img width="35"  src="static/img/cancel.png"></div>
  </div>
    <div class="fieres4">
    <input id="input4" class="form-control reydu1" type="password" name="doublepass" placeholder
    ="Повторите пороль">
    <div class="here7"><img width="35" src="static/img/onetwo.png"></div>
    <div class="here8"><img width="35"  src="static/img/cancel.png"></div>
  </div>
    <button type="button" id="heandClick" class="koko1">Зарегистрировать</button>
    <button type="button" id="heandClick12" class=" koko">Вход</button>
    <span class="snapopen">Вы успешно прошли<br> регистрацию<br> <img src="static/img/onetwo.png
    "></span>

</div>

</form>
```

Рисунок 3.5.1 – Разметка формы регистрации

После создания разметки добавим стили, которых придерживаемся и подключим каждый input нашей формы к созданной валидации для проверки на правильность введенных данных (см. рисунок 3.5.2).

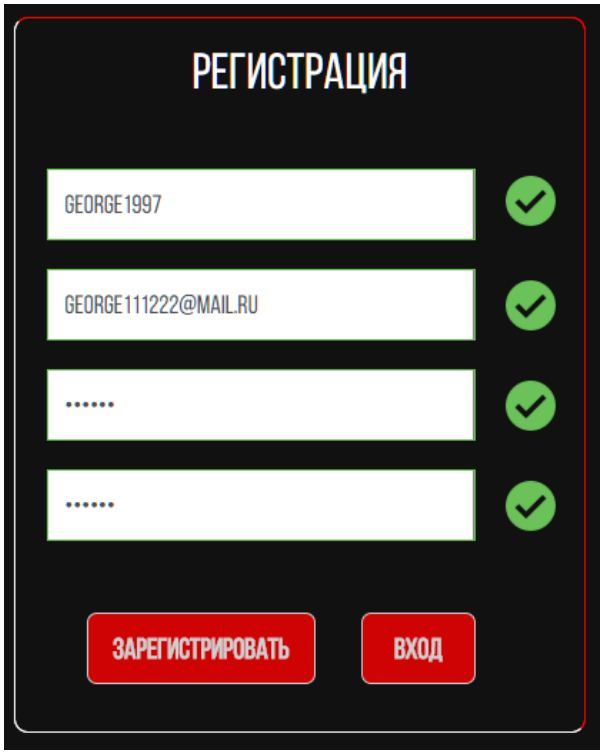

Рисунок 3.5.2 – Реализация формы регистрации

Валидация представляется собой отдельную для каждого поля проверку, которая происходит сразу, когда пользователь выходи из данного окна ввода, не при отправке, а при вводе, т.е данная валидация является очень удобной и выявляет недостатки введенных данных на стадии ввода (см. рисунок 3.5.3).

```
$(document).ready(function() {
    $('#input1').blur(function() {
        var inputnum1 = $('#input1').val();
        if (inputnum1.length < 3 || inputnum1.match(/^\d+$/)) {
            $('#input1').addClass("failclass")
            $('#input1').removeClass("successclass")
             $('.here3').removeClass("rotan")
             $('.here4').addClass("rotan")
        } else {
            $('#input1').removeClass("failclass") &&
                $('#input1').addClass("successclass")&&
                $('.here3').addClass("rotan")
                 $('.here4').removeClass("rotan")
        }
    })
```

Рисунок 3.5.3 – Валидация одного поля регистрации

Функция blur() отвечает за проверку заполненного поля input сразу после того, как заполняющий эту форму покидает input и переходит к следующему. Валидация оснащена добавлением стилей при том условии, когда валидация возвращает ошибочные значение, и оснащена успешными стилями при правильной отправке. После добавление стилей к основной форме регистрации она имеет достаточно красивый внешний вид.

Регистрация оснащена двумя кнопками первая это зарегистрировать пользователя т.е. отправить его данные на сервер, только при том условии если валидация возвращает положительный результат и выдать ошибки при том если валидация не прошла успешно, и пользователь должен ввести все данные снова. При нажатии на кнопку зарегистрировать происходит событие JavaScript, при котором происходит Ajax запрос на отправку данных, где вся наша форма с полями конвертируется и отправляется на сервер в качестве данных json.

```
function Registrate() {
    $.ajax({
        method: "POST",
        url: '/registr',
        cache: false,
        data: $('.form1').serialize(),
        statusCode: {
            418: function() {
                console.log("fail")
            }
        },
        success: function(datas) {
            console.log("Normalin")
        }
    })
```

Рисунок 3.5.4 – Отправка данных из формы регистрации на сервер /registr

Данные, которые мы отправили на сервер приходят на url адрес на серверной части и далее отправляются в базу данных, где в дальнейшем хранятся и используются для авторизации пользователя при входе.

```
app.post('/registr',function (req, res, next) {
console.log(req.body.username)
var reqObj = req.body;
var insertSql = "Insert into userdata SET ?";
var insertValues = {
"login":reqObj.username,
"password":reqObj.pass,
"mail":reqObj.email
};
var query = connection.query(insertSql, insertValues, function (err) {
if(err){
console.log('SQL error' + err);
}else {
res.json({message: "Data in database"})
}
})
})
```

Рисунок 3.5.5 – Обработка данных и заполнение таблицы userdata в БД

После чего пользователь заносится в базу данных и хранится непосредственно в ней. Для безопасности данных каждого пользователя и анонимности его сессии существует двухэтапная кодировка данных в таблице userdata.

При нажатии на кнопку вход был создан переход на аналогичную форму входа с двумя полями input для авторизации уже зарегистрированного пользователя, что очень удобно. Разметка формы входа имеет аналогичную разметку формы регистрации (см. рисунок 3.5.6).

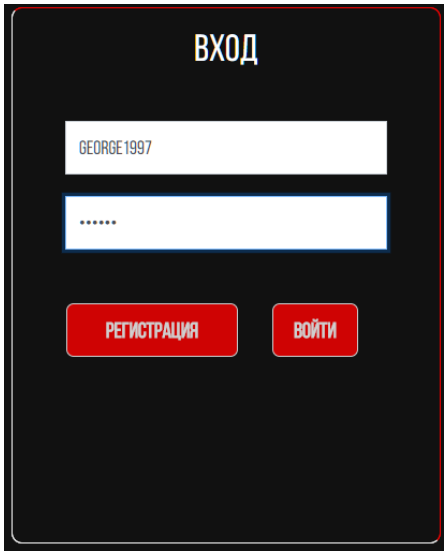

Рисунок 3.5.6 – Реализация формы входа

Так же имеет ссылку на компонент регистрации, если пользователь еще не зарегистрирован. Если пользователь успешно прошел регистрацию как это указанно выше, он может пройти авторизацию, введя свои данные в форму входа и нажав на кнопку войти.

При нажатии на кнопку войти была создана функция, в которой находится Ajax запрос на проверку введенных данных с базой, после нажатии данные отправляются на сервер и обрабатываются согласно адресу.

```
function Registrate1(){
        $.ajax({
        method: "POST",
        url: '/login',
        cache: false,
        data: $('.form2').serialize(),
        statusCode: {
            418: function() {
                console.log("fail")
            }
        },
        success: function(datas) {
            if (datas.message=="ok"){
                location.reload()
            }
        }
    })
```

Рисунок 3.5.7 – Отправка данных из формы входа на сервер /login

Далее данные проверяются с данными в базе данных, а именно в таблице userdata, после чего сервер отправляет либо код успешной авторизации, либо код ошибок, связанных с неверным указанием пароля, либо если такого логина не существует соответственно ошибку о несуществующем пользователе.

```
app.post('/login', function (req, res, next) {
console.log(req.body)
connection.query('select login, password from userdata where login = "'+req.body.username+'"'
if(!!error){
console.log('Error in the query');
} else
{
console.log(rows)
if(rows.length>0)
{
if(req.body.pass == rows[0].password)
{
res.status(200);
req.session.name = req.body.username;
res.json({message: 'ok'});
} else
{
res.status(418);
res.json({message: 'Неверный пароль'});
}
} else
{
res.status(202);
res.json({message: 'Данного пользователя не существует'});
}
}
});
});
```

Рисунок 3.5.8– Проверка введенных данных на существование и правильность

После того, как пользователь произвел успешный вход нам необходимо получить его данные, чтобы визуально показать существование сессии и создать кнопку прекращения сессии, а именно кнопу выхода пользователя. Для этого создадим функцию проверяющую, вошел ли пользователь или нет, и если он вошел, то создадим интерфейс активной сессии в шапке header.

```
function getUser(){
     $.ajax({
        method: "POST",
        url: '/user',
        cache: false,
        statusCode: {
            418: function() {
                console.log("fail")
            }
        },
        success: function(datas) {
            console.log(datas)
            if(datas.message){
            var pereme = document.getElementById('login01');
            pereme.innerHTML="";
            var button = document.createElement("button");
            var p = document.createElement("p");
            var div = document.createElement("div")
            $(div).addClass("divloginout");
            $(button).addClass("buttonlogout");
            $(p).addClass("textlogin");
            button.setAttribute("id","buttonoutlog");
```

Рисунок 3.5.9– Отправка на сервер положения пользователя /user

После чего на сервере проходит проверка авторизации пользователя, и передаются данные пользователя в созданные посредством JavaScript элементы, которым присваиваются значения имени из базы данных userdata. После чего создается активная сессия пользователя с возможностью выйти.

```
app.post('/user',function(req,res){

sess = req.session;

if(sess.name) {

res.json({message: sess.name});
} else {
res.status(202);
res.json({error: 'Вы не авторизированный пользователь'})
}
});
```

Рисунок 3.5.10 – Проверка положения пользователя и отправка данных из БД

Полученные данные вставляются в созданные в функции get User() поля и присваиваются в нашей разметке к шапке header тем самым вывод имя пользователя и кнопки выйти (см. рисунок 3.5.11).

Рисунок 3.5.11– Реализации активной сессии пользователя

Выход из сессии производится путем нажатия на созданную кнопку, выйти при которой совершается очередной запрос на сервер для прекращения сессии и обновления страницы (см. рисунок 3.5.12).

```
app.get('/logout',function(req,res){

var sid = req.session.id;
req.session.destroy(function(err) {
if(err) {
console.log(err);
} else {
console.log(sid);
res.json({message: 'ok'});
}
});
});
```

Рисунок 3.5.12– Прекращения сессии /logout

3.6 Вывод товара и оформление покупки

Как планировалось ранее, каталог товаров состоит из персональных компьютеров и дополнительной периферии. Изначально каталог представляет небольшое количество тех компьютеров, которые имею в себе функцию конфигуратора, а не только покупки, после чего был создан так называемый эффект collapse для развертывания полного списка персональных компьютеров с возможностью покупки.

Чтобы вывести компьютеры, имеющие в себе возможность изменять конфигурацию, мы должны создать запрос, который будет запрашивать данные в базе данных по адресу таблицы /computers. После успешной передачи данных, посредством JavaScript создадим переменные и присвоим им значение разметки, после чего так же с помощью JavaScript обратимся к блоку на разметке index.ejs и по идентификатору вставим созданные блоки с компьютерами в разметку. При выводе большого количества компьютеров создадим цикл, который будет создавать блоки с компьютерами.

```
$(document).ready(function(){
        $.ajax({
        method: "GET",
        url: '/computers',
        cache: false,
        success: function(datas) {
        for (i=0; i<datas.length; i++){
            var div = document.createElement("div");
            var center = document.createElement("center");
            var img = document.createElement("img");
            var p = document.createElement("p");
            var p1 = document.createElement("p");
            var div1 = document.createElement("div");
            var button = document.createElement("button");
            var button1 = document.createElement("button");
            var a=document.createElement("a");
            $(div).addClass("intodiv");
            $(center).addClass("imgdiv");
            $(p).addClass("text22");
            $(p1).addClass("text23");
            $(img).addClass("imgimgimg3");
            $(div1).addClass("groupbutt");
            $(button).addClass("bittion1");
            $(button1).addClass("bittion2");
            img.setAttribute("width","200px");
            img.setAttribute("height","220px");
            a.setAttribute("href","#workd1");
            button1.setAttribute("name",datas[i].name);
            img.src = datas[i].img;
            p.innerHTML = datas[i].name;
            p1.innerHTML ="Цена"+" "+ datas[i].cost;
            button.innerHTML = "Купить";
            button1.innerHTML = "Конфигуратор";
            a.appendChild(button1)
            div1.appendChild(button)
            div1.appendChild(a);
            center.appendChild(img);
            div.appendChild(center);
            div.appendChild(p);
            div.appendChild(p1);
            div.appendChild(div1);
            var parent = document.getElementById("divintro");
            parent.appendChild(div);
```

Рисунок 3.6.1– Запрос на таблицу и создание блоков /computers

Этот запрос отправляется на сервер по адресу /computer, где из таблицы computers передает данные img, cost, name, которые мы присваиваем переменным через JavaScript (см. рисунок 3.6.2).

```
app.get('/computers', function (req, res, next) {
connection.query('select name,cost,img from computers', function (error, rows, fields) {
if(!!error){
console.log('Error in the query');
} else
{
    res.json(rows);
}
});
});
```

Рисунок 3.6.2– Поля, которые сервер передает из таблицы computers

После того, как мы получили все нужные нам значение о компьютерах из базы и поместили их в блоки, а блоки подключили к разметке остается добавить стили для улучшения внешнего вида.

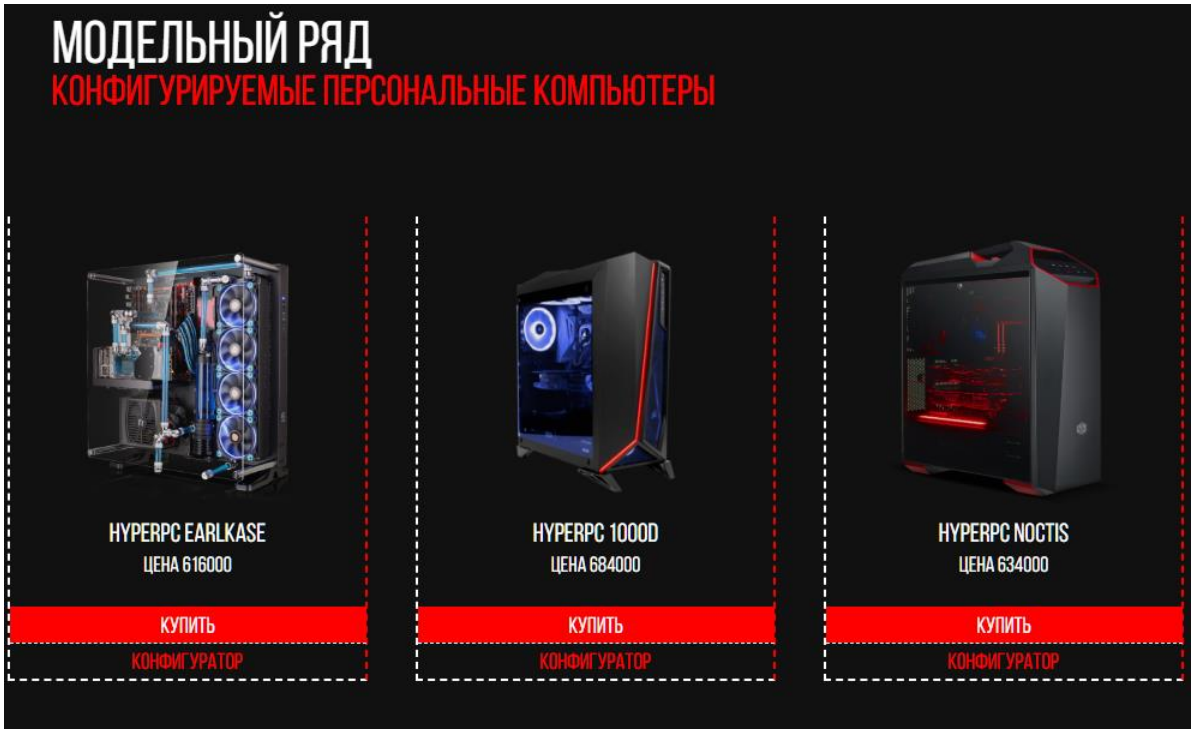

Рисунок 3.6.3 – Реализация вывода компьютеров из таблицы computers

Аналогично создадим все каталоги товаров периферии наушников, клавиатур и мышек, мониторов. Они создаются аналогично и имеют такие же функциональные возможности блока, как и у каталога компьютеров помимо конфигуратора. Т.е. все каталоги за исключением первого будут иметь только параметр покупки.

Компьютеры из данной категории имеют возможность конфигуратора и сразу покупки. При нажатии на конфигуратор пользователь будет перемещен на компонент конфигуратор, где ему представится возможность изменить конфигурацию данного компьютера, либо узнать о каждой части компьютера информацию.

При нажатии на кнопку купить, выполняется очередная функция, с помощью которой мы получаем цену компьютера его фотографию и название с помощью объекта события target, event и передаем данные именно того компьютера у которого была нажата кнопка купить. После того, как мы получили данные об этом компьютере, и нам нужно поместить их в корзину и создать цикл добавления товара в корзину столько сколько пользователь решил приобрести. Что бы использовать данные переданные при нажатии купить, нужно опять создать переменные в JavaScript и присвоить им имена из базы посредством объекта события event и присвоить их как дочерний элемент к корзине.

```
button.onclick = function(event){
    $('.cartopened').css("display","block");
    var all_cart_div = document.createElement("div");
    var img_cart_div = document.createElement("div");
    var text_cart_div1 = document.createElement("div");
    var text_cart_div2 = document.createElement("div");
    var svg_cart_div = document.createElement("div");
    var img_cart = document.createElement("img");
    var p1_cart = document.createElement("p");
    var p2_cart = document.createElement("p");
    const svg_cart = document.createElementNS("http://www.w3.org/2000/svg", "svg");
    path_cart = document.createElementNS('http://www.w3.org/2000/svg',"path");
    $(all_cart_div).addClass("cartelemntone1");
    $(p1_cart).addClass("textppp1");
    $(p2_cart).addClass("textppp2");
    $(svg_cart).addClass("svgcarttt");
    svg_cart.setAttribute("width","24");
    svg_cart.setAttribute("height","24");
    svg_cart.setAttribute("viewBox","0 0 24 24");
    img_cart.setAttribute("width","60px");
    img_cart.setAttribute("height","auto");
    path_cart.setAttribute("d","M12 0c-6.627 0-12 5.373-12 12s5.373 12 12 12 12-5.373
    p1_cart.innerHTML = $(event.target.parentNode.parentNode).find(".text22").text();
    p2_cart.innerHTML = $(event.target.parentNode.parentNode).find(".text23").text();
    img_cart.src = $(event.target.parentNode.parentNode).find("img").attr("src");
    svg_cart.appendChild(path_cart);
    img_cart_div.appendChild(img_cart);
    text_cart_div1.appendChild(p1_cart);
    text_cart_div2.appendChild(p2_cart);
    svg_cart_div.appendChild(svg_cart);
    all_cart_div.appendChild(img_cart_div);
    all_cart_div.appendChild(text_cart_div1);
    all_cart_div.appendChild(text_cart_div2);
    all_cart_div.appendChild(svg_cart_div);
     var id_cart = document.getElementById("cartdivmainidf1");
     id_cart.appendChild(all_cart_div);


   svg_cart.onclick = function(event){
        all_cart_div.remove();
        removeatrue();
   }
```

Рисунок 3.6.4– Создание экземпляра компьютера в корзине

После этого добавим стили к блокам, которые создаются после нажатия кнопки купить посредством JavaScript [8]. Так же нужно учесть тот момент, что корзина обязательно должна быть адаптивной, потому что большое количество товаров растянет ее больше высоты экрана пользователя, что создает условия, при которой оформления заказа будет невозможным, для этого создадим отдельную функцию эластичности, которая будет отвечать за расширения корзины.

```
function atrue(){
    if($('.cartopened').height()>300){
        $('.cartopened').addClass("hellowadaptive")
    }
}
```

Рисунок 3.6.5– Создание гибкости в корзине

Реализация созданных блоков при клике на кнопку купить легко проверится нажатием на нее, и самая главная цель достигнута правильная передача данных с сервера соответствует компьютеру, у которого мы и нажали кнопку купить.

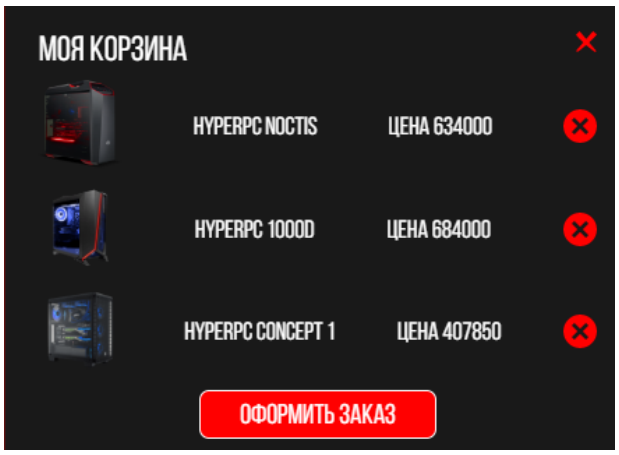

Рисунок 3.6.6 – Реализация добавления товара в корзину

При нажатии оформления заказа открывается модальное окно, которое содержит форму оформления заказа со всеми необходимыми полями и подсказками. Так же в конце формы установлена сумма всех товаров с корзины, предоставляемая к оплате. Разметка модально окна представляет собой обычную форму, которая так же подключена к валидации и имеет параметры required.

Данная форма отправления заказа является модальной и не нагружает систему сильно, потому как не загружает дополнительный компонент, а является всплывающим окном при нажатии на контекстное действие. Валидация данной формы проверяет соответствие введенных данных и регистрируемых, так как эта форма является отдельной и не зависит от регистрации, потому что мы вносим в нее данные адреса доставки или поля не является обязательным при условии, что выбрана услуга самостоятельного вывоза.

Для наглядности того, что нужно добавить в поля данной форму был добавлен атрибут placeholder [2], который отвечает за то, что написано в input до того, как покупатель внесет какие-либо данные.

```
<form class="ffirsamqwe112334111dsw">
   <p style="text-align: center;"><input id="q1" class="
   inputmodaltrye221" type="name" name="name" required  placeholder="
   Ваше имя"> </p>
   <p style="text-align: center;"><input id="q2" class="
   inputmodaltrye221" type="tel" name="tel" required pattern="
   +7[0-9]{3}-[0-9]{3}-[0-9]{2}-[0-9]{2}" placeholder="
   +7(XXX)-XXX-XX-XX"> </p>
   <p class="redlinoneaq21">после оформления заказа с вам перезвонят</p
   >
   <p style="text-align: center;"><input id="q3" class="
   inputmodaltrye221" type="email" name="email" required  placeholder="
   email"> </p>
   <p style="text-align: center;"><input  id="q4" class="
   inputmodaltrye221" type="text" name="text" required  placeholder="
   адресс"> </p>
  <select class="delevery">
       <option>доставка</option>
        <option>самовывоз</option>
  </select>
   <p class="redlinoneaq21">Оплата при доставке или при самовывозе</p>
   <div>
     <p id="obshayacffffjd"></p>
   </div>
   <p style="text-align: center;"><button class="modalbuttonnum0n1211"
   type="submit">ОФОРМИТЬ ЗАКАЗ</button></p>
</form>
```

Рисунок 3.6.7– Разметка формы оформления заказа

После добавления стилей форма принимает красивый вид согласно общему дизайну веб-страницы. Удобство использования модальных окон создается благодаря отсутствию загрузки компонента формы, тем самым загруженность страницы улучшается в разы.

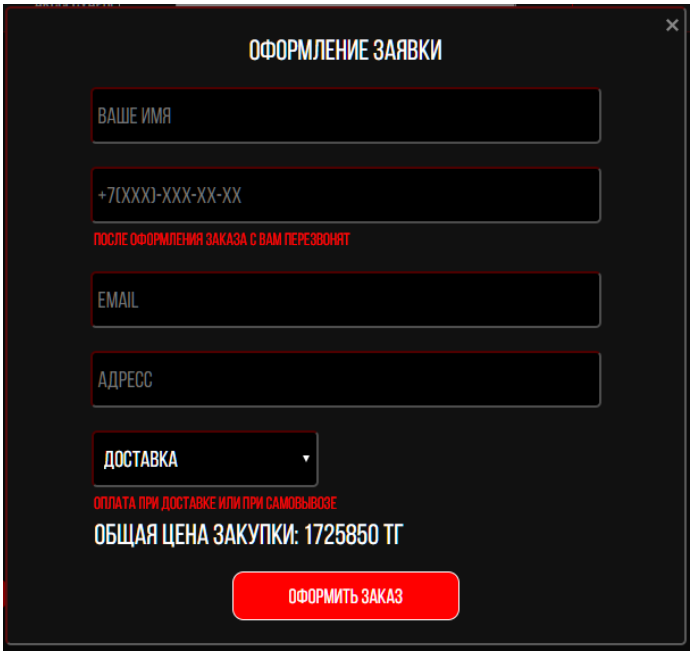

Рисунок 3.6.8 – Реализация формы оформления заказа

После нажатия на кнопку оформить заказ все данные передаются на сервер посредством функции, содержащей Ajax /sendemail02 [11].

```
$('.modalbuttonnum0n1211').click(function(){
    var result ={data:$('#cartdivmainidf1').text(),
    name_ofc:$('#q1').val(),
    tel:$('#q2').val(),
    mail:$('#q3').val(),
    address:$('#q4').val(),
    obshce1:$('#obshayacffffjd').text()};
    console.log(result);
          $.ajax({
        method: "POST",
        url: '/sendemail02',
        cache: false,
        data:result,
        statusCode: {
            418: function() {
                console.log("fail")
            }
        },
        success: function(datas) {
            console.log("Normalin")
        }
    })

})
```

Рисунок 3.6.9– Запрос на отправку всех данные из формы оформления заказа

После чего данные обрабатываются на сервере и отправляются на личную почту консультанта или менеджера, который занимается подтверждением заказа и обсуждением сборки персонального компьютера.

```
let

mailOptions = {
from: '"От Заказчика" <salihanski@gmail.com>',
to: 'georgegreek12331@gmail.com',
subject: 'Обращение с сайта ',
text: 'Это проверочное сообщение',
html: output
}
smtpTransport.sendMail(mailOptions, async function(error, info){
if (error) {
console.log(error);
} else {
console.log('Email sent: ' + info.response);
}
});
}
```

Рисунок 3.6.10 – Отправка данных с формы от сервера на указанную почту

3.7 Разработка конфигуратора

Конфигуратор является самой основной технологией на веб-сервисе. Разработанный конфигуратор является независимой единицей, но также с ним связаны компьютеры с возможностью конфигурации, т.е. только те компьютеры, которые мы добавляли первым делом в модельный ряд как конфигурируемые товары. Поэтому при нажатии конфигуратор у компьютера нам открываются все его детали и их характеристики. Но конфигуратор создавался как отдельная единица и им можно пользоваться независимо, при покупке одной детали или, же при сборке с нуля собственной конфигурации.

При создании разметки конфигуратора было принято решение создать удобный интерфейс с собственной навигацией и что бы избежать большую

высоту блока конфигуратора было принято решение использовать метод collapse и сделать категории раскрывающимися (см. рисунок 3.7.1).

```
<main class="configuration_main" >
  <p class="text_conf">Конфигуратор<br><a class="likingpark" href=""><p class="logo12">GEORGE<span class="logo2">PC</span></p></a>
  </p>
  <p class="text_alnext"><button onclick="location.href='/inform';">Узнать больше</button></p>
  <div class="menu_configurator">
    <div class="left_menu_configurator">
     <div class="accordion" id="accordionExample">
      <!-- firstlinkcollapse -->

  <a class="link_conf_main" data-toggle="collapse" href="#onelink" role="button" aria-expanded="true" aria-controls="onelink">
    <div class="flex_conf">
    <p class="text_link_conf">Комплектующие</p>
    <center class="rlinkscg" id="linkone"><div>
<svg class="svg_cong" xmlns="http://www.w3.org/2000/svg" width="25" height="25" viewBox="0 0 24 24"><path d="M0 7.3312.829-2.83
9.175 9.339 9.167-9.339 2.829 2.83-11.996 12.17z"/></svg>
 </div></center>
 <center class="rotofatos3" id="hexdsaq"><div>
<svg class="svg_cong" xmlns="http://www.w3.org/2000/svg" width="25" height="25" viewBox="0 0 24 24"><path d="M0 16.6712.829 2.83
9.175-9.339 9.167 9.339 2.829-2.83-11.996-12.17z"/></svg>
 </div></center>
    </div>
  </a>
```

Рисунок 3.7.1 – Деление разметки конфигуратора на 3 части

Каждая категория загружается в конфигуратор аналогично выводу категорий на общем сайте, но при создании конфигуратор нужно создать выбираемые элементы, а именно input type radio, что позволит нам выбирать какой-либо элемент. Каждая категория имеет свою независимую таблицу, в базе данных состоящую из большого количества атрибутов и характеристик. На примере вывода видеокарт будет продемонстрирован вывод данных с таблицы с возможностью выбора.

Для того, чтобы вывести все видеокарты их характеристики, описание, картинку, цену, необходимо создать Ajax запрос на сервер /videocard [14].

```
$(document).ready(function(){
        $.ajax({
        method: "GET",
        url: '/videocard',
        cache: false,
        success: function(datas) {
         var a = datas;
        for (i=0; i<datas.length; i++){
         div_video = document.createElement("div");
         input_video = document.createElement("input");
         label_video = document.createElement("label");
        const svg_video = document.createElementNS("http://www.w3.org/2000/svg", "svg");
        path_video = document.createElementNS('http://www.w3.org/2000/svg',"path");
         input_video.setAttribute("type","radio");
         input_video.setAttribute("name",datas[i].id);
         input_video.setAttribute("id",datas[i].name);
         label_video.setAttribute("for",datas[i].name);
         label_video.innerHTML = datas[i].name;
         svg_video.setAttribute("width","20");
         svg_video.setAttribute("height","20");
         svg_video.setAttribute("viewBox","0 0 24 24");
         path_video.setAttribute("d","M12 2c5.514 0 10 4.486 10 10s-4.486 10-10 10-10-4.486-10-10 4.486-10
         svg_video.appendChild(path_video);
         div_video.appendChild(input_video);
         div_video.appendChild(label_video);
         div_video.appendChild(svg_video);
         document.getElementById("videokartib").appendChild(div_video);
```

Рисунок 3.7.2– Запрос на данные из таблицы /videocard

На сервере данный запрос обрабатывается и подключается к таблице videocard, после чего передает все ее атрибуты, а именно картину, цену, характеристики, дополнительную информацию обратно на клиентскую часть разработки [7].

```
app.get('/videocard', function (req, res, next) {
connection.query('select name,cost,img,information from videocard', function (error, rows, fields) {
if(!!error){
console.log('Error in the query');
} else
{
    res.json(rows);
}
});
});
```

Рисунок 3.7.3 – Передача данных о видеокартах на клиентскую часть

После чего мы получаем готовый список видео карты и с помощью цикла создадим разметку в JavaScript и присвоим им данные, передаваемые с базы данных по имени атрибута в таблице [9].

При выборе, какого-либо элемента нам так же необходимо выводить его характеристики, его цену и название, а также изображение для наглядности для этого в том же цикле пропишем событие при выборе, какой-либо видеокарты.

```
input_video.onclick = function(event){

    $.ajax({
method: "POST",
url: '/videocard123',
data:{data:event.target.id},
cache: false,
success: function(datas) {
    $('#rofloturik').empty();
    var img_video = document.createElement("img");
    var div_for_video = document.createElement("div");
    var p_for_video = document.createElement("p");
    p_for_video.innerHTML=datas[0].information;
    div_for_video.appendChild(p_for_video);
    document.getElementById("rofloturik").appendChild(div_for_video);


    img_video.setAttribute("width","280px");
    img_video.setAttribute("height","250px");
    img_video.src=datas[0].img;
    var pedessa = document.createElement("p");
    $(pedessa).addClass("cenaewq");
    pedessa.innerHTML="цена:"+datas[0].cost+" "+"тг";
    pedessa.setAttribute("id","cena_video")
    document.getElementById("fotkaicena").innerHTML="";

    document.getElementById("fotkaicena").appendChild(img_video);
     document.getElementById("fotkaicena").appendChild(pedessa);
     finalcost();
}

    })


}
```

Рисунок 3.7.4– Событие при выборе любой видеокарты

После чего добавим все необходимые стили к нашим input и присвоим их к разметке посредством идентификатора, как дочерние элементы в цикле пока видеокарты не кончатся (см. рисунок 3.7.5).

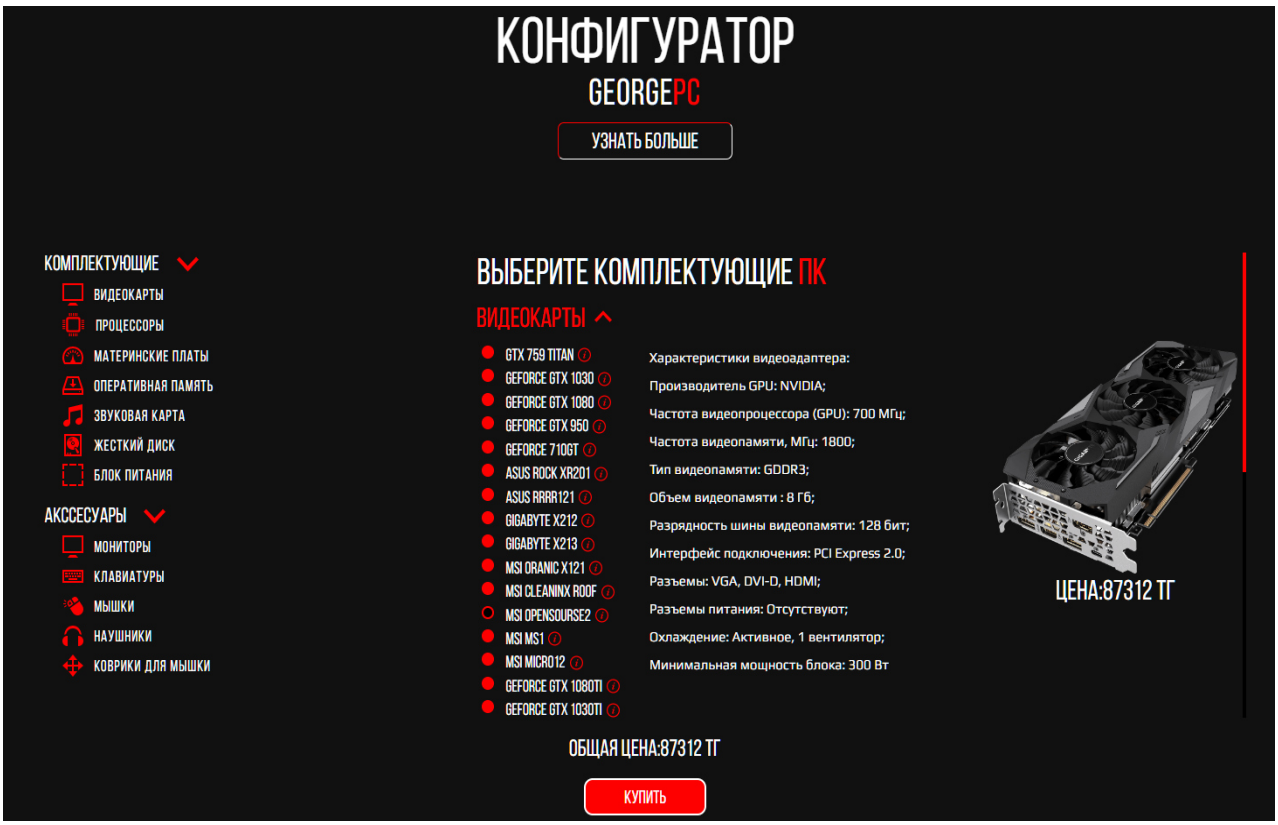

Рисунок 3.7.5 – Реализация конфигуратора

Далее в конфигураторе есть общая цена, которая подсчитывается аналогично той, что подсчитывается в оформлении заказа и была описана выше [19].

```
cenakovrydluamusiey = cenakovrydluamusiey.slice(0, -3);
cenakovrydluamusiey = parseInt(cenakovrydluamusiey);
cenanaushnikiconf = cenanaushnikiconf.slice(0, -3);
cenanaushnikiconf = parseInt(cenanaushnikiconf);
cenamouseconfigir = cenamouseconfigir.slice(0, -3);
cenamouseconfigir = parseInt(cenamouseconfigir);
cenamonikiconfigy = cenamonikiconfigy.slice(0, -3);
cenamonikiconfigy = parseInt(cenamonikiconfigy);
cenaklavaconfigyy = cenaklavaconfigyy.slice(0, -3);
cenaklavaconfigyy = parseInt(cenaklavaconfigyy);
cenavideokarty = cenavideokarty.slice(0, -3);
cenavideokarty = parseInt(cenavideokarty);
cenaproccesory = cenaproccesory.slice(0, -3);
cenaproccesory = parseInt(cenaproccesory);
cenamaterinkyy = cenamaterinkyy.slice(0, -3);
cenamaterinkyy = parseInt(cenamaterinkyy);
cenaoperativka = cenaoperativka.slice(0, -3);
cenaoperativka = parseInt(cenaoperativka);
cenazvookyyyyy = cenazvookyyyyy.slice(0, -3);
cenazvookyyyyy = parseInt(cenazvookyyyyy);
cenahardyyyyyy = cenahardyyyyyy.slice(0, -3);
cenahardyyyyyy = parseInt(cenahardyyyyyy);
cenablockyyyyy = cenablockyyyyy.slice(0, -3);
cenablockyyyyy = parseInt(cenablockyyyyy);
```

Рисунок 3.7.6 – Выбор конфигурации готового компьютера

Аналогично добавим одиннадцать деталей, состоящих более чем из двадцати элементов в каждом и аналогично выведем их в конфигуратор и сделаем событие на выбор, какого-либо input как показано выше [8]. Т.е. если, когда мы нажимаем на конфигуратор, у какого, либо компьютера система смотрит, какие детали есть у этого компьютера, и автоматически выбирает их в конфигураторе и выдает информацию [10].

```
        a.onclick = function(event){
    var id = $(this).attr('href'),
    top = $(id).offset().top;
    $('body,html').animate({scrollTop: top}, 1500);
    $.ajax({
method: "POST",
url: '/proverka123',
data:{data:event.target.name},
cache: false,
success: function(datas) {


 document.getElementById(datas[0].videocarta).click();
 document.getElementById(datas[0].processor).click();
 document.getElementById(datas[0].materinskayaplata).click();
 document.getElementById(datas[0].opetativnayapamat).click();
 document.getElementById(datas[0].zvookovaia).click();
 document.getElementById(datas[0].harddisk).click();
 document.getElementById(datas[0].blockpit).click();
 document.getElementById("roomkavoki").click();
 document.getElementById("roomkavoki1").click();
 document.getElementById("roomkavoki2").click();
 document.getElementById("roomkavoki3").click();
 document.getElementById("roomkavoki8").click();
 document.getElementById("roomkavoki9").click();
 document.getElementById("roomkavoki10").click();
 document.getElementById("linkoneroofler").click();
 document.getElementById("linkoneroofler1").click();
 document.getElementById("linkoneroofler2").click();
 document.getElementById("linkoneroofler3").click();
 document.getElementById("linkoneroofler8").click();
 document.getElementById("linkoneroofler9").click();
 document.getElementById("linkoneroofler10").click();
}

    })
```

Рисунок 3.7.7 – Выбор конфигурации готового компьютера

После чего товар аналогично всем покупка создает блок в JavaScript и помещает его в корзину и происходит оформления заказа аналогично тому, как оформлялся заказ выше с дальнейшей передачей формы на личную почту консультанта или менеджера, который занимается подтверждением заказа и обговаривает все нужные детали до начала сборки персонального компьютера.

Так же следует отметить, что конфигуратор содержит в себе ссылку перехода на информационный блок, в котором написано об операционной

системе, предоставляемой веб-сервисом, а также о том, как производится сборка компьютера.

3.8 Реализация footer и обратная связь

Обратная связь представляет собой однострочную форму, которая вызывается после нажатия заказать звонок в footer. Footer организован под типу header, и включает в себя ссылки на функциональные переходы веб-страницы.

Рисунок 3.8.1 – Реализация footer

Обратная связь открывает модальное окно при нажатии заказать звонок, отправляет данные аналогично оформлению заказа на почту (см. рисунок 3.8.2).

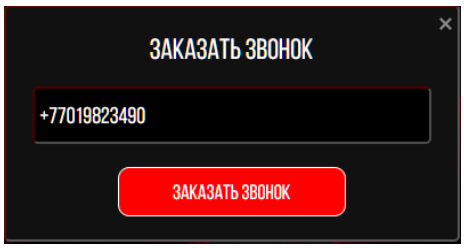

Рисунок 3.8.2– Реализация обратной связи

После отправки форма конвертируется в формат json и выполняет Ajax запрос, который отправляет наш телефон на сервер, где он в дальнейшем отправляется на почту консультанту (см. рисунок 3.8.3).

Рисунок 3.8.3– Отправка на сервер значения формы

В итоге можно сделать несколько выводов, было использовано большое количество современных технологий, конструкций и библиотек. Что бы приложение стало популярным и им пользовалось большое количество людей необходимо использовать только современный стек технологий как для серверной части, так и для клиентской части разработки веб-приложения. Как выяснилось, база данных является основным элементом работы с большим количеством данных, как например, с огромным количеством делателей в данном проекте.

Главная цель, а именно реализация веб-приложения, который сможет стать функционально особенным и содержать в себе огромную базу данных, которой можно манипулировать посредством конфигуратора, а также сам конфигуратор, нейтрализует поломки и несовместимость, и предлагает пользователю самому собрать компьютер своей мечты или изменить имающийся. Конфигуратор предоставляет на выбор любые современные детали по разным категориям с детальными описаниями характеристик.

Можно сделать вывод, что данное веб-приложении является актуальным, потому как имеет умную систему совместимости, имеет удобный интерфейс, конфигуратор с большим количеством данных и информации.

4 Технико-экономическое обоснование

Цель данного дипломного проекта заключается в разработке WEB-приложения, с помощью которого можно очень быстро в режиме онлайн собрать свой компьютер и увидеть его мощность, цену, срок годности и непосредственно все выбранные вами компоненты. Так же заказать выбранный нами продукт, данный проект носит название конфигуратор технологий и поэтому он является универсальным и может быть перенастроен на сборку любой другой технологии.

Для разработки данного WEB-приложения была создана группа специалистов: WEB-дизайнер, Frontend developer, Backend developer, директор проекта. В данном проекте первым делом нужно было создать техническое задание, построить график работы, план выполнения данного проекта. Далее за работу берется WEB-дизайнер, который берет техническое задание и делает по нему свой индивидуальный дизайн, после одобрения основной идеи, дизайнер реализует свою идею.Frontend developer получает готовую папку с дизайном с использованными шрифтами и макет формата .psd, и начинает реализовывать данный макет в WEB-приложение с использованием современных технологий WEB-разработки, т.е. он делает полностью всю клиентскую часть данного проекта и делает запросы и обрабатывает их, которые в дальнейшем будет передавать, и принимать с базы данных Backend developer.

Backend developer в его обязанности входит вся серверная часть проекта создания базы данных и передача туда данных по запросам и наоборот вызов информации из базы данных и на нем лежит ответственность за хранения данных за их безопасность, поэтому он тестирует запросы и сопровождает их. Технико-экономическое обоснование содержит следующие пункты:

- определение сложности разработки WEB-приложение;
- расчет затрат на разработку WEB-приложение;
- определение ценности готового продукта;
- оценка результатов работы WEB-приложение.

4.1 Определение сложности разработки WEB-приложение

Для определения того насколько проект будет сложным и трудоемким нужно разделить нашу разработку на этапы и под этапы, чтобы было легче понять на что уйдет больше времени и сил. Определение трудоемкости является основным этапом в экономической части нашего проекта, для наглядности существует специальная форма разделения работ по этапам, определение трудоемкости показывает насколько сложным будет проект и на какие части его нужно разделить что бы получить желаемый результат при расчетах т.е. экономическая часть любого проекта зачастую определяет его прибыльности и цену на продажу были получено 17 этапов, которые показывает как нужно разделить выполнение данного проекта, для

использования минимально затраченного времени этапов, которые приведены ниже в таблице 4.1.

Таблица 4.1.1 - Распределение работ по этапам и оценка их трудоемкости

Этапы разработки ПО	Вид работы	Трудоемкость, чел. час.
Этап 1	Создание технического задания	15
Этап 2	Организовать график работы	5
Этап 3	Выбор средств разработки, среда разработки и технологии разработки	9
Этап 4	Изучение требуемых программ для реализации проекта	25
Этап 5	Планировка выполнение этапов реализации	5
Этап 6	Написание и оформление теоретической части	10
Этап 7	Практическая часть	25
Этап 8	Реализация проекта	60
Этап 9	Устранение неполадок	15
Этап 10	Создание дизайна	10
Этап 11	Реализация клиентской части WEB-приложения	15
Этап 12	Проверка работы и отладка клиентской части	15
Этап 13	Реализация серверной части WEB-приложения	10
Этап 14	Проверка работы и отладка	10
Этап 15	Проверка на адаптивность и кроссбраузерность	10
Этап 16	Создание полноценного отчета	20
Этап 17	Выводы по проделанной работе	5
Итого: трудоемкость выполнения		264

Продолжительность рабочего дня равна 8 часам. В результате для реализации программного обеспечения необходимо 33 рабочих дней получено по расчетам.

$$\frac{264}{8} = 33, \text{ количество рабочих дней.}$$

4.2 Расчет затрат на разработку WEB-приложение

Определение затрат необходимых для разработки программного обеспечения производится на основе имеющейся сметы, которая включает следующие элементы:

– материальные затраты;
– затраты на оплату труда;
– социальный налог;
– амортизация основных фондов;
– прочие затраты.

Материальные затраты делятся на основные и вспомогательные затраты на материалы, энергию и другие затраты необходимые для разработки WEB-приложение. Расчет материальных затрат происходит по форме, предоставленной в таблице 4.2.

Таблица 4.2.1 - Затраты на материальные ресурсы

Наименование материала	Марка	Ед. измерения	Количество	Цена за ед. в тенге	Сумма в тенге
Упаковки бумаги для распечатки	Sveto-Copy	Упаковка	1	750	750
Карандаши для ТЗ	Koh-i-Noor	Штук	2	190	380
Ежедневник	Cross	Штук	1	1400	1400
Ручки	Delta	Штук	2	300	600
Клавиатура для ПК	Delux	Штук	1	4000	4000
Картридж для принтера	Canon	Штук	1	2900	2900
Итого:					10030

При разработке данного WEB-приложения я использовал ПК Acer Aspire TC-605, который полностью оснащен всеми необходимыми программными обеспечениями и приобрел полностью поддержанную операционную систему x64 и необходимым пакетом драйверов. Поэтому затрат на системные требования не было.

Общую сумму, необходимую на материальные средства ($З_м$) можно рассчитать по следующей формуле:

$$З_м = \sum P_i * Ц_i, \quad (4.1)$$

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.

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

Таблица 4.2.2 - Затраты на оборудование

Наименование материала	Марка	Ед. измерения	Количество	Цена за ед. в тенге	Сумма в тенге
Персональный компьютер	Acer Aspire TC-605	Штук	1	600 000	600 000
Роутер	TP-link 601	Штук	1	50 000	50 000
Хостинг	HOSTER.KZ	Штук	1	6000	6000
Домен	HOSTER.KZ	Штук	1	2500	2500
Операционная система	Windows 10	Штук	1	20 000	20 000
Принтер	Canon image FORMULA	Штук	1	60 000	60 000
Итого:					738 500

Зм = 10030+ 738 500= 748 530 (тг)

В итоге получается, что для создания моего дипломного проекта, а именно WEB-приложения мне необходимо748 530(тг) на материальные затраты и оборудования.

4.3 Расчет затрат на электроэнергию

При разработке данного проекта я использовал электроприборы, которые потребляли электроэнергию соответственно нужно просчитать затраты на электроэнергию; для этого учтем электроприборы, которые использовались при разработке проекта. Так же стоит отметить, что все приборы, которые были использованы, работают по определенному трафику для физических лиц, трафик установлен согласно законодательству о эксплуатации электро - энергии Республики Казахстан, и является общепринятым числом. Цена за единицу использования электроэнергии составляет поставленное гостом число в размере 18,32.

От того насколько много было использовано приборов, потребляющих электроэнергию, зависит многое и влияет на конечную себестоимость полученного в результате продукта, потому как необходимо платить за потребляемую энергию по общей ставке, а она зависит от того какое

количество компьютеров необходимо для реализации и самое главное время работы каждого электроприбора.

Таблица 4.3.1 - Затраты на электроэнергию

Наименование приборов	Паспортная мощность, кВт	Коэффициент мощности	Время работы оборудования, ч	Цена ЭЭ тг/кВтч	Сумма, тг.
Персональный компьютер	0,80	0,70	264	18,32	2 708,40
Роутер	0,05	0,90	264	18,32	217,60
Принтер	0,20	0,50	200	18,32	366,40
Освещение	0,30	0,70	264	18,32	1 015,70
Итого:					4 308,10

$$З_{эл.эн.обор.} = 4\ 308{,}10 (тенге)$$

На дополнительные потребности расходы подсчитываются на основе повышенного показателя в объеме 5% от расходов на электроэнергию:

$$З_{доп.нужды} = 5\% * З_{эл.эн.обор.}, \quad (4.4)$$

Определим затраты на дополнительные потребности согласно формуле (4.4):

$$З_{доп.нужды} = 0.05 * 4\ 308{,}10 = 215{,}40\ (тенге)$$

Исходя из всех расчетов, полные расходы на электроэнергию составляют:

$$Э = 215{,}40 + 4\ 308{,}10 = 4523{,}50\ (тенге)$$

4.4 Расчет затрат на оплату труда

Для создания моего WEB-приложения понадобилась работа следующих специалистов:

– директор проекта, который занимается планировкой времени;
– веб-дизайнер реализация технического задания и создания макета;
– frontend developer создания макета сайта из шаблона дизайнера;
– backend developer занимается серверной частью проекта.

Сумму расходов на оплату труда можно рассчитать по следующей формуле:

$$З_{тр} = \sum ЧС_i * T_i \quad (4.5)$$

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

T_i - трудоемкость разработки модели, чел.×ч; i - категория работника;

n - количество работников, занятых разработкой ПП.

Во время реализации проекта рабочее время участников не равномерно, поэтому имеет смысл установить часовую ставку каждого работника и общий объем заработной платы.

Часовую ставку сотрудника можно рассчитать по следующей формуле:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} \quad (4.6)$$

где $ЗП_i$ - месячная заработная плата i-го работника, тг;

$ФРВ_i$- месячный фонд рабочего времени i-го работника, час.

Месячная заработная плата руководителя равняется 200 000 тенге, WEB-дизайнер получает заработную плату 120 000 тенге, Frontendполучает 200 000 тенге, Backend developer имеет заработную плату 220 00 тенге. Рассчитаем часовую ставку каждого работника согласно формуле (4.6):

$$ЧС_{директор\ проекта} = \frac{200\,000}{33 * 8} = 757{,}57\,\frac{тг}{ч}$$

$$ЧС_{WEB-дизайнера} = \frac{120\,000}{33 * 8} = 454{,}54\,\frac{тг}{ч}$$

$$ЧС_{Frontend} = \frac{200\,000}{33 * 8} = 757{,}57\,\frac{тг}{ч}$$

$$ЧС_{Backend} = \frac{220\,000}{33 * 8} = 833{,}33\,\frac{тг}{ч}$$

Поданным XAMPP Control panel затраты на все этапы разработки т.е на все этапы деятельности показала следующую трудоемкость:

Ставка директора проекта равна 757,57, трудоемкость его деятельности равняется 130 часам;

Часовая ставка WEB-дизайнер составляет 454,54 (тг/ч), трудоемкость дизайнера равняется 50 часам;

Часовая ставка Frontend developer составляет 757,57, трудоемкость разработки равняется 100 часов;

Часовая ставка Backend developer составляет 833,33, трудоемкость разработки равняется 60 часов**.**

$$З_{тр} = 757{,}57 * 130 + 454{,}54 * 50 + 757{,}57 * 100 + 833{,}33 * 60$$
$$= 98\,484 + 22\,727 + 75757 + 49\,999{,}90 = 246\,967{,}90$$

Расчеты затрат по оплате труда показаны в таблице (4.5).

Таблица 4.4.1 - Затраты на оплату труда

Категория работника	Квалификация	Трудоемкость разработки ПП, час.	Часовая ставка, тг/ч	Сумма, тг.
Директор проекта	Проектный руководитель	130	757,57	98 484
WEB-дизайнер	Дизайнер	50	454,54	22 727
Frontend developer	Программист	100	757,57	75757
Backend developer	Программист	60	833,33	49 999,90
Итого:				246 967,90

4.5 Расчет затрат по социальному налогу

Согласно Налоговому кодексу Республики Казахстан социальный налог составляет 9,5% от фонта оплаты труда;

$$С_н = (ФОТ - ПО) * 0{,}095 \quad (4.7)$$

где ПО - отчисления в пенсионный фонд, они составляют 10% от ФОТ.

$$ПО = 246\,967{,}9 * 0{,}1 = 24\,696{,}79 \text{ тенге}$$
$$С_н = (246\,967{,}9 - 24\,696{,}79) * 0{,}095 = 21\,115{,}90 \text{ тенге}$$

4.6 Амортизация основных фондов и прочие затраты

Нормы амортизации ОФ:

$$A_г = \frac{С_{об} * Н_а}{100} \quad (4.8)$$

$С_{об}$ – стоимость оборудования;
$Н_а$ – норма амортизации (норма амортизация = 25).

Формула (4.8) суммы для амортизационных отчислений за год для ПК:

$$A_{\text{г}} = \frac{600\,000 * 25}{100} = 150\,000{,}00 \text{ тенге}$$

Теперь необходимо рассчитать норму амортизации за период разработки:

$$A_{\text{г}} = \frac{150\,000 * 33}{365} = 13\,598{,}90 \text{ тенге}$$

Подобным образом необходимо рассчитать норму амортизации для всего оборудования. Результаты расчетов приведены в таблице (4.7).

Таблица 4.6.1 - Амортизация основных фондов (ОФ)

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Сумма амортизации за год, тг	Сумма амортизации за время разработки, тг
Персональный компьютер	600 000	25	150 000	13 598,90
Принтер	60 000	25	15 000	82,19
Роутер	50 000	20	10 000	904,10
Хостинг	6000	20	1200	65,75
Домен	2500	15	375	20,54
Операционная система Windows	20000	20	4000	361,64
Итого:			180 575	15 033,12

На основе всех представленных расчетов необходимо оформить смету расходов, которая приведена в таблице (4.8).

Таблица 4.6.2- Смета затрат на разработку

Статьи затрат	Сумма, тг
Затраты на оборудование	738 500,00
Затраты на материальные ресурсы	10030,00
Затраты на оплату труда	246 967,90
Социальные налоги	21 115,90
Затраты на электроэнергию	4523,50
Амортизация основных фондов	15 033,12

Итого по смете:	1 036 170,42

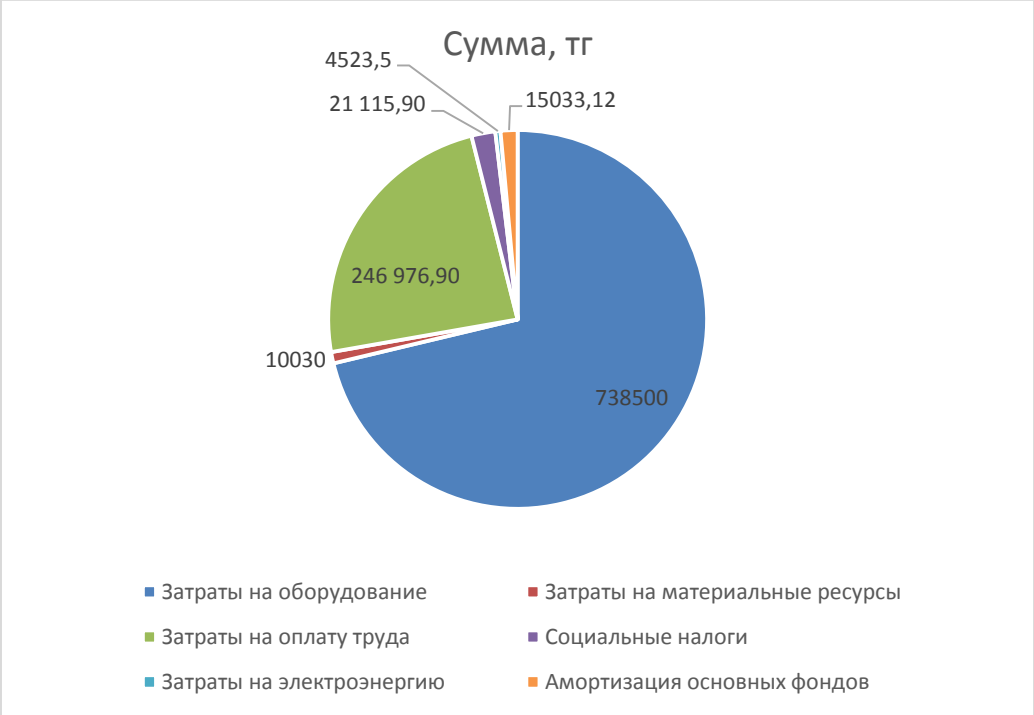

Рисунок 4.6 – Смета затрат на разработку проекта.

4.7 Определение возможной (договорной) цены ПО

Стоимость WEB-приложения определяется на основе качества разработанного продукта, сроков его разработки и производительности продукта. Стоимость $Ц_Д$ для WEB-приложения можно рассчитать по следующей формуле:

$$Ц_Д = З_{нир} * \left(1 + \frac{P}{100}\right), \quad (4.9)$$

$З_{нир}$ – затраты на разработку программного обеспечение, тг;

Р – Средний уровень рентабельности ПО, (%). Данный параметр принят равным 25%.

$$p = 1036\ 170{,}42 * \left(\frac{25}{100}\right) = 259\ 042{,}60 \text{тенге},$$

Где p – прибыль.

$$Ц_Д = З_{нир} + p = 1\ 036\ 170{,}42 + 259\ 042{,}60 = 1\ 295\ 213{,}02$$

Далее необходимо определить стоимость реализации с учетом НДС, ставка НДС устанавливается законодательством РК. На 2019 года ставка НДС

составляет 12%. Стоимость реализации, учитывая НДС можно рассчитать по следующей формуле:

$$Ц_р = Ц_д + Ц_д * НДС, \quad (4.10)$$

$$Ц_р = 1\,295\,213{,}02 + 1\,295\,213{,}02 * 0{,}12 = 1\,450\,638{,}58 \text{ тенге}$$

Данную цену можно округлить до 1 500 000,00 тенге.

4.8 Вывод по технико-экономической части

Таким образом, цена реализации с учетом НДС равна 1 450 638,58 тг за продукт (1 500 000), себестоимость равна 1 036 170,42 тг и прибыль равна 259 042,60 тг.

5 Охрана труда и безопасность жизнедеятельности

Темой дипломного проекта было выбрано WEB-приложение, состоящее из двух частей, первая часть этап разработки технического задания, и второй этап программная разработка. При разработке были соблюдены все правила техники безопасности, так как работа является сидячей, а именно разработка сайта то основными пунктами были: освещение, вентиляция, а также работа с электрическими приборами и правила их эксплуатации.

В данной части дипломного проекты была рассмотрена тема вентиляции помещения. Вентиляция — это основной параметр безопасности жизнедеятельности во время труда и является обязательным параметром для расчета.

5.1 Расчет аспирационной системы для производственного помещения

Таблица 5.1 – Исходные данные

Город	Алматы
Параметры помещения (Д*Ш*В), м	12 * 8 * 4
Данные по оборудованию	
Количество компьютеров, шт.	6
Мощность $P_{об}$, кВт/ч	0.95
КПД η	0.7
Данные по источнику света	
Мощность $N_{осв}$, Вт/м^2	60
Вид источника освещения	люминесцентные лампы
Количество сотрудников	2 (мужчины), 2 (женщины)
Окна	
Количество, шт.	3
Площадь 1 окна, м^2	0.5
Расположение	СВ
Вид жалюзи	вертикальные металлические
Расчетное время суток, ч.	12-13
Температура в помещении	
Летом, ^{o}C	23
Зимой, ^{o}C	21
Вид положения работы	сидячая

5.1.1 Расчет тепловых нагрузок в помещении

В помещениях различного назначения действуют в основном тепловые нагрузки, возникающие снаружи помещения (наружные); тепловые нагрузки, возникающие внутри здания (внутренние).

5.1.2 Расчет тепловых нагрузок снаружи помещения

Наружные тепловые нагрузки показывают параметры 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 является отрицательной и направления потока тепла меняется;

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

– теплопоступления от инфильтрации.

Следует отметить, что наружные тепловые нагрузки могут обладать различными свойствами, т.е. могут быть положительными в зависимости от времени года и времени суток.

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):

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

$V_{пом}$ – объем помещения (6.2), м^3

$$V_{\text{пом}} = 12 * 8 * 4 = 384\ м^3; \quad (5.2)$$

$X_о$ – удельная тепловая характеристика, Вт/м^{3о}С, $X_о = 0.42$;

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

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

Для теплого времени года (5.3):

$t_{Нрасч}$= 29,4 ^{o}С

$t_{Врасч}$ = 26 ^{o}С

$$Q_{огр} = 384 * 0.42 * 3{,}4 = 548{,}4 \approx 548{,}4\ Вт \quad (5.3)$$

Для холодного времени года (5.4):

$t_{Нрасч}$ = -9 ^{o}С

$t_{Врасч}$ = 19 ^{o}С

$$Q_{огр} = 384 * 0.42 * |-28| = 4515{,}84 \approx 4516 \ Вт \qquad (5.4)$$

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 (5.5):

$$Q_P = (q^I F_O^I + q^{II} F_O^{II}) * \beta_{с.з} \qquad (5.5)$$

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;

$\beta_{с.з}$- коэффициент тепло-пропускания. $\beta_{с.з} = 0.15$.

При отсутствие наружных затеняющих в козырьков, ребер и т.д. для периода облучения остекления солнцем, когда его лучи проникают через окно в помещение $F_O^I = F_O \ ; \ F_O^{II} = 0$, (5.6):

$$Q_P = q^I F_O * \beta_{с.з} = \left(q_{вп} + q_{вр}\right) * K_1^C * K_2 * n * S_O \qquad (5.6)$$

$q_{вп}$, $q_{вр}$- тепловые потоки от прямой рассеянной радиации, Вт/м^2. Для широты в 44^{o}СШ до полудня в 11-12 ч. При расположении С: $q_{вп}$, $q_{вр}$

$q_{вп} = 73$ Вт/м^2

$q_{вр} = 77$ Вт/м^2

F_O – площадь светового проема (5.7) (n – число окон, S_O – площадь одного окна);

$$F_O = n * S_o = 3 * 0.5 = 1.5 \ м^2 \qquad (5.7)$$

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 (6.8):

$$Q_P = (73 + 77) * 0.72 * 0.9 * 0.9 * 1.5 * 0.15 = 21{,}9 \ Вт \qquad (5.8)$$

Поступление солнечного излучения равно (5.9):

$$Q_P = 21{,}9\ Вт \tag{5.9}$$

5.1.3 Расчет тепловых нагрузок внутри помещения

Внутренние тепловые нагрузки в жилых, офисных или относящийся к сфере обслуживание помещения слагаются в основном из тепла:

– выделяемого людьми;

– выделяемого лампами и осветительными, электробытовыми приборами;

– выделяемого компьютерами, печатающими устройствами и фотокопировальными машинами и пр. (в офисных и других помещениях).

В производственных и технологических помещениях различного назначения дополнительными источниками тепловыделение могут быть: нагретое производственное оборудование; горячие материалы, в том числе жидкости 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 путём конвекции и лучеиспусканий, и скрытого тепла, затрачиваемого на испарение влаги с поверхности кожи и из легких.

Летом при $24\ ^oC$ один мужчина выделяет явного тепла 61 Вт, а общего – 102 Вт (при работе стоя, легком движении). Выделение явного тепла составит (5.10):

$$Q_л^я = 61 * 2 + 61 * 2 * 0.85 = 225{,}7\ \ Вт \tag{5.10}$$

Выделение общего тепла (5.11):

$$Q_л^о = 102 * 2 + 102 * 2 * 0.85 = 377{,}4\ \ Вт \tag{5.11}$$

Зимой при $20\ ^oC$ один мужчина выделяет явного тепла 82 Вт, а общего – 103 Вт. Тогда выделение явного тепла в помещении составит (5.12):

$$Q_л^я = 82 * 2 + 82 * 2 * 0.85 = 303{,}4\ \ Вт \tag{5.12}$$

Выделение общего тепла (5.13):

$$Q_л^о = 103 * 2 + 103 * 2 * 0{,}85 = 381{,}1\ Вт \tag{5.13}$$

Теплопоступление от осветительных приборов, оргтехники и оборудования рассчитывается следующим образом. Теплопоступление от ламп определяется по формуле (5.15):

$$Q_{\text{осв}} = \eta * N_{\text{осв}} * F_{\text{пол}} \quad (5.15)$$

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

$N_{осв}$ – установленная мощность ламп ($N_{осв} = 60\ Вт/м^2$);

$F_{пол}$ – площадь пола ($F_{пол} = 12 \cdot 8 = 96\ м^2$);

Тогда (5.16):

$$Q_{осв} = 0{,}5 \cdot 60 \cdot 96 = 2880\ Вт \quad (5.16)$$

Тепло, выделяемое производственным оборудованием, определяется по формуле (5.17):

$$Q_{об} = N_{уст} * K \quad (5.17)$$

где $N_{уст}$ – паспортная мощность оборудования ($N_{уст} = 1.1\ кВт/ч$);

K - единиц оборудования ($K = 3$);

Тогда (5.18):

$$Q_{об} = 1{,}8 * 10^3 * 6 * 0{,}95 = 10260\ Вт \quad (5.18)$$

Теплопритоки, возникающие за счет находящейся оргтехники – это 30% мощности оборудования (5.19):

$$Q_{об} = 1{,}8 * 103 * 6 * 0{,}3 = 3240\ Вт \quad (5.19)$$

5.2 Расчет теплового баланса помещения

На основании выполненных расчетов поставим баланс теплопоступлений в помещении:

Летом (5.20):

$$Q_{изб} = 64{,}6 + 225{,}7 + 2880 + 10260 + 3240 + 548{,}4 = 17218{,}7\ Дж \quad (5.20)$$

Зимой (5.21):

$$Q_{изб} = 64{,}6 + 303{,}4 + 2880 + 10260 + 3240 + 4515{,}84 = 21263{,}84\ Дж \quad (5.21)$$

Т.к. тепловой баланс для зимы больше летнего теплового баланса, то рассчитаем тепло-напряжённость воздуха по формуле (5.22, 5.23):

$$Q_H = \frac{Q_{изб}*860}{V_{пом}} \quad (5.23)$$

$$Q_H = \frac{21263.84*860}{384*10^3} = 47{,}62\ ккал/м^3 \quad (5.23)$$

При $Q_H > 20\ ккал/м^3, \Delta t = 8\ ^oC$

Определение количества воздуха, необходимое для поступления в помещение (5.24) и расчет (5.25):

$$L = \frac{Q_{изб} * 860}{C * \Delta t * \gamma} \quad (5.24)$$

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

$$L = \frac{21263.84*860}{0.24*10^3*8*1.206} = 7897{,}53\ \frac{м^3}{час} \quad (5.25)$$

Определение кратности воздухообмена (5.26, 5.27):

$$n = \frac{L}{V_{пом}} \quad (5.26)$$

$$n = \frac{7897{,}53}{384} = 20{,}56\ час^{-1} \quad (5.27)$$

5.3 Выбор кондиционера

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. Он 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 7897,53 м3/час то будет использовано один кондиционер BALLU BCFB/OUT-48HN1, который выдает необходимый нам расход воздуха.

Таблица 5.3.1 – Характеристики выбранного кондиционера.

Электропитание	Расход воздуха внутреннего блока	Расход воздуха внешнего блока	Производительность по теплу	Производительность по холоду	Мощность компрессора	Электронагреватель
В	$м^3/ч$	$м^3/ч$	кВт	кВт	кВт	кВт
380/3/50	8000	900	15,24	5.2	5,37	6

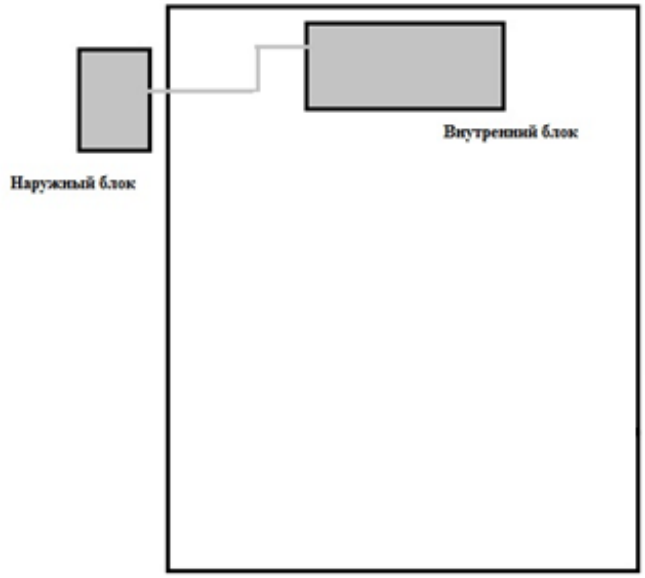

Рисунок 5.3.1 – Схема расположение кондиционера в производственном помещении

5.4 Вывод

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 жизнидеятельности учавствует во всех аспектах жизнидеятельности человека в любых профессиях и быте. На примере разработке 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 помещение и от количества людей находящимся в нем а так же от 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

В результате проделанной работы, было разработано полноценное WEB-приложение. Для реализации данной работы было изучено большое количество литературы. Были выбраны самые современные технологии разработки для улучшения быстродействия и удобства при использовании.

Был создан удобный интерфейс для всех пользователей веб-сервиса, конфигуратор выполняет все планируемые действия и является универсальным, как и планировалось.

Для реализации данного дипломного проекта были выполнены все поставленные цели, без которых разработка проекты была бы невозможна:

- определение основных целей и задач;
- разделение реализации проекта на несколько частей;
- выбор самых современных технологий разработки;
- моделирование структуры функционирования веб-сайта;
- создания безопасной и многофункциональной базы данных;
- использование сборщика проектов;
- удобное использование всех модулей;
- создание и тестирование всей функциональности готового проекта;
- экономическое исследование дипломного проекта;
- устранены недостатки условий труда, связанные с вентиляцией.

В конечном результате был получен разработанный продукт, который не имеет себе аналогов по функциональности и информативности, что делает разработку актуальной и легко внедряемой. Была произведена отладка и проверка всего функционала WEB-приложения, и в итоге получился готовый продукт, на котором любой пользователь сети интернет может пройти регистрацию, авторизацию. Может купить готовый компьютер или может изменить конфигурацию любого собранного компьютера, используя только совместимые детали, а также может собрать свой компьютер с нуля либо купить любую деталь компьютера или непосредственно периферию отдельно. Следовательно, был получен продукт, который является интернет-магазином и имеет в себе конфигуратор технологии по своему назначению.

Список литературы

1 «Node.js в действии» / Майк Контелон; [пер. с англ. Макр Хатера]. – Чайка, 2019. – 540 с.: ил.

2 «HTML and CSS: Design and Build Websites» / Джон Даккет., Майк Контелон – Москва: «Эксмо», 2019. – 283 с., ил.

3 «CSS-каскадные таблицы стилей. Подробное руководство» /Майэр Э.А.; [пер. с англ. Марк Хатера]. – Москва: Символ-Плюс, 2018. -780 с.: ил.

4 «Динамические сайты на HTML, CSS, JavaScript и Bootstrap.» / Е.В Дубовик. – Москва: Наука и Техника, 2017. – 430 с., ил.

5 «JavaScript и jQuery» / Д. С. Макфарланд. – Москва: Издательство «Эксмо», 2015. – 220 с.

6 «JavaScript и jQuery. Интерактивная веб-разработка» / Джон Дакетт; [пер. с англ. М. А. Райтмана]. — Москва: Эксмо, 2017. — 640 с.: ил.

7 «Функциональное программирование на JavaScript» / Л. Атенсио. – Москва.: «Вильямя Альфа-книга», 2018. – 450 с.

8 «Изучаем программирование на JavaScript» / Э. Фримен. – СПб.: Питер, 2015. – 367 с.

9 «Backend разработка на JavaScript и node.js» / А.М Заяц., Н.П Васильев – Москва: Лань, 2019. – 1100.

10 «Шаблоны проектирования Node.JS» / М. Кассиаро, Л. Маммино – Москва: «ДМК Пресс», 2017. – 1340 с., ил.

11 «Front-end. Клиентская разработка для профессионалов. Node.js, ES6, REST» / К. Аквино; – спб: Питер, 2017. – 420 с.

12 «Java Persistence API и Hibernate» / Г.Георги., Кинг Г. Н. – СПб: ДМК Пресс, 2017. – 823 с.

13 «Изучаем jQuery 1.3. Эффективная веб-разработка на JavaScript» / Дж. Чаффер, К. Шведберг. – Москва: издательство «Символ-плюс», 2017. – 1211 с.

14 «Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript» / И. Браун – Москва.: «Питер», 2016.

15 «Angular и TypeScript. Сайтостроение для профессионалов» / Я. Файн. – СПб.: «Питер», 2018.

16 «MySQL по максимуму. Оптимизация, репликация, резервное копирование» / П. Зайцев – СПб.: «Питер», 2018.

17 «MySQL. Сборник рецептов» / Д. Дюбуа. – СПб.: Символ-Плюс, 2019. – 523 с.: ил.

18 «SQL для профессионалов. Программирование» / Д. Селко. – СПб.: «Лори», 2018.

19 «SQL и реляционная теория.» / К.Д Дейт. – 201. URL: http://kkkn.if.ua/book/files/SQL-i-relacionnaja-teoria.pdf.

20 «CSS для профи» / К. Грант – СПб: «Питер», 2019. – 190 с.

Приложение А
(Обязательное)

Техническое задание

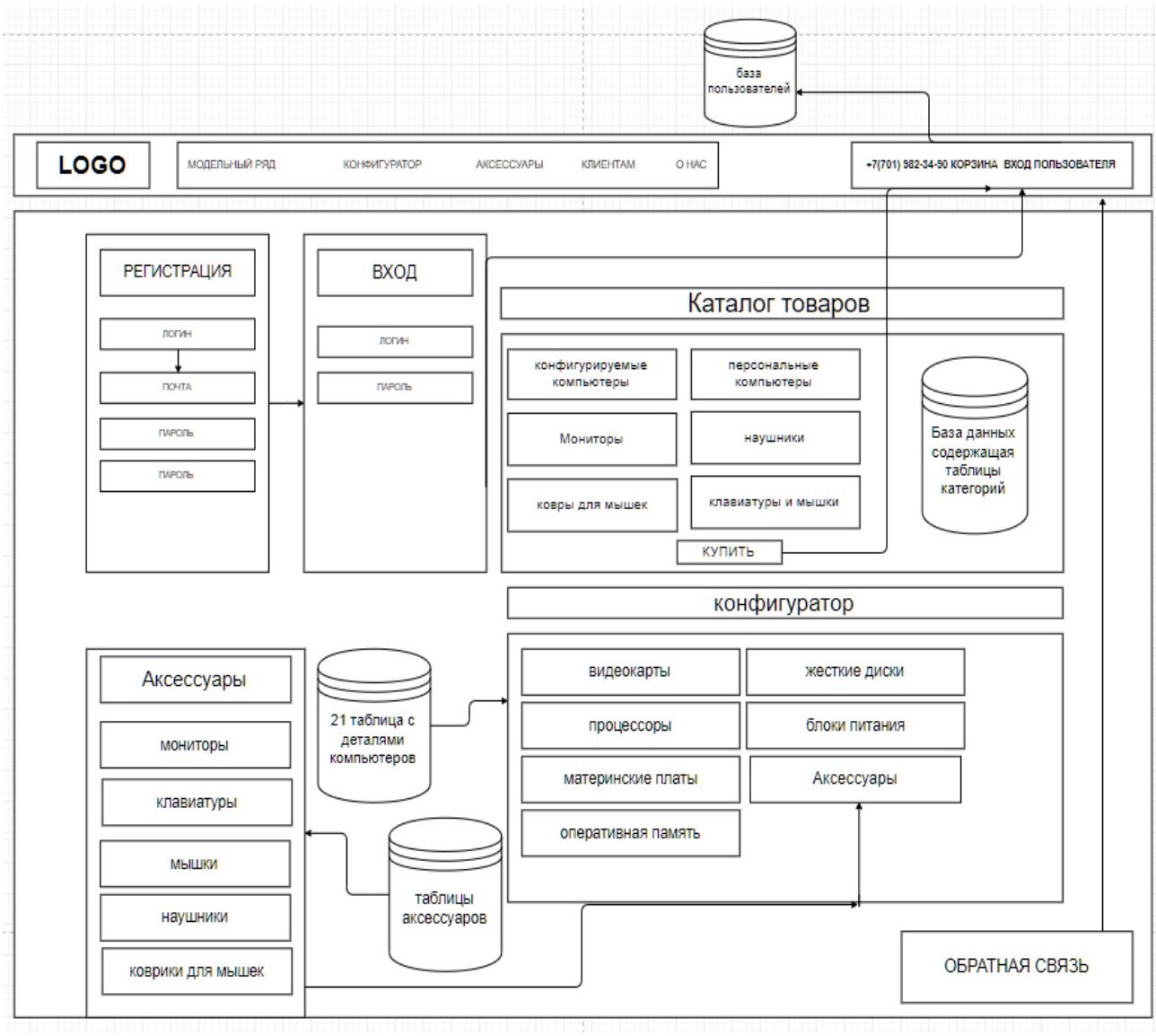

Рисунок А.1 – Техническое задание

Приложение Б
(Обязательное)

Листинг программы

```
function Registrate() {
   $.ajax({
      method: "POST",
      url: '/registr',
      cache: false,
      data: $('.form1').serialize(),
      statusCode: {
         418: function() {
            console.log("fail")
         }
      },
      success: function(datas) {
         console.log("Normalin")
      }
   })
}
$('.modalbuttonnum0n1').click(function(){
    $.ajax({
      method: "POST",
      url: '/sendemail',
      cache: false,
      data: $('.ffirsamqwe1123').serialize(),
      statusCode: {
         418: function() {
            console.log("fail")
         }
      },
      success: function(datas) {
         console.log("Normalin")
      }
   })
})
$(document).ready(function(){
   $('.linknum0ne').on("click","a", function(event){
         event.preventDefault();
         var id = $(this).attr('href'),
         top = $(id).offset().top;
         $('body,html').animate({scrollTop: top}, 1500);
   });
});
$(document).ready(function(){
   $('.hello').on("click","a", function(event){
         event.preventDefault();
         var id = $(this).attr('href'),
         top = $(id).offset().top;
```

```
 $('body,html').animate({scrollTop: top}, 1500);
   });
});
$(document).ready(function(){
   $('.menunavigation').on("click","a", function(event){
         event.preventDefault();
         var id = $(this).attr('href'),
         top = $(id).offset().top;
         $('body,html').animate({scrollTop: top}, 1500);
   });
});
$('#heandClick123').click(function(){
    Registrate1()
})
function Registrate1(){
       $.ajax({
       method: "POST",
       url: '/login',
       cache: false,
       data: $('.form2').serialize(),
       statusCode: {
          418: function() {
             console.log("fail")
          }
       },
       success: function(datas) {
          if (datas.message=="ok"){
             location.reload()
          }
       }
   })
}
function getUser(){
    $.ajax({
       method: "POST",
       url: '/user',
       cache: false,
       statusCode: {
          418: function() {
             console.log("fail")
          }
       },
       success: function(datas) {
          console.log(datas)
          if(datas.message){
          var pereme = document.getElementById('login01');
          pereme.innerHTML="";
          var button = document.createElement("button");
          var p = document.createElement("p");
```

```
var div = document.createElement("div")
          $(div).addClass("divloginout");
          $(button).addClass("buttonlogout");
          $(p).addClass("textlogin");
          button.setAttribute("id","buttonoutlog");
          $(button).on("click",function(){
                    $.ajax({
       method: "GET",
       url: '/logout',
       cache: false,
       statusCode: {
          418: function() {
             console.log("fail")
          }
       },
       success: function(datas) {
          if (datas.message=="ok"){
             location.reload()
          }
       }
    })
          });
          p.innerHTML=datas.message;
          button.innerHTML="Выйти"
          div.appendChild(p);
          div.appendChild(button);
          document.getElementById('login01').appendChild(div);
          }
       }
    })
}
$(document).ready(function(){
    getUser()
 $('#heandClick12').click(function(){
   $('.form1').css('z-index','0') && $('#vhodilka2').css('display','none')&&
   $('.form2').css('z-index','1') && $('#traif').css('display','block')
 })
  $('#heandClick321').click(function(){
   $('.form2').css('z-index','0') && $('#traif').css('display','none')&&
   $('.form1').css('z-index','1') && $('#vhodilka2').css('display','block')
 })
})
$(document).ready(function(){
       $.ajax({
       method: "GET",
       url: '/computers',
       cache: false,
       success: function(datas) {
       for (i=0; i<datas.length; i++){
```

```
 var div = document.createElement("div");
        var center = document.createElement("center");
        var img = document.createElement("img");
        var p = document.createElement("p");
        var p1 = document.createElement("p");
        var div1 = document.createElement("div");
        var button = document.createElement("button");
        var button1 = document.createElement("button");
        var a=document.createElement("a");
        $(div).addClass("intodiv");
        $(center).addClass("imgdiv");
        $(p).addClass("text22");
        $(p1).addClass("text23");
        $(img).addClass("imgimgimg3");
        $(div1).addClass("groupbutt");
        $(button).addClass("bittion1");
        $(button1).addClass("bittion2");
        img.setAttribute("width","200px");
        img.setAttribute("height","220px");
        a.setAttribute("href","#workd1");
        button1.setAttribute("name",datas[i].name);
        img.src = datas[i].img;
        p.innerHTML = datas[i].name;
        p1.innerHTML ="Цена"+" "+ datas[i].cost;
        button.innerHTML = "Купить";
        button1.innerHTML = "Конфигуратор";
        a.appendChild(button1)
        div1.appendChild(button)
        div1.appendChild(a);
        center.appendChild(img);
        div.appendChild(center);
        div.appendChild(p);
        div.appendChild(p1);
        div.appendChild(div1);
        var parent = document.getElementById("divintro");
        parent.appendChild(div);
        button.onclick = function(event){
           $('.cartopened').css("display","block");
           var all_cart_div = document.createElement("div");
           var img_cart_div = document.createElement("div");
           img_cart.setAttribute("width","60px");
           img_cart.setAttribute("height","auto");
           path_cart.setAttribute("d","M12 0c-6.627 0-12 5.373-12 12s5.373 12 12 12 12-5.373
12-12-5.373-12-12-12zm4.151 17.943l-4.143-4.102-4.117 4.159-1.833-1.833 4.104-4.157-4.162-
4.119 1.833-1.833 4.155 4.102 4.106-4.16 1.849 1.849-4.1 4.141 4.157 4.104-1.849 1.849z")
           p1_cart.innerHTML = $(event.target.parentNode.parentNode).find(".text22").text();
           p2_cart.innerHTML = $(event.target.parentNode.parentNode).find(".text23").text();
           img_cart.src = $(event.target.parentNode.parentNode).find("img").attr("src");
           svg_cart.appendChild(path_cart);
```

```
            img_cart_div.appendChild(img_cart);
             text_cart_div1.appendChild(p1_cart);
             text_cart_div2.appendChild(p2_cart);
             svg_cart_div.appendChild(svg_cart);
             all_cart_div.appendChild(img_cart_div);
             all_cart_div.appendChild(text_cart_div1);
             all_cart_div.appendChild(text_cart_div2);
             all_cart_div.appendChild(svg_cart_div);
              var id_cart = document.getElementById("cartdivmainidf1");
              id_cart.appendChild(all_cart_div);
            svg_cart.onclick = function(event){
                all_cart_div.remove();
                removeatrue();
            }
atrue();
        }
           a.onclick = function(event){
        var id = $(this).attr('href'),
        top = $(id).offset().top;
        $('body,html').animate({scrollTop: top}, 1500);
        $.ajax({
      method: "POST",
      url: '/proverka123',
      data:{data:event.target.name},
      cache: false,
      success: function(datas) {
       document.getElementById(datas[0].videocarta).click();
       document.getElementById(datas[0].processor).click();
       document.getElementById(datas[0].materinskayaplata).click();
       document.getElementById(datas[0].opetativnayapamat).click();
       document.getElementById(datas[0].zvookovaia).click();
       document.getElementById(datas[0].harddisk).click();
       document.getElementById(datas[0].blockpit).click();
       document.getElementById("roomkavoki").click();
       document.getElementById("roomkavoki1").click();
       document.getElementById("roomkavoki2").click();
       document.getElementById("roomkavoki3").click();
       document.getElementById("roomkavoki8").click();
       document.getElementById("roomkavoki9").click();
       document.getElementById("roomkavoki10").click();
       document.getElementById("linkoneroofler").click();
       document.getElementById("linkoneroofler1").click();
       document.getElementById("linkoneroofler2").click();
       document.getElementById("linkoneroofler3").click();
       document.getElementById("linkoneroofler8").click();
       document.getElementById("linkoneroofler9").click();
       document.getElementById("linkoneroofler10").click();
      }
        })
```

```
}
        }
        }
        })
        })
app.set('view engine', 'ejs');
app.use('/static', express.static('./static'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended:false}));
var options = {
host: 'localhost',
user: 'root',
password: '',
database: 'diplomdb'
}
var sessionStore = new SessionStore(options)
var cfg;
app.use(session(cfg ={
key : 'session_name',
secret : 'session_secret',
resave : true,
store: sessionStore,
saveUninitialized: true ,
cookie: {
}
}));
var connection = mysql.createConnection({
host: 'localhost',
user: 'root',
password: '',
database: 'diplomdb'
});
connection.connect(function(error) {
if(error){
console.log('Error');
} else{
console.log('Connected');
}
});
var sess;
app.get('/',function (req, res, next) {
sess = req.session;
if(sess.name){
res.render('index',{});
}else{
res.render('index',{});
}
});
app.get('/inform', function (req, res,next) {
```

```
res.render('inform', {});
})
app.get('/user', function (req, res,next) {
res.render('user', {});
})
app.post('/registr',function (req, res, next) {
console.log(req.body.username)
var reqObj = req.body;
var insertSql = "Insert into userdata SET ?";
var insertValues = {
"login":reqObj.username,
"password":reqObj.pass,
"mail":reqObj.email
};
var query = connection.query(insertSql, insertValues, function (err) {
if(err){
console.log('SQL error' + err);
}else {
res.json({message: "Data in database"})
}
})
})
app.post('/login', function (req, res, next) {
console.log(req.body)
connection.query('select login, password from userdata where login = "'+req.body.username+'"',
function (error, rows, fields) {
if(!!error){
console.log('Error in the query');
} else
{
console.log(rows)
if(rows.length>0)
{
if(req.body.pass == rows[0].password)
{
res.status(200);
req.session.name = req.body.username;
res.json({message: 'ok'});
} else
{
res.status(418);
res.json({message: 'Неверный пароль'});
}
} else
{
res.status(202);
res.json({message: 'Данного пользователя не существует'});
}
}
```

```
});
});
app.post('/user',function(req,res){
sess = req.session;
if(sess.name) {
res.json({message: sess.name});
} else {
res.status(202);
res.json({error: 'Вы не вошли (Или от вас воняет)'});
}
});
app.get('/logout',function(req,res){
var sid = req.session.id;
req.session.destroy(function(err) {
if(err) {
console.log(err);
} else {
console.log(sid);
res.json({message: 'ok'});
}
});
});
app.get('/computers', function (req, res, next) {
connection.query('select name,cost,img from computers', function (error, rows, fields) {
if(!!error){
console.log('Error in the query');
} else
{
        res.json(rows);
}
});
});
app.get('/computerforbuy', function (req, res, next) {
connection.query('select name,cost,img from computerforbuy', function (error, rows, fields) {
if(!!error){
console.log('Error in the query');
} else
{
        res.json(rows);
}
});
});
app.get('/monitori', function (req, res, next) {
connection.query('select name,cost,img from monitori', function (error, rows, fields) {
if(!!error){
console.log('Error in the query');
} else
{
        res.json(rows);
```

```
}
});
});
app.get('/klaviaturimishki', function (req, res, next) {
connection.query('select name,cost,img from klaviaturimishki', function (error, rows, fields) {
if(!!error){
console.log('Error in the query');
} else
{
	res.json(rows);
}
});
});
app.post('/proverka123', function (req, res, next) {
connection.query('select
name,cost,img,videocarta,processor,materinskayaplata,opetativnayapamat,zvookovaia,harddisk,bl
ockpit from confcomps  where name = "'+req.body.data+'"', function (error, rows, fields) {
if(!!error){
console.log('Error in the query');
} else
{
	res.json(rows);
}
});
});
app.post('/prosessor123', function (req, res, next) {
connection.query('select name,cost,img,information from prossesor  where name =
"'+req.body.data+'"', function (error, rows, fields) {
if(!!error){
console.log('Error in the query');
} else
{
	res.json(rows);
}
});
});
app.post('/zvookoviekarty123', function (req, res, next) {
connection.query('select name,cost,img,information from zvookoviekarty  where name =
"'+req.body.data+'"', function (error, rows, fields) {
if(!!error){
console.log('Error in the query');
} else
{
	res.json(rows);
}
});
});
app.post('/harddisk123', function (req, res, next) {
```

```
connection.query('select name,cost,img,information from harddisk  where name =
"'+req.body.data+'"', function (error, rows, fields) {
if(!!error){
console.log('Error in the query');
} else
{
            res.json(rows);
}
});

});
app.post('/blockpit123', function (req, res, next) {
connection.query('select name,cost,img,information from blockpit  where name =
"'+req.body.data+'"', function (error, rows, fields) {
if(!!error){
console.log('Error in the query');
} else
{
            res.json(rows);
}
});

});
app.post('/videocard123', function (req, res, next) {
connection.query('select name,cost,img,information from videocard  where name =
"'+req.body.data+'"', function (error, rows, fields) {
if(!!error){
console.log('Error in the query');
} else
{
            res.json(rows);
}
});
});
app.post('/operativka123', function (req, res, next) {
connection.query('select name,cost,img,information from operativka  where name =
"'+req.body.data+'"', function (error, rows, fields) {
if(!!error){
console.log('Error in the query');
} else
{
            res.json(rows);
}
});
});
app.post('/materinskiepl123', function (req, res, next) {
connection.query('select name,cost,img,information from materinskiepl  where name =
"'+req.body.data+'"', function (error, rows, fields) {
if(!!error){
```

Приложение В
(Обязательное)

Акты внедрения

Акт внедрения веб-приложения «GeorgePC»

Настоящий Акт свидетельствует, что веб-приложение «GeorgePC», разработанное Хиониди Георгием, внедрено в ИП «Черткова Н. О.».

Процесс внедрения проходил с 12 по 15 мая 2019 г.

Заявленные характеристики системы предполагали наличие следующих основных возможностей:

- заполнение и редактирование данных;
- покупка персональных компьютеров и периферии;
- сборка компьютеры из деталей;
- авторизация пользователей.

В ходе опытной эксплуатации веб-приложения подтверждено, что оно обладает всеми заявленными возможностями.

На момент подписания настоящего Акта система установлена и эксплуатируется сотрудниками данной компании. Программой пользуются 10 человек.

Генеральный директор