**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc101270086)

[1 ОБЗОР ЛИТЕРАТУРЫ 7](#_Toc101270087)

[1.1 Обзор аналогов 7](#_Toc101270088)

[1.1.1 Thomann.de 7](#_Toc101270089)

[1.1.2 Guitarland.by 8](#_Toc101270090)

[1.2 Обзор технологий 9](#_Toc101270091)

[1.2.1 Ruby 10](#_Toc101270092)

[1.2.2 JavaScript 11](#_Toc101270093)

[1.2.3 Ruby on Rails 11](#_Toc101270094)

[1.2.4 Шаблон проектирования MVC 12](#_Toc101270095)

[1.2.5 Архитектура клиент-сервер 13](#_Toc101270096)

[1.2.6 База данных 14](#_Toc101270097)

[1.3 Постановка задачи 15](#_Toc101270098)

[2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ 16](#_Toc101270099)

[2.1 Блок базы данных 16](#_Toc101270100)

[2.2 Блок авторизации пользователя 17](#_Toc101270101)

[2.3 Блок администрирования 18](#_Toc101270102)

[2.4 Блок пользовательского интерфейса 18](#_Toc101270103)

[2.5 Блок взаимодействия с корзиной 18](#_Toc101270104)

[2.6 Блок веб-сервера 18](#_Toc101270105)

[2.7 Блоки управления и взаимодействия с позициями 20](#_Toc101270106)

[3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 21](#_Toc101270107)

[3.1 Описание модели данных 21](#_Toc101270108)

[3.2 Описание структуры и взаимодействия между классами 24](#_Toc101270109)

[ЗАКЛЮЧЕНИЕ 34](#_Toc101270110)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 35](#_Toc101270111)

[ПРИЛОЖЕНИЕ А 36](#_Toc101270112)

[ПРИЛОЖЕНИЕ Б 37](#_Toc101270113)

[ПРИЛОЖЕНИЕ В 38](#_Toc101270114)

# **ВВЕДЕНИЕ**

В последние годы наметилась тенденция к тому, что люди все больше и больше совершают покупки в онлайн магазинах. Миллионы транзакций ежеминутно совершают люди покупая товары онлайн. Онлайн магазины предоставляют нам доступ к широкому ассортименту товаров в различных формах и размерах, что значительно экономит время, которое человек проводит в очередях при поиске и подборе товара. Немаловажным преимуществом интернет-магазинов является возможность доставки приобретенного товара, что крайне актуально в нынешней непростой эпидемиологической ситуации, так как исключает надобность посещения человеком места торговли товаром. Также онлайн магазины позволяют экономить средства производителя за ненадобностью содержания физического магазина.

На данный момент в ряде крупных стран интернет-торговля занимает 20% товарооборота. Из вышеприведенных фактов следует, что данная сфера пользуется популярностью и будет в дальнейшем расти, и, следовательно, создание нового сервиса для онлайн-продаж является целесообразным. Интернет-аудитория увеличивается каждый день, ведущие сервисы по продаже товаров совершенствуются, однако это не является препятствием для создания нового сервиса. При подробном анализе рынка существующих сервисов, нетрудно определить преимущества и недостатки ведущих онлайн-магазинов. Наиболее важной проблемой при создании онлайн магазина является сохранность данных кредитных карт. Уровень безопасности транзакций крайне важен для всех сторон продажи товара. В целях достижения максимального уровня безопасности транзакций и хранения данных о платежных картах покупателей, разработчик должен предусмотреть множество уязвимостей в программном обеспечении интернет-магазина.

Задачей дипломного проекта является разработка веб-приложения по продаже музыкального оборудования. Для выполнения поставленной задачи необходимо произвести обзор аналогов и, на основе анализа и корректно выстроенного плана разработки, создать конкурентноспособный продукт. Для создания работоспособного продукта потребуется уделить внимание вопросу разработки клиент-серверных приложений и выполнить проектирование проекта.

В соответствии с поставленной целью были определены следующие задачи:

1. Выбор платформы создания системы.

2. Разработка пользовательского интерфейса.

3. Разработка протокола взаимодействия клиентского интерфейса с серверной частью программного модуля.

4. Тестирование программного модуля.

5. Расчет экономических затрат на создание проекта.

# **1 ОБЗОР ЛИТЕРАТУРЫ**

## **1.1 Обзор аналогов**

В данном подразделе рассмотрим существующие аналоги реализуемого программного обеспечения.

### **1.1.1 Thomann.de**

Thomann.de [1] (см. рисунок 1.1) – немецкий специализированный сайт по продаже музыкального оборудования. Главная страница представлена удобным интерфейсом с приятным оформлением, без лишнего нагромождения.

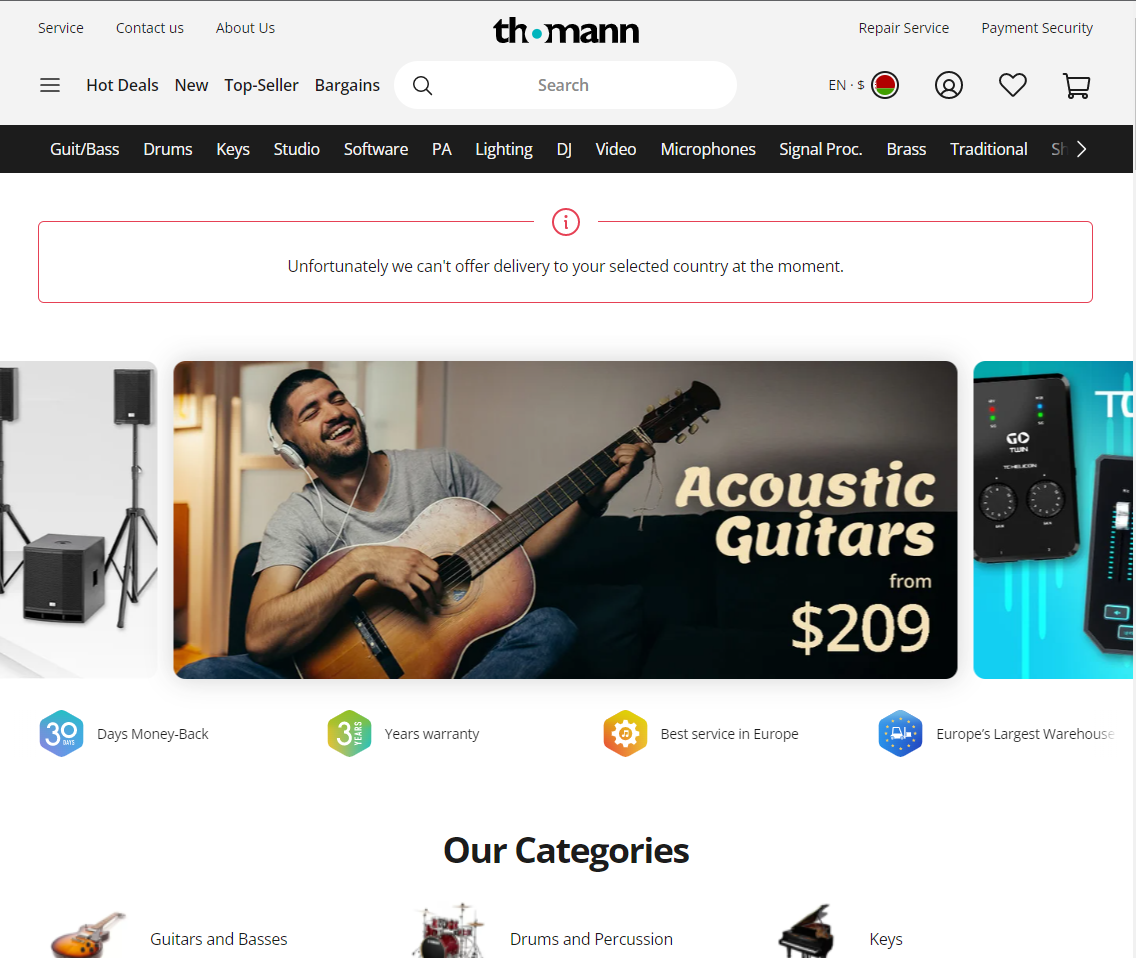


Рисунок 1.1 – Главная страница Thomann.de [1]

На сайте отсутствует явная реклама. Однако на сайте присутствует неуместный раздел – раздел новостей. В случае если пользователь захочет узнать новости по тематике определенного звукового оборудования, то он воспользуется специализированным сайтом. В данном случае функционал раздела новостей является избыточным. На данном сайте можно найти большое количество разнообразной музыкальной техники: от бас-гитар до световых установок. Существенным недостатком является отсутствие локализации для русского языка, что может принести некотором пользователям дискомфорт при пользовании сайтом и оформлении заказов.

Резюмируя вышеперечисленное, можно выделить явные достоинства и недостатки.

Достоинства:

- широкий ассортимент товара;

- практичный интерфейс;

- удобная адаптация для мобильных устройств.

Недостатки:

- отсутствие русской локализации;

- невозможность в данный момент доставки в Республику Беларусь.

### **1.1.2 Guitarland.by**

Guitarland.by [2] (см. рисунок 1.2) – сайт магазина музыкальных инструментов. Деятельность данного сайта направлена на более узкий спектр музыкального оборудования.

Здесь в ассортименте представлены инструменты для живого исполнения и записи. Однако кроме нужных разделов на сайте присутствует достаточное количество лишних разделов. Таковыми являются новости и события, которые можно прочитать на новостных сайтах. Вторым таким разделом является статьи Guitarland, который было бы целесообразнее вести в социальных сетях магазина, а на самом сайте размещать ссылки на страницы в социальных сетях. Существенным недостатком данного сайта является отсутствие прямой связи с продавцом напрямую через сайт. Также сайт не оптимизирован для использования в мобильной версии, что ухудшает удобство просмотра сайта.

Исходя из предложенных ранее фактов, резюмируем достоинства и недостатки данного ресурса.

Достоинства:

- узконаправленность (удобство для клиента, который знает чего хочет);

- наличие сезонных скидок на большое количество товаров;

- торговое помещение в центре Минска.

Недостатки:

- отсутствие адаптации для мобильных устройств;

- присутствие большого количества лишней информации на странице сайта.



Рисунок 1.2 – Главная страница Guitarland.by [2]

## **1.2 Обзор технологий**

Разрабатываемая в рамках данного дипломного проекта программная система является веб-приложением. Под веб-приложением понимается клиент-серверное приложение, в котором роль клиента играет браузер, а сервером является веб-сервер, который может хранится в облачном хранилище. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Благодаря такому подходу клиент может не зависеть от конкретной операционной системы. Из этого следует, что веб-приложения являются кроссплатформенными сервисами.

Разрабатываемая программная система имеет монолитную архитектуру. Концепция монолитного программного обеспечения подразумевает, что различные компоненты приложения объединяются в одну программу на одной платформе. Обычно монолитное приложение состоит из базы данных, клиентского пользовательского интерфейса и серверного приложения. Все функции такого приложения управляются в одном месте.

Для разработки серверной части приложения использовался паттерн MVC. MVC представляет из себя архитектурный шаблон, который означает, что приложение MVC будет разделено на три части, а именно:

- модель;

- представление;

- контроллер.

Модель представляет из себя данные, передаваемые между представлениями и контроллерами, а также модель может содержать операции и преобразования для манипулирования этими данными.

Представления применяются для визуализации части модели в виде пользовательского интерфейса.

Контроллеры обрабатывают поступающие запросы, выполняют операции с моделью и выбирают представления для визуализации пользователю.

В большинстве случаев для разработки веб-сервера используются следующие языки программирования: Java, JavaScript, Python, PHP, C#, Ruby.

**1.2.1 Ruby**

Для написания веб-сервера данного дипломного проекта используется язык программирования Ruby [3]. Ruby – это интерпретируемый, высокоуровневый, динамичный, универсальный язык программирования с открытым исходным кодом, который фокусируется на простоте и производительности. Он был спроектирован и разработан в середине 1990-х годов Юкихиро Мацумото в Японии.

Ruby – это язык динамического программирования со сложной, но выразительной грамматикой и базовой библиотекой классов с богатым и мощным API. Ruby - вобрал в себя черты таких языков, как Lisp, Smalltalk и Perl, но использует грамматику, которой без особого труда смогут овладеть программисты, работающие на языках С и Java.

Ruby является абсолютным объектно-ориентированным языком, но в нем также неплохо уживаются процедурные и функциональные стили программирования. Приоритетом Ruby является удобство и минимизация затрат труда программиста при разработке программы, освобождение программиста от рутинной работы, которую компьютер может выполнять быстрее и качественнее. Особое внимание, в частности, уделено будничным рутинным занятиям (обработка текстов, администрирование), и для них язык настроен особенно хорошо. В противоположность машинно-ориентированным языкам, работающим быстрее, Ruby – язык, наиболее понятный человеку, даже очень далекому от программирования. Любая работа с компьютером выполняется людьми и для людей, и необходимо заботиться в первую очередь о затрачиваемых усилиях людей [4].

### **1.2.2 JavaScript**

Для разработки пользовательского интерфейса используется язык программирования JavaScript и библиотека jQuery.

JavaScript работает на клиентской стороне приложения и используется для определения того, как веб-страницы ведут себя при возникновении события [5]. JavaScript является простым в изучении, широко используемым, а также мощным языком для управления поведением веб-страниц.

JQuery – легковесная библиотека JavaScript, слоганом которой является “пиши меньше, делай больше”. Цель jQuery - сделать использование JavaScript на проекте намного проще. JQuery берет много общих задач, для выполнения которых требуется много строк JavaScript-кода, и обертывает их в методы, которые можно вызвать одной строкой. JQuery также упрощает многие сложные вещи из JavaScript, такие как вызовы AJAX и манипуляции с DOM элементами [6].

### **1.2.3 Ruby on Rails**

Ruby on Rails – фреймворк, написанный на языке программирования Ruby с открытым исходным кодом, реализует архитектурный шаблон Model-View-Controller для веб-приложений [7]. Ruby on Rails включает в себя инструменты, которые облегчают общие задачи разработки «из коробки», такие как scaffolding, которые могут автоматически генерировать некоторые модели и представления, необходимые для базового сайта. В конфигурации по умолчанию модель отображает одну из таблицы в базе данных. Например, класс модели User обычно определяется в файле «user.rb» в каталоге app/models и привязывается к таблице «users» в базе данных.

Контроллер – это компонент Rails, который отвечает на внешние запросы от веб-сервера к приложению, определяя, какой файл отображения нужно отрисовать. Контроллеру также может потребоваться запросить одну или несколько моделей для получения информации и передать их в отображение.

Ruby on Rails также заслуживает внимания за широкое использование библиотек JavaScript для написания Ajax-запросов. Для этого прекрасно подойдет jQuery, который полностью поддерживается как замена Prototype и поддерживается по умолчанию в Rails с версии 3.1.

Основным преимуществом языка программирования Ruby и фреймворка Ruby on Rails является скорость разработки. На практике скорость разработки проектов на Ruby on Rails выше на 30-40 процентов по отношению к любому другому языку программирования или фреймворку. Такой прирост к скорости разработки объясняется обширным набором готовых к работе штатных инструментов Ruby on Rails, возможностью использовать готовые решения других разработчиков и удобством программирования на Ruby.

Кроме того, в отличие от других фреймворков, в составе Ruby on Rails есть отличные средства автоматизированного тестирования, что ускоряет переход проекта от стадии «программа написана» к стадии «программа работает без ошибок». Зачастую именно этот переход отнимает больше всего времени при реализации практически любого проекта.

### **1.2.4 Шаблон проектирования MVC**

Шаблон проектирования MVC [8] (см. рисунок 1.3) – это не столько шаблон, сколько метод или идея организации приложения, его структуры. Сама идея заключается в том, чтобы отделить бизнес логику приложения от его представления. Как следует из названия MVC (Model – View – Controller) состоит из трех основных компонентов: модель, представление и контроллер.

При этом модификация каждого из трех модулей происходит независимо. Данный подход описывает один из принципов SOLID – принцип единой ответственности. Это значит, что каждый объект (модуль) имеет одну ответственность и все поведение объекта должно быть направлено на выполнение только этой одной ответственности.

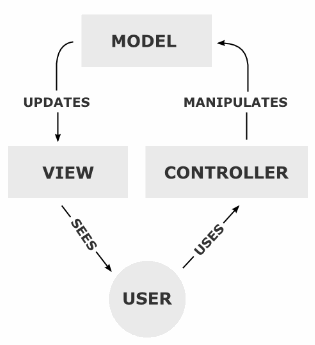
****

Рисунок 1.3 – Шаблон MVC [8]

Модель в шаблоне MVC – основная часть логики, которая отвечает за выборку данных из базы, изменение данных, расчеты. Вообще модель можно представить, как набор каких-либо функций, некий ящик, который принимает в себя данные, обрабатывает их и возвращает обратно. Данные в модель приходят из контроллера, затем эти данные обрабатываются, будь то запросы к базе данных, либо же бизнес вычисления. Данные, пришедшие из контроллера, могут сохраняться в базе данных. После обработки данных моделью, результат работы возвращается обратно в контроллер. В больших приложениях, как правило, имеется много моделей, каждая из которых отвечает за конкретный модуль, будь то создание пользователей или вывод списка записей в таблицу. В модели определяется ограничения по обрабатываемым данным и правила манипуляции данными.

Представление отвечает за вывод данных для пользователя. Обычно представление содержит в себе мало логики и предназначено только для отображения результата работы модели. В современных сайтах большое внимание уделяется клиентской части приложения. Поэтому скрипты в представлениях порой достигают очень больших размеров, что замедляет работу приложения. Представление реагирует на действия пользователя и передает данные в контроллер. Обычно представление реализуется посредством шаблонов HTML-разметки, которые заполняются данными. Данные приходят из модели, контроллер заполняет представление и шаблон HTML документа отправляется клиенту.

Контроллер – блок в шаблоне MVC, который получает данные, нормализует их, производит валидацию данных, и после прохождения этих процедур передает данные дальше в нужную модель. С другой стороны, контроллер получает данные из модели, и выбирая нужное представление, отображает их. Контроллер не должен содержать в себе никакой бизнес логики. Вся основная бизнес-логика должна содержаться в слое модели. Контроллер выступает связующим звеном между моделью и представлением.

### **1.2.5 Архитектура клиент-сервер**

Обычно компьютеры, входящие в какую-либо систему, не являются равноправными. Так же, как и в социальной группе есть лидер и подчиненные. Некоторые компьютеры несут в себе информацию, а другие же пользуются этой информацией посредством запросов к главному компьютеру.Сервером называется компьютер, который хранит в себе информацию, и который обрабатывает полученные от клиента запросы, а затем выдает ответ. Как правило, на сервере установлено специальное программное обеспечение для выполнения какой-либо сервисной задачи без участия человека. В свою очередь веб-сервером называют как программное обеспечение, которое выполняет функции сервера, так и сам компьютер, на котором это программное обеспечение работает. Клиентом же называют компьютеры, которые посылают запросы на сервер и получают ответ. Клиент и сервер могут находиться как в рамках одной сети, так и в рамках различных сетей. Предполагается, что приложение запущено и работает на сервере.

Суть клиент-серверной архитектуры заключается в разделении приложения на три группы:

- отображение и ввод данных для пользователя;

- функции связывающие отображения и бизнес-логику;

- функции управления ресурсами (работа с базой данных).

Из этого следует, что любое приложение можно разделить на три компонента:

- компонент отображения;

- прикладной компонент;

- компонент управления ресурсами.

Отправка данных от клиента серверу осуществляется с помощью приложения, называемого браузером. Браузер не имеет привязки к операционной системе, поэтому веб-приложения являются кроссплатформенными. Браузер в свою очередь интерпретирует HTML документ своим образом и отображается пользователю. Чтобы браузер мог отображать данные для пользователя, серверу необходимо посылать ответ в формате HTML.

Веб-сервер отвечает за обработку данных, полученных от клиента. Веб-сервер получает данные от клиента, активизируя действие по определенному пути, передает данные в логические модули программы. После этого приложение обрабатывает полученный запрос и веб-сервер формирует HTML страницу, которую отправляет назад клиенту. Сервер содержит ряд шаблонов HTML, на которые накладываются данные и отсылаются клиенту.

Отличительной особенностью веб клиент-серверной архитектуры является то, что в роли клиента выступает браузер, который посылает запросы на сервер, посредством HTTP/HTTPS протоколов. Веб-сервер принимает запросы от клиента, обрабатывает данные, и посылает ответ клиенту. А взаимодействие между клиентом и сервером осуществляется через интернет соединение.

### **1.2.6 База данных**

В качестве системы управления базами данных будет использоваться PostgreSQL.

PostgreSQL — свободная объектно-реляционная система управления базами данных (СУБД). PostgreSQL является одной из наиболее популярных систем управления базами данных [9]. Сам проект postgresql эволюционировал из другого проекта, который назывался Ingres. Формально развитие postgresql началось еще в 1986 году. Тогда он назывался POSTGRES. А в 1996 году проект был переименован в PostgreSQL, что отражало больший акцент на SQL. И 8 июля 1996 года состоялся первый релиз продукта. С тех пор вышло множество версий postgresql. Текущей версией является версия 14.2. Однако регулярно также выходят подверсии. PostgreSQL поддерживается для всех основных операционных систем – Windows, Linux, MacOS.

Особенности PostgreSQL:

- высокая производительность;

- надежность;

- гибкость и простота в использовании;

- многоверсионность;

- репликация;

- масштабируемость.

## **1.3 Постановка задачи**

Сегодня рынок программного обеспечения выставляет высокие требования ко всем разрабатываемым программным продуктам. Для современных программных средств важными требованиями являются мультиплатформенность, масштабируемость и переносимость. Использование вышеперечисленных технологий при разработке дипломного проекта позволяет сократить время на разработку, увеличить качество кода и засчет этого выполнить данные требования

Объектом исследования является процесс создания объектов и обработки данных, поступающих на сервер. Предметом исследования является создание программного комплекса, обеспечивающего полноценную работу веб-приложения.  
 Основными требованиями, которые были заложены в основу при разработке программного комплекса дипломного проекта стали: простота использования и расширяемость.

На основании вышесказанного, для разработки программного продукта были определены следующие задачи:

- разработка структуры БД;

- разработка серверной части;

- разработка клиентской части.

Веб-приложение будет представлять из себя сайт, который будет предоставлять следующие функции:

- регистрация пользователя в системе;

- изменение данных пользователя;

- создания, редактирование и удаление объявления о продаже;

- отправка письма на электронную почту о получении заказа;

- администрирование пользователей, товаров.

# **2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ**

Проанализировав теоретическую часть разрабатываемой системы, был получен список требований, который необходим для эффективного, стабильного функционирования разрабатываемого веб-приложения. В ходе анализа требований было принято решение разделить программный продукт на функциональные блоки. Распределение функционала по блокам обеспечит удобство в разработке, а также предоставит возможность проверки основных характеристик блока без внешнего воздействия. Выделенные функциональные блоки:

- блок базы данных;

- блок авторизации пользователя;

- блок взаимодействия с корзиной;

- блок работы сервера;

- блок управления позициями;

- блок взаимодействия с позициями;

- блок администрирования;

- блок пользовательского интерфейса.

Структурная схема, иллюстрирующая перечисленные блоки и связи между ними приведена на чертеже ГУИР.400201.078 C1.

Рассмотрим каждый структурный блок подробнее.

## **2.1 Блок базы данных**

Блок базы данныхвключает данные, используемые веб-приложением. При реализации используется реляционная база данных PostgreSQL. От других СУБД PostgreSQL отличается поддержкой востребованного объектно-ориентированного и/или реляционного подхода к базам данных. Например, полная поддержка надежных транзакций, т.е. атомарность, последовательность, изоляционность, прочность (Atomicity, Consistency, Isolation, Durability (ACID). Существует обширный список типов данных, которые поддерживает PostgreSQL. Кроме числовых, с плавающей точкой, текстовых, булевых и других ожидаемых типов данных (а также множества их вариаций), PostgreSQL имеет преимущество в качестве поддержки uuid, денежного, перечисляемого, геометрического, бинарного типов, сетевых адресов, битовых строк, текстового поиска, xml, json, массивов, композитных типов и диапазонов, а также некоторых внутренних типов для идентификации объектов и местоположения логов.

Основными достоинствами PostgreSQL являются:

- надежность (полное соответствие принципам ACID - атомарность, непротиворечивость, изолированность, сохранность данных);

- производительность (основывается на использовании индексов, интеллектуальном планировщике запросов, тонкой системы блокировок, системе управления буферами памяти и кэширования, превосходной масштабируемости при конкурентной работе);

- расширяемость (означает, что пользователь может настраивать систему путем определения новых функций, агрегатов, типов, языков, индексов и операторов);

- поддержка SQL;

- поддержка JSON;

- богатый набор типов данных;

- простота использования.

## **2.2 Блок авторизации пользователя**

Блок авторизации пользователяявляется частью системы, которая отвечает за проверку существования пользователя и в случае его существования в системе, генерирует авторизационный токен для этого пользователя. Для данных целей будет использоваться популярная в Rails среде библиотека Devise. Данная библиотека состоит из следующих модулей:

- Database Authenticatable – хэширует и сохраняет пароль в базе данных для проверки подлинности пользователя при входе в систему;

- Confirmable – отправляет электронные письма с инструкциями по подтверждению аккаунта и проверяет подтверждена учетная записи или нет;

- Recoverable – при необходимости сбрасывает пароль и отправляет инструкции по восстановлению;

- Registerable – предоставляет механизм регистрации нового пользователя;

- Trackable – запоминает количество входов, время и IP-адрес;

- Timeoutable – закрывает открытые сессии пользователя если они не были активны заданный период времени;

- Validatable – проверяет при входе корректность введенных данных так же можно определить свои валидации;

- Lockable – блокирует аккаунт после определенного количества неудачных попыток входа. Может быть разблокирован по электронной почте или по истечении указанного периода времени;

- Rememberable – добавляет возможность «запомнить аккаунт»;

- Omniauthable – поддержка авторизации через социальные сети.

В разрабатываемом программном продукте будут использоваться только некоторые из этих модулей, а именно: Database Authenticatable, Recoverable, Rememberable, Validatable, Registerable. Так же присутствуют методы, упрощающие работу с пользователями, которые будут подробно описаны в следующих главах.

## **2.3 Блок администрирования**

Блок администрирования –это часть системы, отвечающая за управление всеми ресурсами сайта. В Rails для этого есть несколько решений, в данном проекте используется плагин Rolify. Rolify предоставляет возможность назначать зарегистрированным пользователям роли, с помощью которых каждому отдельному пользователю предоставляются доступные для данной роли действия. Для роли admin в пользовательском интерфейсе открываются дополнительные опции для доступа ко всем ресурсам приложения, а также появляется возможность для проведения любых манипуляций с данными. Кроме того, стоит отметить, что Rolify гибко настраивается в зависимости от потребностей проекта.

## **2.4 Блок пользовательского интерфейса**

Блок пользовательского интерфейса является клиентской частью веб-приложения. Данный блок представляет собой совокупность средств, при помощи которых пользователь взаимодействует с приложением через браузер. Для построения интерфейса будет использоваться несколько технологий: JavaScript, JQuery. Также в Rails есть расширение HTML файлов «.erb», которое позволяет использовать Ruby-код непосредственно в HTML-разметке и предоставляет множество заготовленных хелперов для отрисовки элементов. JQuery – используется для Ajax-запросов на сервер. Ajax – это технология, которая позволяет обновлять данные не перезагружая страницу полностью.

## **2.5 Блок взаимодействия с корзиной**

Блок взаимодействия с корзиной – это блок, основной задачей которого является хранение и помещение в него товаров пользователем. В дальнейшем из продуктов, которые хранятся в корзине, формируется заказ. Для реализации данного блока используется Redis. Redis – это быстрое хранилище данных типа «ключ-значение». Оно используется как для баз данных, так и для реализации кэша из-за быстрой скорости обработки операций (хранит базу данных в оперативной памяти).

## **2.6 Блок веб-сервера**

Блок веб-сервера *–* это блок, который обрабатывает запросы, отправленные клиентом, в качестве серверной части будет использоваться фреймворк Ruby On Rails.

Фреймворк Ruby On Rails написан с помощью языка программирования Ruby.

Ruby – интерпретируемый язык программирования высокого уровня. Язык динамический, объектно-ориентированный, реализует многопоточность, которая не зависит от операционной системы, имеет «сборщик мусора».

Ruby on Rails предоставляет из себя архитектуру MVC для веб-приложений, а также обеспечивает их интеграцию с веб-сервером и сервером базы данных. Ruby on Rails определяет следующие принципы разработки приложений, помогающие разработчику в создании элегантных программных решений, усвоенные сообществом разработчиков:

- предоставляет механизмы повторного использования, позволяющие минимизировать дублирование кода в приложениях (принцип Don’t repeat your self);

- по умолчанию используются соглашения по конфигурации, типичные для большинства приложений (принцип Convention over configuration);

- основными компонентами приложений Ruby on Rails являются модель, представление и контроллер;

- Ruby on Rails использует REST-стиль построения веб-приложений.

Модель используется для отображения данных остальным компонентам приложения. Объекты модели производят загрузку и сохранение данных в базе данных, а также реализуют бизнес-логику.

Для хранения объектов модели в СУБД по умолчанию в Rails использована библиотека Active Record.

Представление создает пользовательский интерфейс с использованием полученных от контроллера данных. Представление также передает запросы пользователя на взаимодействие с данными в контроллер (само представление обычно не изменяет модель). Контроллер в Rails – это набор логики, запускаемой после получения HTTP-запроса сервером. Контроллер отвечает за вызов методов модели и запускает формирование представления.

Вокруг Rails сложилась большая экосистема плагинов – подключаемых «гемов» (англ. gem). «Гем» – это библиотека, составленная определенным образом. То есть это набор кода (модули, классы, представления и так далее) которые решают определенную задачу. Утилита gem занимается тем, что управляет этими библиотеками. Например 'gem install colorize' скачает из интернета библиотеку (далее "гем") в каталог, в который прежде был установлен Ruby. После чего в коде можно написать require ‘colorize’ и пользоваться методами, которые данный гем предоставляет. Гем может требовать для установки другие гемы. Для того чтобы не ставить/обновлять гемы по-одному каждый раз, был создан Bundler. который сам по себе является гемом. Работает это следующим образом: в Gemfile описываются требуемые в данном конкретном проекте гемы. После чего запускается команда bundle install. Bundler просматривает Gemfile проекта и устанавливает (с помощью утилиты gem) нужные гемы, а также создает файл Gemfile.lock, в котором описывает установленный гем и, если понадобились при установке, то дополнения к данному гему.

**2.7 Блоки управления и взаимодействия с позициями**

Блоки управления объявлениями и взаимодействия с позициями призваны отвечать за работу с позициями. Через эти блоки проводятся базовые CRUD-операции с позициями. Аббревиатура CRUD в переводе с английского означает C(create) – создать, R(read) – читать, U(update) – обновлять, D(delete) – удалять. Это основные операции, используемые для реализации приложений с постоянный хранением данных и приложений реляционных баз данных.

CRUD постоянно используется для всего, что связано с базами данных и их проектированием. Разработчики программного обеспечения ничего не могут сделать без операций CRUD. Например, при разработке веб-сайтов используется REST (передача репрезентативного состояния), который является надмножеством CRUD, используемого для ресурсов HTTP.

С другой стороны, CRUD не менее важен для конечных пользователей. Без него были бы невозможны такие базовые операции как регистрация пользователя, размещение новых записей. Большинство приложений, которые мы используем, позволяют нам добавлять или создавать новые записи, искать существующие, вносить в них изменения или удалять их.

Create позволяет вносить новые строки в таблицу базы данных. Сделать это можно с помощью команды INSERT INTO. Команда начинается с ключевого слова INSERT INTO, за которым следует имя таблицы, имена столбцов и значения, которые нужно вставить.

Функция чтения соизмерима с функцией поиска, поскольку позволяет извлекать определенные записи и считывать их значения. Она относится к команде SELECT.

Обновление – это изменение существующей записи в таблице. Это используется для внесения изменений в существующие записи в таблице базы данных. При выполнении команды UPDATE определяется целевая таблица и столбцы, которые необходимо обновить.

Удаление используется для удаления записи из таблицы базы данных. SQL имеет встроенную функцию удаления для одновременного удаления одной или нескольких записей из базы данных. Некоторые приложения реляционных баз данных могут разрешать жесткое удаление (безвозвратное удаление) или мягкое удаление (обновление статуса строки).

# **3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

В данном разделе описывается функционирование и структура разрабатываемого программного средства.

Взаимоотношения между классами разрабатываемого программного обеспечения приведены на диаграмме классов ГУИР.400201.078 РР.1.

## **3.1 Описание модели данных**

**3.1.1 Таблица Users**

Данная таблица служит для хранения основных данных об аккаунте пользователя.

Поля таблицы users:

- id – первичный ключ;

- name – имя пользователя в приложении;

- email – электронная почта пользователя;

- phone\_number – номер телефона пользователя в приложении;

- reset\_password\_token – зашехированный пароль, который был выслан пользователю для восстановления аккаунта;

- reset\_password\_sent\_at – время и дата отправления пароля для восстановления аккаунта;

- created\_at – дата и время создания аккаунта;

- updated\_at – дата и время последних изменений аккаунта;

- encrypted\_password – захешированный пароль пользователя.

При создании таблицы для колонок id, encrypted\_password, email, created\_at, updated\_at был использован специальный атрибут «null: false», который задает условие создания и заполнения данных колонок так, что они не могут быть пустыми. В случае если данным колонкам не будет явно задано значение, будет выставлено значение по умолчанию. Для колонок encrypted\_password, email это значение пустой строки. Поля id, created\_at, updated\_at будут автоматически заполнены Ruby on Rails. У различных колонок могут быть разные типы данных. У колонок name, encrypted\_password, email, reset\_password\_token тип данных «character varying». Этот тип данных является символьной строкой переменной длины. Колонки id, phone\_number имеют тип «integer». Колонки reset\_password\_sent\_at, created\_at, updated\_at имеют тип timestamp.

**3.1.2 Таблица Roles**

Данная таблица служит для хранения данных о созданных в системе ролях для пользователей, которые разграничивают доступный пользователям функционал.

Поля таблицы roles:

- id – первичный ключ;

- name – название роли в приложении;

- created\_at – дата и время создания роли;

- resource\_type – составная часть сложного индекса;

- resource\_id – составная часть сложного индекса;

- updated\_at – дата и время последних изменений роли.

Колонки id, created\_at, updated\_at не могут быть пустыми. Колонка resource\_id имеет тип «integer». Колонка name имеет тип «character varying» и заполняется при создании новой роли. Колонка id имеет тип «integer». Колонки created\_at, updated\_at имеют тип timestamp и заполняется автоматически Ruby on Rails.

**3.1.3 Таблица UserRoles**

Данная таблица является связующей таблицей между таблицами Users и Roles. В ней фиксируется принадлежность пользователя к некой роли, а, соответственно, и наличие у пользователя каких-либо преференций при пользовании приложением.

Поля таблицы users\_roles:

- id – первичный ключ;

- user\_id – внешний ключ для связи с таблицей users;

- role\_id – внешний ключ для связи с таблицей roles.

Колонки id, user\_id, role\_id имеют тип «integer».

**3.1.4 Таблица Versions**

Данная таблица служит для хранения версий пользователей при их изменениях.

Поля таблицы versions:

- id – первичный ключ;

- item\_type – составная часть сложного индекса;

- item\_id – составная часть сложного индекса;

- event – событие изменения;

- whodunnit – имя пользователя, который изменил состояние предыдущей версии;

- object – объект изменения;

- created\_at – дата и время создания объекта.

При создании таблицы для колонок id, item\_type, item\_id, event был использован специальный атрибут «null: false», который задает условие создания и заполнения данных колонок так, что они не могут быть пустыми. Для колонок item\_type, event, в случае если они не будут заполнены, то они будут заполнены по умолчанию пустой строкой. Поля id, created\_at будут автоматически заполнены Ruby on Rails. У колонок item\_type, event, whodunnit тип данных «character varying». Колонки id, item\_id имеют тип «integer». Колонка created\_at имеет тип timestamp. Колонка object имеет тип text.

**3.1.5 Таблица Products**

Данная таблица хранит информацию о музыкальном оборудовании, которое выставлено на продажу и содержит его описание.

Поля таблицы products:

- id – первичный ключ;

- name – название позиции в приложении;

- photo – изображение позиции;

- price – цена данной позиции оборудования;

- created\_at – дата и время создания позиции;

- updated\_at – дата и время последних изменений позиции;

- description – описание данной позиции оборудования.

При создании таблицы для колонок id, created\_at, updated\_at использован специальный атрибут «null: false», который задает условие создания и заполнения данных колонок так, что они не могут быть пустыми. Поля id, created\_at, updated\_at будут автоматически заполнены Ruby on Rails. Поле name имеет тип данных «character varying». Поле description имеет тип text. Поле price имеет тип «integer». Поле photo имеет тип «character varying». В колонке photo будут хранится ссылки на загруженные пользователем изображения. Поля created\_at, updated\_at имеют тип timestamp. Колонка id имеет тип «integer».

**3.1.6 Таблица Orders**

Данная таблица тесно связана с таблицей users и хранит в себе информацию о созданном пользователем заказе позиций музыкального оборудования.

Поля таблицы orders:

- id – первичный ключ;

- name – имя человека, оформляющего заказ позиций;

- order\_price – полная стоимость заказа;

- user\_id – внешний ключ для связи с таблицей users;

- dest\_address – адрес, куда нужно доставить заказанные позиции;

- phone\_number – номер телефона заказчика;

- created\_at – дата и время создания заказа;

- updated\_at – дата и время последних изменений заказа.

Колонки id, user\_id, order\_price, phone\_number имеют тип «integer». Колонки dest\_address, name имеют тип «character varying». Колонки created\_at, updated\_at имеют тип timestamp.

**3.1.7 Таблица OrderProducts**

Данная таблица является связующей таблицей между таблицами Orders и Products. В ней фиксируется принадлежность заказу позиций, которые пользователь добавил себе в корзину и оформил данный заказ.

Поля таблицы users\_roles:

- id – первичный ключ;

- product\_id – внешний ключ для связи с таблицей products;

- order\_id – внешний ключ для связи с таблицей orders;

- created\_at – дата и время создания связи заказа и позиции;

- updated\_at – дата и время последних изменений связи заказа и позиции.

Колонки id, order\_id, product\_id имеют тип «integer». Колонки created\_at, updated\_at имеют тип timestamp.

## **3.2 Описание структуры и взаимодействия между классами**

При разработке приложения использовался паттерн MVC, поэтому оно имеет структуру, состоящую из контроллера, сервиса и репозитория. Также стоит отметить, что все проектируемые сервисы разработаны с соблюдением всех правил и норм REST архитектуры. Рассмотрим классы каждого из данных слоев приложения.

**3.2.1 Класс ApplicationController**

Данный класс-контроллер является главным контроллером приложения, от которого наследуются все остальные контроллеры приложения. ApplicationController в свою очередь наследуется от ActionController::Base, это базовый модуль Rails приложения, который предоставляет большое количество методов для работы с контроллерами.

Методы класса:

- configure\_permitted\_params – protected метод экземпляра, добавляет параметры к объекту пользователя;

- current\_user? – protected метод экземпляра, проверяет по идентификатору является ли данный пользователь текущим пользователем;

- token – protected метод экземпляра, записывает значение сессионного токена;

- authorize – protected метод экземпляра, выполняет перенаправление на страницу с авторизацией, если пользователь не является текущим пользователем;

- authorize? – protected метод экземпляра, возвращает токен, если пользователь не является текущим пользователем;

- basket – protected метод экземпляра, создает новую корзину для текущей сессии;

- redis\_basket – protected метод экземпляра, создает новую redis-корзину с помощью текущей записи в redis;

- after\_sign\_in\_path\_for – protected метод экземпляра, возвращает корневой путь;

- after\_sign\_out\_path\_for – protected метод экземпляра, возвращает пользователя на предыдущую страницу.

**3.2.2 Класс AuthenticationController**

Данный класс-контроллер предназначен для авторизации пользователей.

Методы класса:

- new – публичный метод экземпляра класса, используется для отображения шаблона, в котором отображаются все поля, которые необходимо заполнить при создании нового пользователя, после ввода отправляет информацию, которую ввел пользователь, для дальнейшей обработки в метод create;

- create – публичный метод экземпляра класса, используется для создания пользователя, после получения данным методом информации от пользователя, он ее обрабатывает, проверят соответствие валидаторам, заданным в модели, и, если все проверки пройдены, то сохраняет информацию, если есть какие-то ошибки возвращает их пользователю с описанием ошибки;

- destroy – публичный метод экземпляра класса, используется для удаления пользователя;

- user\_params – приватный метод экземпляра класса, используется для отделения параметров о пользователе от других параметров, которые приходят в запросе.

**3.2.3 Класс BasketsController**

Данный класс-контроллер предназначен для управления корзиной.

Методы класса:

- index – публичный метод экземпляра класса, используется для получения всех позиций, находящихся в корзине и отображения их пользователю;

- add – публичный метод экземпляра класса, используется для добавления новой корзины;

- remove – публичный метод экземпляра класса, используется для удаления корзины;

- clear – публичный метод экземпляра класса, используется для очищения корзины.

**3.2.4 Класс CheckoutController**

Данный класс-контроллер предназначен для управления оформлением заказов.

Методы класса:

- new – публичный метод экземпляра класса, используется для отображения шаблона, в котором отображаются все поля, которые необходимо заполнить при создании нового заказа, после ввода отправляет информацию, которую ввел пользователь, для дальнейшей обработки в метод create;

- create – публичный метод экземпляра класса, используется для создания заказа, после получения данным методом информации от пользователя, он ее обрабатывает, проверят соответствие валидаторам, заданным в модели, и, если все проверки пройдены, то сохраняет информацию, если есть какие-то ошибки возвращает их пользователю с описанием ошибки;

- order\_params – приватный метод экземпляра класса, используется для отделения параметров о заказе от других параметров, которые приходят в запросе.

**3.2.5 Модуль CheckoutsService**

Данный модуль используется для создания заказов и включается в CheckoutController.

Методы модуля:

- initialize – публичный метод модуля, используется для инициализации переменных класса в момент создания объекта;

- make\_order – публичный метод модуля, используется для построения структуры заказа;

- build\_form\_params – приватный метод модуля, используется для установления параметров заказа;

- make\_order\_product – приватный метод модуля, используется для записи собранных в корзине позиций в заказ.

**3.2.6 Модуль BasketsService**

Данный модуль используется для управления корзиной, созданной с помощью redis, и включается в BasketsController. Данный модуль используется в ApplicationController, ProductsController.

Методы модуля:

- initialize – публичный метод модуля, предназначен для инициализации переменных класса в момент создания объекта;

- add – публичный метод модуля, предназначен для добавления позиции в корзину;

- remove – публичный метод модуля, предназначен для удаления позиции из корзины;

- all – публичный метод модуля, предназначен для отображения всего содержимого корзины, если таковая имеется;

- clear – публичный метод модуля, предназначен для удаления корзины;

- size – публичный метод модуля, предназначен для подсчета количества позиций, которые на данный момент находятся в корзине;

- sum – публичный метод модуля, предназначен для подсчета суммы позиций, находящихся в данный момент в корзине;

- token – приватный метод модуля, записывает значение сессионного токена.

**3.2.7 Класс ProductsController**

Данный класс-контроллер используется для управления позициями, размещенными в приложении.

Методы класса:

- index – публичный метод экземпляра класса, используется для получения всех позиций, находящихся в приложении и отображения их пользователю;

- show – публичный метод экземпляра класса, используется для получения одной позиции и отображения ее пользователю, поиск данной позиции производится по идентификатору позиции;

- add\_to\_basket – публичный метод экземпляра класса, используется для добавления позиции в корзину;

- delete\_from\_basket – публичный метод экземпляра класса, используется для удаления позиции из корзины;

- product\_params – приватный метод экземпляра класса, используется для отделения параметров о позиции от других параметров, которые приходят в запросе.

**3.2.8 Класс AdminsController**

Данный класс-контроллер используется для авторизации пользователей с ролью admin. Он является главным в модуле Admin, который наследуется от ApplicationController, и отвечает за работу администрации в приложении.

Методы класса:

- admin? – публичный метод экземпляра класса, предназначен для проверки является ли текущий пользователь администратором;

- authorize\_admin – публичный метод экземпляра класса, выполняет перенаправление на главную страницу приложения, если пользователь не является администратором;

- authenticate\_user! – коллбэк, который вызывается перед всеми публичными методами класса и используется для проверки аутентификации пользователя.

**3.2.9 Класс OrdersController**

Данный класс-контроллер используется для того, чтобы администраторы имели возможность управлять заказами. Он является частью модуля Admin.

Методы класса:

- index – публичный метод экземпляра класса, используется для получения всех заказов, находящихся в приложении и отображения их администратору;

- show – публичный метод экземпляра класса, используется для получения заказа и отображения его администратору, поиск данного заказа производится по идентификатору заказа;

- new – публичный метод экземпляра класса, используется для отображения шаблона, в котором отображаются все поля, которые необходимо заполнить при создании нового заказа, после ввода отправляет информацию, которую ввел пользователь для дальнейшей обработки в метод create;

- create – публичный метод экземпляра класса, используется для создания заказа, после получения данным методом информации от пользователя, он ее обрабатывает, проверят соответствие валидаторам, заданным в модели, и, если все проверки пройдены, то сохраняет информацию, если есть какие-то ошибки возвращает их пользователю с описанием ошибки;

- edit – публичный метод экземпляра класса, используется для изменения уже существующего заказа, отображает все поля заказа на шаблоне, которые администратор может изменить и после ввода передает методу update для обработки;

- update – публичный метод экземпляра класса, используется для изменения существующего заказа, после получения данным методом информации от пользователя, он ее обрабатывает, проверят соответствие валидаторам, заданным в модели, и, если все проверки пройдены, то сохраняет информацию, если есть какие-то ошибки возвращает их пользователю с описанием ошибки;

- destroy – публичный метод экземпляра класса, используется для удаления существующего заказа. Заказ для удаления находится по идентификатору, передаваемому в параметрах при запросе;

- order\_params – приватный метод экземпляра класса, используется для отделения параметров о заказе от других параметров, которые приходят в запросе;

- make\_products\_array – приватный метод экземпляра класса, используется для создания массива из позиций;

- authorize\_admin – коллбэк, который вызывается перед всеми публичными методами класса и используется для проверки авторизации администратора.

**3.2.10 Класс ProductsController**

Данный класс-контроллер используется для того, чтобы администраторы имели возможность управлять позициями. Он является частью модуля Admin.

Методы класса:

- index – публичный метод экземпляра класса, используется для получения всех позиций, находящихся в приложении и отображения их администратору;

- show – публичный метод экземпляра класса, используется для получения позиции и отображения ее администратору, поиск данной позиции производится по идентификатору позиции;

- new – публичный метод экземпляра класса, используется для отображения шаблона, в котором отображаются все поля, которые необходимо заполнить при создании новой позиции, после ввода отправляет информацию, которую ввел администратор для дальнейшей обработки в метод create;

- create – публичный метод экземпляра класса, используется для создания позиции, после получения данным методом информации от администратора, он ее обрабатывает, проверят соответствие валидаторам, заданным в модели, и, если все проверки пройдены, то сохраняет информацию, если есть какие-то ошибки возвращает их администратору с описанием ошибки;

- edit – публичный метод экземпляра класса, используется для изменения уже существующей позиции, отображает все поля позиции на шаблоне, которые администратор может изменить и после ввода передает методу update для обработки;

- update – публичный метод экземпляра класса, используется для изменения существующей позиции, после получения данным методом информации от администратора, он ее обрабатывает, проверят соответствие валидаторам, заданным в модели, и, если все проверки пройдены, то сохраняет информацию, если есть какие-то ошибки возвращает их администратору с описанием ошибки;

- destroy – публичный метод экземпляра класса, используется для удаления существующей позиции. Позиция для удаления находится по идентификатору, передаваемому в параметрах при запросе;

- add\_to\_basket – публичный метод экземпляра класса, используется для добавления позиции в корзину;

- delete\_from\_basket – публичный метод экземпляра класса, используется для удаления позиции из корзины;

- product\_params – приватный метод экземпляра класса, используется для отделения параметров о позиции от других параметров, которые приходят в запросе;

- authorize\_admin – коллбэк, который вызывается перед всеми публичными методами класса и используется для проверки авторизации администратора.

**3.2.11 Класс RolesController**

Данный класс-контроллер используется для того, чтобы администраторы имели возможность управлять ролями. Он является частью модуля Admin.

Методы класса:

- index – публичный метод экземпляра класса, используется для получения всех ролей, находящихся в приложении и отображения их администратору;

- show – публичный метод экземпляра класса, используется для получения роли и отображения ее администратору, поиск данной роли производится по идентификатору роли;

- new – публичный метод экземпляра класса, используется для отображения шаблона, в котором отображаются все поля, которые необходимо заполнить при создании новой позиции, после ввода отправляет информацию, которую ввел администратор для дальнейшей обработки в метод create;

- create – публичный метод экземпляра класса, используется для создания роли, после получения данным методом информации от администратора, он ее обрабатывает, проверят соответствие валидаторам, заданным в модели, и, если все проверки пройдены, то сохраняет информацию, если есть какие-то ошибки возвращает их администратору с описанием ошибки;

- edit – публичный метод экземпляра класса, используется для изменения уже существующей роли, отображает все поля роли на шаблоне, которые администратор может изменить и после ввода передает методу update для обработки;

- update – публичный метод экземпляра класса, используется для изменения существующей роли, после получения данным методом информации от администратора, он ее обрабатывает, проверят соответствие валидаторам, заданным в модели, и, если все проверки пройдены, то сохраняет информацию, если есть какие-то ошибки возвращает их администратору с описанием ошибки;

- destroy – публичный метод экземпляра класса, используется для удаления существующей роли. Роль для удаления находится по идентификатору, передаваемому в параметрах при запросе;

- role\_params – приватный метод экземпляра класса, используется для отделения параметров о роли от других параметров, которые приходят в запросе;

- authorize\_admin – коллбэк, который вызывается перед всеми публичными методами класса и используется для проверки авторизации администратора.

**3.2.12 Класс UsersController**

Данный класс-контроллер используется для того, чтобы администраторы имели возможность управлять пользователями. Он является частью модуля Admin.

Методы класса:

- index – публичный метод экземпляра класса, используется для получения всех пользователей, находящихся в приложении и отображения их администратору;

- show – публичный метод экземпляра класса, используется для получения пользователя и отображения его администратору, поиск данного пользователя производится по идентификатору пользователя;

- new – публичный метод экземпляра класса, используется для отображения шаблона, в котором отображаются все поля, которые необходимо заполнить при создании нового пользователя, после ввода отправляет информацию, которую ввел администратор для дальнейшей обработки в метод create;

- create – публичный метод экземпляра класса, используется для создания пользователя, после получения данным методом информации от администратора, он ее обрабатывает, проверят соответствие валидаторам, заданным в модели, и, если все проверки пройдены, то сохраняет информацию, если есть какие-то ошибки возвращает их администратору с описанием ошибки;

- edit – публичный метод экземпляра класса, используется для изменения уже существующего пользователя, отображает все поля пользователя на шаблоне, которые администратор может изменить и после ввода передает методу update для обработки;

- update – публичный метод экземпляра класса, используется для изменения существующего пользователя, после получения данным методом информации от администратора, он ее обрабатывает, проверят соответствие валидаторам, заданным в модели, и, если все проверки пройдены, то сохраняет информацию, если есть какие-то ошибки возвращает их администратору с описанием ошибки;

- destroy – публичный метод экземпляра класса, используется для удаления существующего пользователя. Пользователь для удаления находится по идентификатору, передаваемому в параметрах при запросе;

- user\_params – приватный метод экземпляра класса, используется для отделения параметров о пользователе от других параметров, которые приходят в запросе;

- authorize\_admin – коллбэк, который вызывается перед всеми публичными методами класса и используется для проверки авторизации администратора.

**3.2.13 Класс User**

Данный класс является объектным отображением таблицы users. Данный класс используется для управления данными о пользователях в аккаунте. Данный класс связан с классом Order. Это реализовано с помощью связи один ко многим и определено в классе с помощью метода has\_many :orders.

**3.2.14 Класс Order**

Данный класс является объектным отображением таблицы orders. Данный класс используется для управления данными о заказах. Он связан с классами OrderProduct, Product, User. Связь с классом User реализована с помощью связи один ко многим и определена в классе с помощью метода belongs\_to :user. Связь с классом Product реализована с помощью связи многие-ко-многим и определена в классе с помощью метода has\_many :products и с помощью промежуточной таблицы order\_product, связь с которой осуществляется с помощью метода has\_many :order\_products.

**3.2.15 Класс OrderProduct**

Данный класс является объектным отображением таблицы order\_products. Данный класс используется для хранения данных о связи заказов и позиций. Данная таблица является промежуточной в связи многие-ко-многим между классами Product и Order и связана с ними методами belongs\_to :product и belongs\_to :order соответственно.

**3.2.16 Класс Product**

Данный класс является объектным отображением таблицы products. Данный класс используется для управления данными о позициях. Данный класс связан с классом Order. Связь реализована с помощью связи многие-ко-многим и определена в классе с помощью метода has\_many :orders и с помощью промежуточной таблицы order\_product, связь с которой осуществляется с помощью метода has\_many :order\_products. Данный класс содержит метод mount\_uploader, который указывает, какой загрузчик файлов будет использоваться в классе. В данном случае, используется PhotoUploader, что отображено в методе mount\_uploader :photo.

**3.2.17 Класс Role**

Данный класс является объектным отображением таблицы roles. Данный класс используется для управления данными о ролях. Данный класс связан с классом User. Связь реализована с помощью связи многие-ко-многим и определена в классе с помощью метода has\_and\_belongs\_to\_many :users, через промежуточную таблицу users\_roles.

# **ЗАКЛЮЧЕНИЕ**

За время прохождения преддипломной практики я подкрепил полученные во время учебы теоретические навыки практическими. Познакомился с организацией, структурой и принципами функционирования предприятия. Практически получил навыки по разработке на языке программирования Ruby, а также по пользовательскому интерфейсу JavaScript. Основной задачей являлась разработка собственного веб-приложения по продаже музыкального оборудования.

Для написания данного веб-приложения использовался язык программирования Ruby. Данный язык был выбран ввиду минимизации затрат при использовании, быстрого написания кода, а также простоты разработки.

В качестве пользовательского интерфейса был практически изучен JavaScript, а также jQuery. JavaScript показался для меня простым в изучении, широко используемым, а также мощным языком для управления поведением веб-страницы.

Используемый в работе над проектом фреймворк Ruby on Rails – фреймворк, написанный на языке программирования Ruby с открытым исходным кодом, реализует архитектурный шаблон для веб-приложений. Ruby on Rails представляет собой архитектуру MVC для веб-приложений, а также обеспечивает их интеграцию с веб-сервером и сервером базы данных. MVC – это не метод организации приложения, а его структура. Суть заключается в том, чтобы отделить бизнес-логику приложения от его представления.

Написание отчета по разработке данного приложения разделено по блокам для удобства изучения и описания.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Сайт продажи музыкального оборудования [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.thomann.de/intl/by/index.html. – Дата доступа: 24.03.2022.

[2] Сайт продажи музыкального оборудования [Электронный ресурс]. – Электронные данные. – Режим доступа: http://guitarland.by/. – Дата доступа: 24.03.2022.

[3] Ruby [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ru.wikipedia.org/wiki/Ruby. – Дата доступа: 27.03.22.

[4] Язык программирования Руби. / D. Flanagan Y. Matsumoto – СПб. Издательство Питер, 2011. – 496с.

[5] JavaScript [Электронный ресурс]. – Электронные данные. – Режим доступа: https://developer.mozilla.org/ru/docs/Learn/JavaScript. – Дата доступа: 28.03.2022.

[6] Официальный сайт JQuery [Электронный ресурс]. – Электронные данные. – Режим доступа: https://jquery.com/. – Дата доступа: 01.04.2022.

[7] Ruby on Rails [Электронный ресурс]. – Электронные данные. – Режим доступа: https://web-creator.ru/articles/why\_ruby\_on\_rails. – Дата доступа: 01.04.22.

[8] Шаблон MVC [Электронный ресурс]. – Электронные данные. – Режим доступа: https://web-creator.ru/articles/mvc. – Дата доступа: 01.04.22.

[9] PostgreSQL [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.postgresql.org/. – Дата доступа: 02.04.2022.

# **ПРИЛОЖЕНИЕ А**

(обязательное)

Вводный плакат

# **ПРИЛОЖЕНИЕ Б**

(обязательное)

Веб-приложение по продаже музыкального оборудования

Схема структурная

# **ПРИЛОЖЕНИЕ В**

(обязательное)

Веб-приложение по продаже музыкального оборудования

Диаграмма классов