**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.