# 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`.