

Министерство образования Республики Беларусь  
Учреждение образования «Белорусский государственный университет  
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Базы данных (БД)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему:

«Веб-приложение «Каталог инструкций»

БГУИР КП 1-40 01 01 05 012 ПЗ

Студент: гр. 551005 Коваленко И.А.

Руководитель: асс. Марина И.М.

Минск 2018

Учреждение образования  
«Белорусский государственный университет информатики и  
радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

(подпись)

Лапицкая Н.В. 2018 г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Коваленко Илье Андреевичу

1. Тема работы Веб-приложение «Каталог инструкций»
2. Срок сдачи студентом законченной работы 15.12.2018
3. Исходные данные к работе: требования к разрабатываемому программному средству.
4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение.

1. Аналитический обзор литературы и существующих аналогов;
2. Обоснование выбора программных средств;
3. Разработка программного средства;

4. Обоснование технических приемов программирования;

5. Тестирование программы;

6. Руководство пользователя программы;

7. Заключение, список литературы, ведомость, приложения.

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

6. Консультант по курсовому проекту Марина И.М.

7. Дата выдачи задания 15.10.2018 г.

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объема работы):

раздел 1, введение к 30.09.2018 – 10 % готовности работы;

разделы 2 к 15.10.2018 – 30 % готовности работы;

разделы 3,4 к 10.11.2018 – 60 % готовности работы;

раздел 5, 6 к 30.11.2018 – 90 % готовности работы;

оформление пояснительной записки и графического материала к 10.12.2018 – 100 % готовности работы.

Защита курсового проекта с 20.11 по 15.12.2018 г.

РУКОВОДИТЕЛЬ \_\_\_\_\_ И.М. Марина

(подпись)

Задание принял к исполнению

\_\_\_\_\_ 15.09.2018 г.

(дата и подпись студента)

## СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ ТЕРМИНОВ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ .....	6
ВВЕДЕНИЕ .....	6
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ.....	8
1.1 Инструкции и основные сведения о них .....	8
1.2 Анализ уже созданных программных средств .....	9
1.3 Постановка требований к разрабатываемому программному средству .....	12
2 ВЫБОР ПРОГРАММНЫХ СРЕДСТВ .....	13
3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА .....	15
3.1 Проектирование архитектуры.....	15
3.2 Выбор компонентов приложения .....	19
3.3 Обоснование технических приемов программирования .....	23
4 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА .....	25
5 ТЕСТИРОВАНИЕ.....	43
6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ .....	48
ЗАКЛЮЧЕНИЕ.....	64
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	66
ПРИЛОЖЕНИЕ А – Исходный текст программы.....	67



## **ПЕРЕЧЕНЬ ТЕРМИНОВ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ**

ОС - Операционная система;

XML -расширяемый язык разметки;

Аjах - асинхронный JavaScript и XML, концепция создания  
пользовательского интерфейса web-приложений

## ВВЕДЕНИЕ

Веб-приложение — клиент-серверное приложение, в котором клиент взаимодействует с сервером при помощи браузера, а за сервер отвечает — веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются межплатформенными службами. [1]

В настоящее время набирает популярность новый подход к разработке веб-приложений, называемый Ajax. При использовании Ajax страницы веб-приложения не перезагружаются целиком, а лишь догружают необходимые данные с сервера, что делает их более интерактивными и производительными.

Также, в последнее время набирает большую популярность технология WebSocket, которая не требует постоянных запросов от клиента к серверу, а создает двунаправленное соединение, при котором сервер может отправлять данные клиенту без запроса от последнего. Таким образом появляется возможность динамически управлять контентом в режиме реального времени.

Для создания веб-приложений на стороне сервера используются разнообразные технологии и любые языки программирования, способные осуществлять вывод в стандартную консоль.

В ходе курсового проектирования планируется создать веб-приложение каталог инструкций, которое позволит пользователям работать с инструкциями. Планируемые функции: создание, просмотр, редактирование, удаление инструкций, возможность комментировать инструкции, устанавливать рейтинг для каждой инструкции, полнотекстовой поиск по всем инструкциям, регистрация и авторизация через социальные сети, просмотр прогресса инструкций для авторизованных пользователей, выбор языка приложения (русский и английский), возможность управления профилем пользователя.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Инструкции и основные сведения о них

Инструкция — документ, содержащий правила, указания или руководства, устанавливающие порядок и способ выполнения или осуществления чего-либо. [2]

С другой стороны инструкция — это правовой акт, утверждаемый или издаваемый в целях установления правил, регулирующих организационные, научно-технические, технологические, финансовые или иные специальные стороны деятельности учреждений, организаций, предприятий (их структурных подразделений и служб), должностных лиц и граждан.

В управленческой практике используются как типовые, так и индивидуальные инструкции. Типовые обычно издают органы власти или управления для системы однотипных организаций или учреждений.

Заголовок должен четко определять те вопросы, объекты или круг лиц, на которые распространяется инструкция. Например, должностная инструкция «Секретаря-референта», Инструкция «По делопроизводству».

Как правило, первый раздел инструкции называется «Общие положения». В нем указывают цели издания, область распространения, порядок и обязательность использования, правовые, нормативные или распорядительные акты, послужившие основой для разработки инструкции.

Текст инструкции состоит из разделов, подразделов, пунктов и подпунктов, которые нумеруются арабскими цифрами. Количество разделов и их логическая взаимосвязь определяются разработчиками. По стилю изложения текст инструкции носит распорядительный характер («устанавливается», «не допускается», «запрещается» и др.) и излагается от третьего лица единственного или множественного числа.

Одним из видов инструкций является инструкция по эксплуатации — описание изделия и правил пользования им. Большинство производителей включают буклеты с инструкцией по эксплуатации в комплект доставки. Такие буклеты содержат описание частей изделия, если необходимо, последовательность его сборки, рекомендации по настройке, пользованию и обслуживанию. Эти описания снабжаются иллюстрациями, схемами и



чертежами. Особое внимание в правилах по эксплуатации уделяется правилам безопасности.

Для изделий, экспортируемых в разные страны, инструкции по эксплуатации часто выполняются на нескольких языках. Инструкции по эксплуатации также могут наноситься непосредственно на изделия, например в виде наклеек или надписей краской. Некоторые компании также помещают инструкции по эксплуатации на своих интернет-сайтах.[3]

## **1.2 Анализ уже созданных программных средств**

Из англоязычных ресурсов наиболее популярными и широко используемыми являются веб-сайты wikiHow и Instructables.

Instructables – веб-сервис, специализирующийся на пользовательских «Сделай сам» инструкциях. Пользователи размещают пошаговые инструкции, сопровождаемые различным визуальным контентом, а затем взаимодействуют между собой при помощи комментариев или тематического форума.

wikiHow – вики-проект и интернет-сообщество, состоящее из обширной базы данных практических руководств и инструкций. Веб-сайт предоставляет контент с руководствами в Интернете, позволяя всем желающим редактировать страницы. Большинство практических статей следуют схожему формату с шагами, советами, предупреждениями, списком вещей, которые могут понадобиться, и дополняются изображениями, чтобы помочь читателю узнать, как выполнить задачу.

Особенности работы сервиса Instructables:

Одна из ключевых особенностей сервиса – это создание пошаговых инструкций, которые состоят из текста и интерактивных элементов (изображения и видеоматериалы). Присутствует возможность выставить оценки инструкциям. Для того, чтобы была возможность пользоваться инструкциями без интернета в сервисе существует возможность экспорта инструкций в формат pdf.

Авторизация и регистрация на сервисе возможна как через внутреннюю систему, так и через социальные сети.

Особенности работы сервиса wikiHow:

wikiHow использует вики-метод непрерывного улучшения, позволяя

редакторам добавлять, удалять или иным образом изменять контент. Когда статья создаётся, члены сообщества наблюдают за ней, чтобы улучшить её качество. Каждое редактирование проверяется во время процесса под названием «Патрулирование последних изменений», где добровольцы просматривают контент в соответствии с внутренними стандартами, исключая некачественные правки, такие как вандализм и тестовые правки, и сохраняя улучшения. Центральной в приложении является панель инструментов сообщества, которая отображает несколько динамических апплетов, которые связывают пользователей с различными инструментами редактирования, такими как программа проверки орфографии, программа автоматического подбора категории, с помощью которого статьям присваиваются категории в соответствии с их темой, и временное хранилище неопубликованных статей, куда копируются некачественные статьи, где их редактируют и переписывают по стилю и формату.

Большое внимание уделено обработке новых статей и их подготовке перед тем, как их увидит гость ресурса. По умолчанию вновь созданные статьи удаляются из индекса поисковых систем; текст статьи размыт и отображается уведомление о том, что страница невидима для читателей, но зарегистрированные пользователи могут отклонить предупреждение, чтобы просмотреть содержимое статьи.

Система под названием «Продвижение новых статей» позволяет некоторым пользователям с правом «Помощник по новым статьям» просматривать эти статьи и при необходимости доводить их до стандартов сервиса по оформлению статей. Статьи, соответствующие этим стандартам, продвигаются, что устраняет эффект размытия, а также убирает уведомление, и делает статью общедоступной и индексируемой для поиска. Если помощник по новым статьям найдёт недавно написанную высококачественную статью, он может отметить её как «Восходящая звезда». По умолчанию публикации, отмеченные как «Восходящая звезда», продвигаются и помещаются на главную страницу в течение неопределённого периода времени. Противоположно этому, статьи ниже стандартов и которые требуют слишком много работы, чтобы «быть готовы для читателей», «понижены в статусе», который удаляет их из списка продвижения новых статей и сохраняет размытие и уведомление и перемещает страницу в категорию «Статьи для проверки

качества». Статьи с контентом, которые противоречат политике удаления сайта, также понижены в статусе (например, статьи, основанные на шутках, противоречащих законодательству темах, а также строго неточные или неполные инструкции).[4]

Авторизация и регистрация, аналогично Instructables, возможна как через внутреннюю систему, так и через социальные сети. В настоящее время, когда почти у каждого пользователя интернета существует аккаунт в социальных сетях, это является важной функциональностью.

Основной функционал, требуемый от подобного веб-сайта представлен на схеме 1.1.

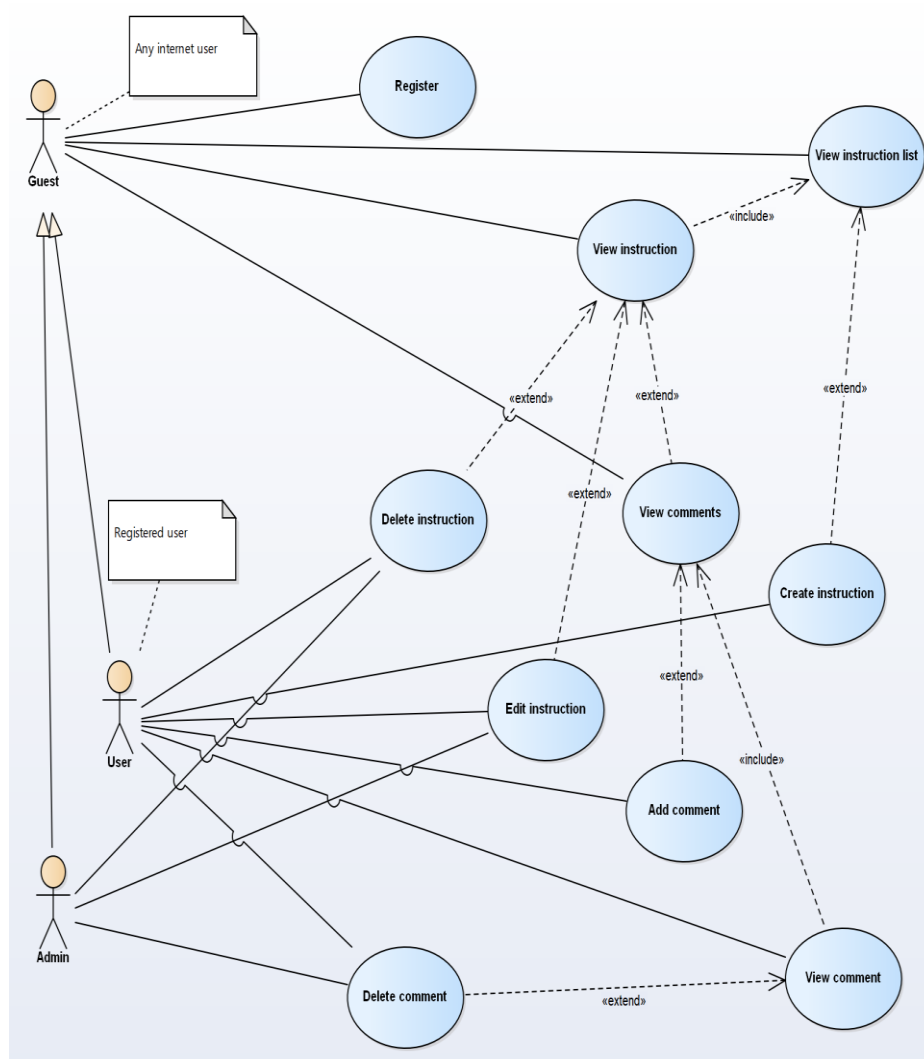


Рисунок 1.1 – Диаграмма использования

### **1.3 Постановка требований к разрабатываемому программному средству**

Создать веб-приложение, позволяющее управлять инструкциями.

Планируемые функции:

- создание, удаление, редактирование, просмотр прогресса инструкций;
- редактирование инструкций и пользователей при помощи форм с автоматическим сохранением данных;
- авторизация через социальные сети;
- поддержка профилей пользователей;
- поддержка двух языков: русского и английского;
- возможность проставлять рейтинг инструкциям;
- оставлять комментарии (с транслированием их в реальном времени всем клиентам, подключенным к странице с инструкцией);
- возможность полнотекстового поиска по сайту.

## 2 ВЫБОР ПРОГРАММНЫХ СРЕДСТВ

В качестве фреймворка для создания веб-приложения был сделан выбор в пользу Ruby On Rails. Rails, написанный на языке программирования Ruby, реализует архитектурный шаблон Model-View-Controller для веб-приложений, а также обеспечивает их интеграцию с веб-сервером и сервером баз данных. Является открытым программным обеспечением и распространяется под лицензией MIT. Фреймворк был выбран потому, что он является передовым в области веб-разработки и позволяет быстро и выразительно решать поставленные задачи. Ruby on Rails использует REST-стиль построения веб-приложений.

Rails базируется на следующих принципах разработки приложений:

- использование механизмов повторного использования, позволяющих минимизировать дублирование кода в приложениях (принцип Don't repeat yourself);

- по умолчанию используются соглашения по конфигурации, типичные для большинства приложений (принцип Convention over configuration) — явная спецификация конфигурации требуется только в нестандартных случаях.

Для разработки была выбрана операционная система macOS, так как UNIX-подобные операционные системы (которой является macOS) отлично подходят для веб-разработки, потому что в подавляющем большинстве серверной операционной системой выбирают одну из бесплатных UNIX-подобных. Поэтому, используя операционную систему с аналогичной архитектурой при разработке и при развертывании приложения, можно избежать ошибок, связанных с совместимостью операционных систем.

В качестве системы контроля версий был выбран Git, ввиду его высокой популярности и распространенности. В качестве хостинга репозитория был сделан выбор в пользу Github, из-за наличия большого (по сравнению с аналогичными сервисами) open-source сообщества.

Для обеспечения высокого качества программного кода часто используются статические анализаторы кода, позволяющие выявлять и исправлять наиболее типовые ошибки в ПО. Для Ruby наилучшим выбором будет использовать статический анализатор Rubocop, который из коробки поддерживает все основные рекомендуемые стандарты оформления кода на

языке программирования Ruby, а также имеет гибкую систему конфигурации, позволяющую добавлять в исключения некоторые правила, которые по мнению разработчика являются не особо важными для проекта. Одна из часто допустимых ошибок в rails приложениях является ошибка N+1, поэтому для автоматизации поиска и своевременного оповещения о данной ошибке будет использован инструмент `gem bullet`.

В качестве среды разработки был выбран редактор Atom с установленными дополнениями, так как он имеет глубокую интеграцию с Git, поддержку статического анализатора кода Rubocop, подсветку синтаксиса Ruby, Rails.

Для тестирования работы приложения был использован браузер Safari, который является стандартным для используемой операционной системы macOS. Браузер выбран ввиду наличия удобных средств для веб-разработки, таких как: консоль разработчика, возможность смены пользовательского агента для тестирования поведения приложения в разных браузерах, встроенные возможности для тестирования адаптивного дизайна (выбор устройства, на котором необходимо провести тестирование, выбор разрешения, ориентации экрана). Для тестирования работы приложения на мобильных устройствах был использован Android смартфон с установленным браузером Mozilla Firefox, из-за его высокой популярности и открытости (на основе него создано целое множество других браузеров).

Хостингом для разрабатываемого приложения был выбрана облачная платформа Heroku, которая в бесплатном варианте поддерживает размещение Rails приложений и предлагает ряд вспомогательных бесплатных сервисов, таких как база данных MySQL и Redis.

## 3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

### 3.1 Проектирование архитектуры

Основные действия пользователя можно продемонстрировать при помощи диаграммы деятельности, представленной на схеме 3.1.

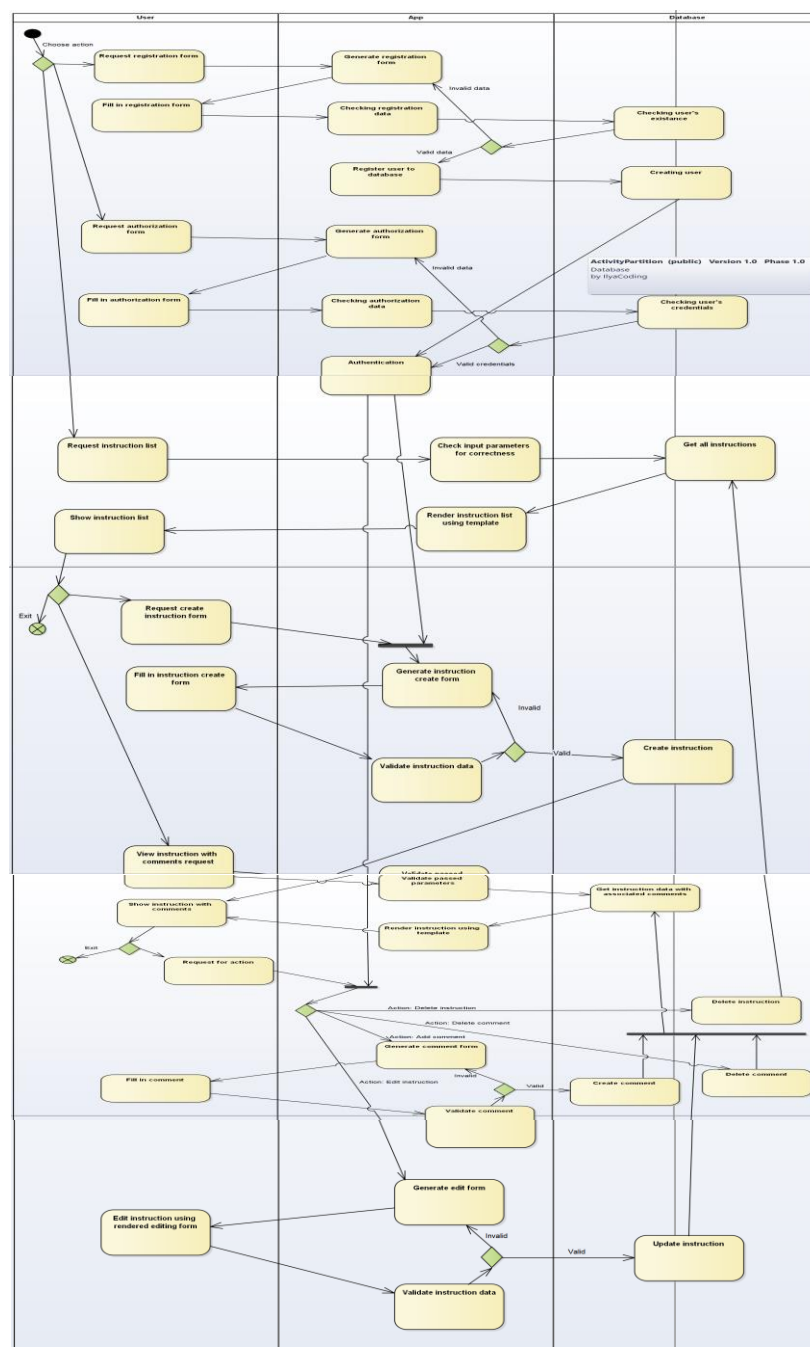


Рисунок 3.1 – Диаграмма деятельности

Продemonстрировать как объект переходит из одного состояния в другое можно при помощи диаграммы состояний, представленной на схеме 3.2.

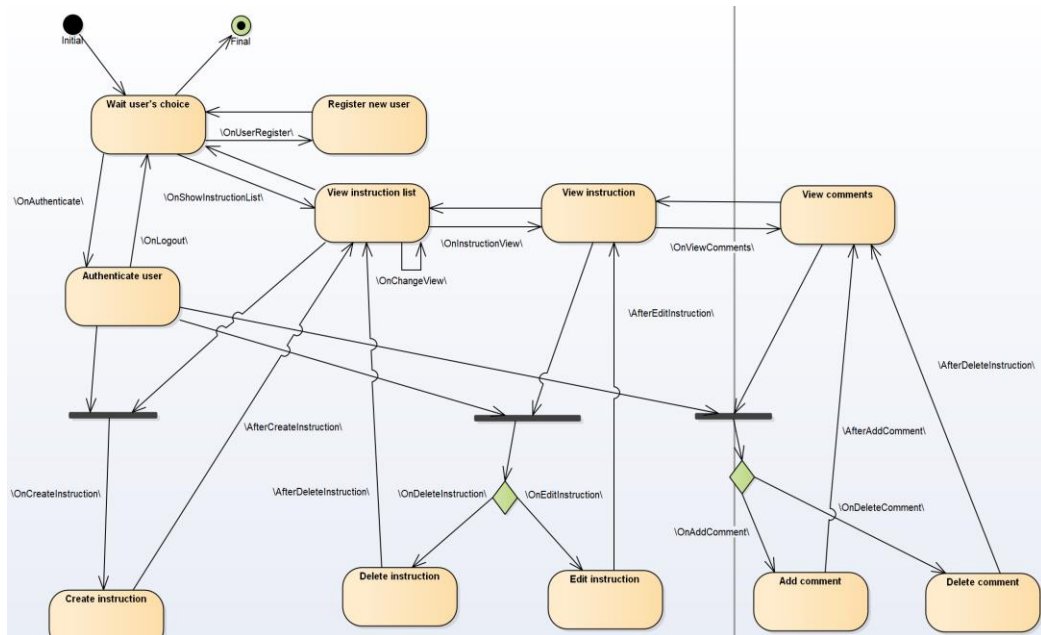


Рисунок 3.2 – Диаграмма состояний

Для того, чтобы показать процесс регистрации во времени можно воспользоваться диаграммой последовательности 3.3.

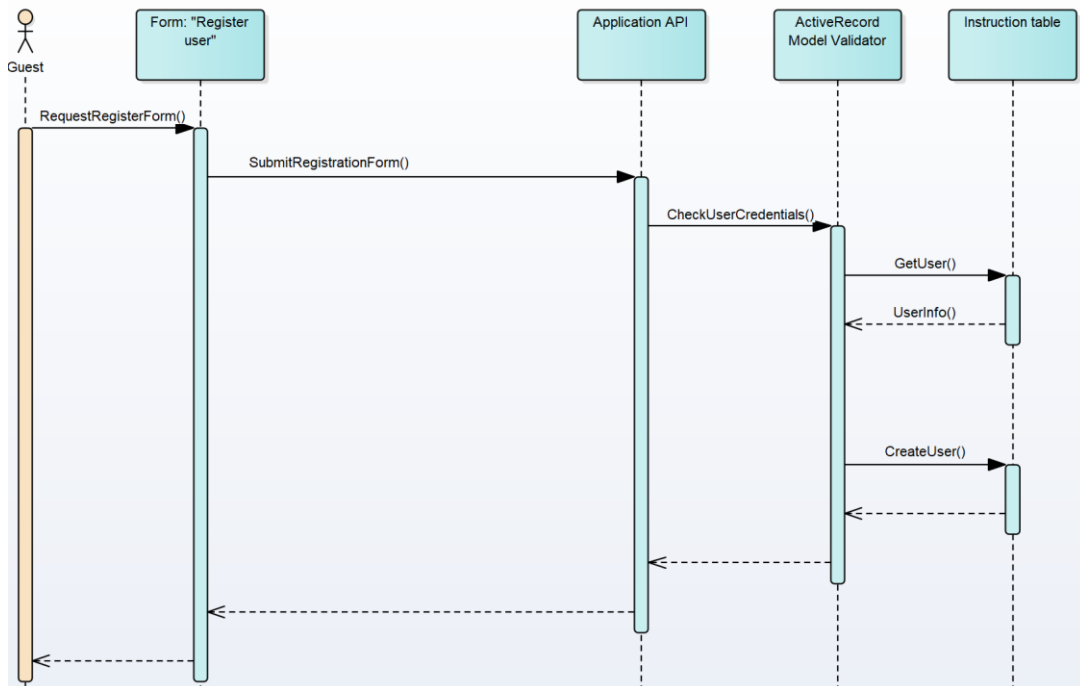


Рисунок 3.3 – Диаграмма последовательности регистрации пользователя



Поиск по сайту было решено сделать как можно более дружелюбным для пользователя согласно блок-схеме, представленной на рисунке 3.4.



Рисунок 3.4 – Схема алгоритма поиска

Процесс того, как будет происходить авторизации пользователей и администраторов в программном средстве, показан на рисунке 3.5.

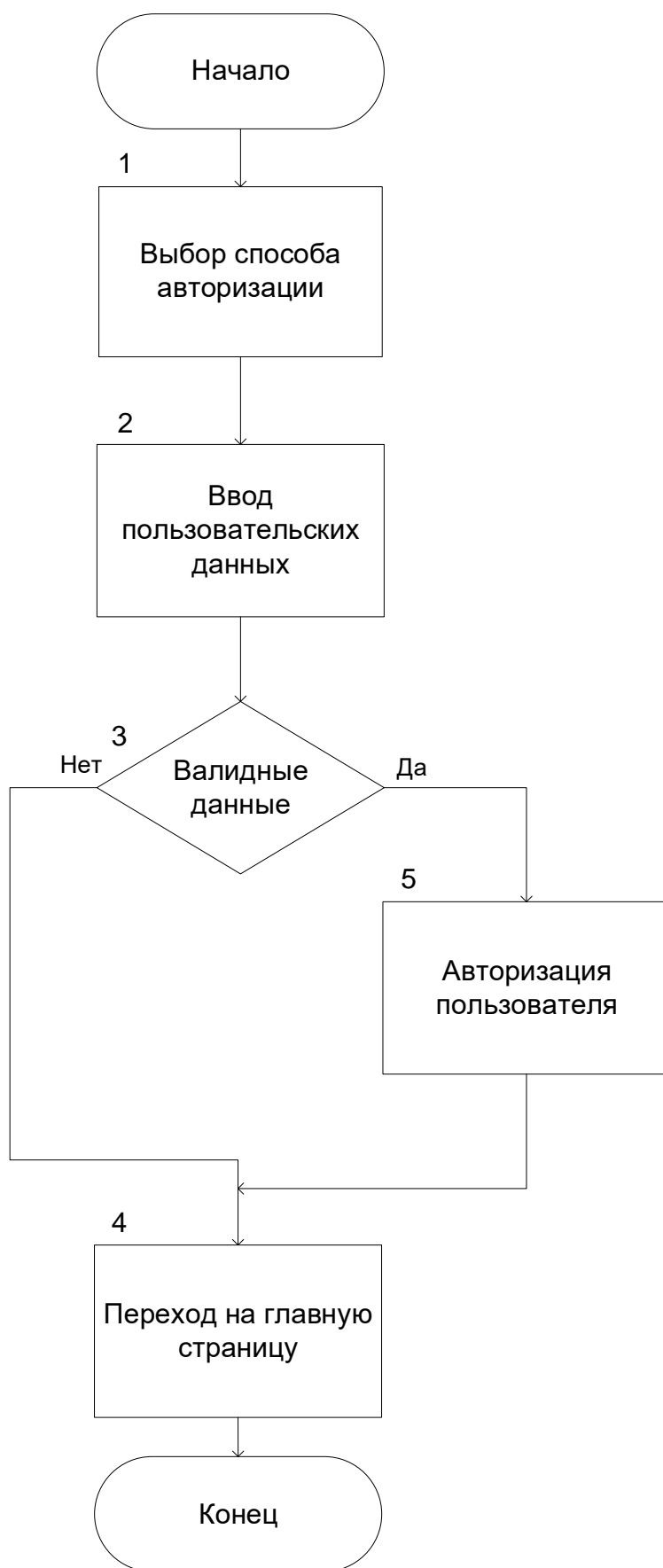


Рисунок 3.5 – Схема алгоритма авторизации пользователя

Алгоритм работы пользователя с отдельно взятой инструкцией представлен на рисунке 3.6.

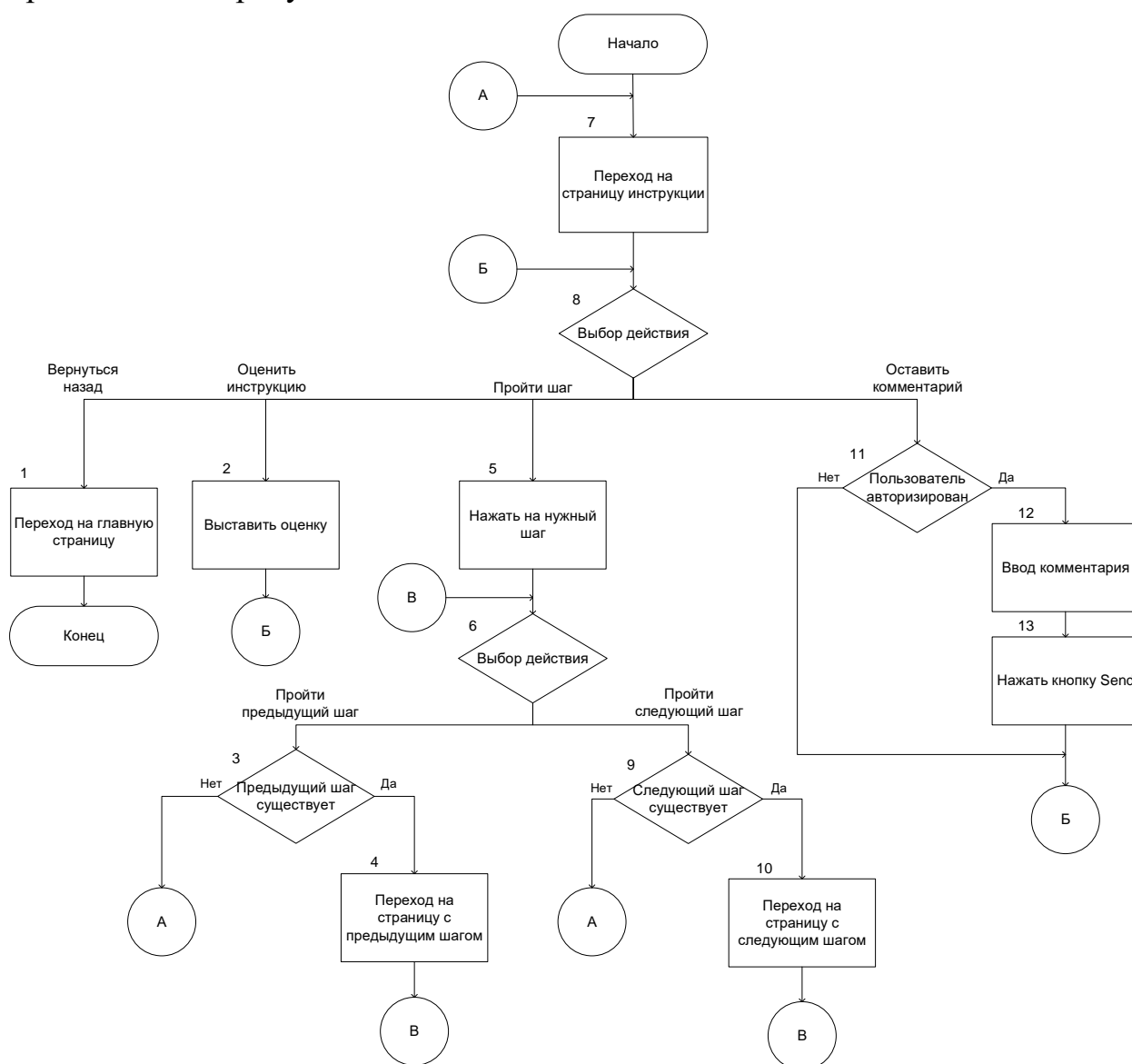


Рисунок 3.6 – Схема алгоритма работы пользователя с инструкцией

### 3.2 Выбор компонентов приложения

При разработке на языке программирования Руби принято придерживаться компонентного подхода. Компонентом в терминах ruby называют gem. Для управления пакетами стандартно в Rails используется RubyGems (англ. gem, gems— драгоценный камень) — система управления пакетами для языка программирования Руби, который предоставляет стандартный формат для программ и библиотек Руби (в самодостаточном

формате), инструменты, предназначенные для простого управления установкой «gems», и сервер для их распространения.

Введем определение понятия файлопровода Rails. Файлопровод представляет фреймворк для соединения и минимизации или сжатия файлов JavaScript и CSS. Он также добавляет возможность писать эти файлы на других языках и препроцессорах, таких как CoffeeScript, Sass и ERB. Это позволяет комбинировать ассеты вашего приложения с ассетами других гемов. Например, jquery-rails содержит копию jquery.js и включает особенности AJAX в Rails.

Для разработки было решено использовать следующие геммы:

1 twitter-bootstrap-rails – gem необходимый для интеграции фреймворка bootstrap в файлопровод Rails приложения. Bootstrap – это набор инструментов от Twitter предназначенных для начала разработки веб-приложений и сайтов. Он включает базовые CSS и HTML для типографии, форм, кнопок, таблиц, сеток и навигации.[5]

2 jquery-rails – gem добавляющий поддержку библиотеки JQuery в веб-приложение.

3 jquery-ui-rails – gem добавляющий поддержку визуальных эффектов на основе библиотеки JQuery UI. Пакет позволяет подключать только используемые эффекты или анимации, что существенно уменьшает выходной JavaScript файл и, следовательно, уменьшает время загрузки страницы для пользователей сайта.

4 jquery-infinite-pages – пакет содержащий JQuery плагин для добавления бесконечной прокрутки страниц.

5 rails-jquery-autocomplete – пакет добавляющий поддержку автодополнения на основе JQuery.

6 angularjs-rails – пакет добавляющий поддержку AngularJS в Rails приложение. По-умолчанию поддерживается минификация (то есть сжатие файлов в файлопроводе в production окружении).

7 rails-angular-ui-sortable – пакет добавляющий поддержку директивы UI.Sortable фреймворка JQuery для поддержки автоматического дополнения вводимой информации.

8 bootstrap-tagsinput-rails – пакет содержащий плагин bootstrap-tagsinput для фреймворка JQuery, позволяющий управлять тегами в элементе формы

input.

9 redcarpet – рубли библиотека для обработки текста в формате Markdown. Она будет использована для оформления текста инструкций. По сути является оберткой над собственной библиотекой операционной системы.[6]

10 local\_time – пакет, необходимый для отображения локального времени для пользователей.

11 acts-as-taggable-on – пакет добавляющий поддержку тегов. Будет использован для добавления тегов к инструкциям, что позволит пользователям лучше взаимодействовать с ресурсом.

12 kaminari – пакет добавляющий поддержку постраничной навигации для различных сущностей. Поддерживает большую часть Ruby фреймворков. Библиотека написана без добавления или переопределения стандартных методов в глобальные классы Руби, такие как Array, Hash, Object, ActiveRecord::Base.

13 devise – гибкое решение аутентификации для Rails на основе Warden. Основные его преимущества: основа на Rack (высокая совместимость с Ruby приложениями), целостное MVC решение для аутентификации, позволяет иметь несколько моделей аутентифицированных одновременно, полностью состоит из модулей и позволяет отключать ненужные или подключать необходимые модули. Состоит из 10 модулей.[7]

14 rolify – пакет позволяющий добавить управление ролями в веб-приложение. Gem имеет функциональность проверки ролей для пользователей и не реализует элементы авторизации.

15 cancancan – пакет для авторизации для Rails. Позволяет задать роли в одном файле ability и не дублировать их в моделях, контроллерах и представлениях, оставляя их код чистым, храня логику управления правами в одном месте. Состоит из двух частей: библиотека авторизации, которая определяет права доступа пользователя к определенным объектам и вспомогательные (так называемые helper) методы для контроллеров, которые помогают держать в контроллере минимум кода.[8]

16 devise-bootstrap-views – пакет, добавляющий возможность генерировать представления для форм авторизации с использованием фреймворка Bootstrap. Это необходимо для создания современного адаптивного дизайна.

17 `omniauth` – библиотека, которая стандартизует поддержку аутентификации пользователей от многих поставщиков. Разработчик определяет стратегии авторизации для различных поставщиков. В настоящий момент реализованы стратегии авторизации в виде пакетов для всех популярных поставщиков аутентификации. Для использования библиотеки необходимо определить одну или более стратегий авторизации. Стратегии поставляются отдельно при помощи `RubyGems` менеджера пакетов. Для более удобной разработки и отладки приложения определена стандартная стратегия `Developer`. Она не рекомендуется для использования в `production` окружении, а создана как заглушка для дальнейшей замены ее на реальные стратегии.[9]

18 `omniauth-facebook` – пакет реализующий стратегию авторизации при помощи социальной сети Facebook.

19 `omniauth-twitter` – пакет реализующий стратегию авторизации при помощи социальной сети Twitter.

20 `omniauth-google-oauth2` – пакет реализующий стратегию авторизации при помощи социальной сети Google.

21 `omniauth-vk` – пакет реализующий стратегию авторизации при помощи социальной сети ВКонтакте.

22 `ratyrate` – пакет, являющийся оберткой над библиотекой `JQuery Raty`. Необходим для добавления возможности пользователям оценить статью при помощи удобного интерфейса. Позволяет добавить кнопку отмены для рейтинга, добавить возможность показа половинок звезд, возможность задать путь к своим картинкам для кастомизации оформления функционала голосования за инструкцию.[10]

23 `gravastic` – пакет, добавляющий возможность получить аватарку (картинку для профиля пользователя) при помощи сервиса `gravatar`. При авторизации через социальные сети приложение будет запрашивать email. Затем, на основе данного email будет запрашиваться соответствующая аватарка.[11]

24 `russian` – пакет, добавляющий поддержку русского языка в стандартную поставку `Rails`.

25 `search_cor` – пакет, расширяющий модель `ActiveRecord` для предоставления функционала полнотекстового поиска по всему содержимому веб-приложения. Позволяет включать в результаты поиска поиск по

вложенным сущностям, давая более точные результаты поиска для пользователя.[12]

### 3.3 Обоснование технических приемов программирования

В ходе проектирования приложения были определены следующие основные сущности: инструкция (состоит из шагов), шаг (состоит из блоков), блоком может быть текст в формате markdown, картинка или видео. Для хранения блоков в базе данных был выбран паттерн STI (Single Table Inheritance) — паттерн проектирования, который позволяет перенести объектно-ориентированное наследование на таблицу реляционной базы данных. В таблице БД должно присутствовать поле идентифицирующее название класса в иерархии. Зачастую, в том числе в Ruby On Rails, поле называют type. Таким образом, мы можем иметь одну таблицу и несколько типов объектов (моделей), которые будут в ней храниться.[13]

В приложении категории содержат инструкции, у пользователей есть много инструкций, много комментариев. Чтобы подсчитать количество комментариев и статей для пользователя нужно выполнить дополнительные запросы. При помощи Rails можно создать специальное кеширующее поле, которое позволит нам не вычислять значение счетчика динамически каждый раз при обращении к профилю пользователя, а брать как есть из дополнительной колонки в таблице пользователей. При этом Rails обеспечивает корректность данного счетчика.

Было решено реализовать отправку комментариев к статьям всем клиентам в реальном времени. Для коммуникаций в реальном времени лучшим решением является использовать веб-сокеты (WebSockets). В Rails имеется встроенная поддержка веб-сокетов Action Cable. Он позволяет писать функционал реального времени на Ruby в стиле и формате остальной части приложения Rails, в то же время являясь производительным и масштабируемым. Он представляет полный стек, включая клиентский фреймворк на JavaScript и серверный фреймворк на Ruby. Вы получаете доступ к моделям, написанным с помощью Active Record или другой ORM.

Publish-Subscribe, относится к парадигме очереди сообщений, когда отправители информации (publishers) посылают данные в абстрактный класс

получателей (subscribers), без указания отдельных получателей. Action Cable использует этот подход для коммуникации между сервером и множеством клиентов.

Соединения (connection) формируют основу взаимоотношения клиента с сервером. Для каждого WebSocket, принимаемого сервером, на стороне сервера будет инициализирован объект соединения. Этот объект становится родителем для всех подписок на канал, которые создаются впоследствии. Само соединение не работает с какой-либо определенной логикой приложения после аутентификации и авторизации. Клиент соединения WebSocket называется потребителем соединения (consumer). Отдельный пользователь создаст одну пару потребитель-соединение на каждую вкладку браузера, окно или устройство, которые он использует.

Для легкости управления файлами JavaScript и CSS было решено использовать файлопровод Rails для компиляции ассетов в production окружении.

Первой особенностью файлопровода является соединение ассетов, что может уменьшить количество запросов, необходимых браузеру для отображения страницы. Браузеры ограничены в количестве запросов, которые они могут выполнить параллельно, поэтому меньшее количество запросов может означать более быструю загрузку вашего приложения.

Sprockets соединяет все JavaScript файлы в один главный файл .js и все CSS файлы в один главный файл .css. Можно настроить эту стратегию, сгруппировав файлы любым способом. В production, Rails вставляет метку SHA256 в каждое имя файла, таким образом файл кэшируется браузером. Кэш можно сделать недействительным, изменив эту метку, что происходит автоматически каждый раз, когда изменяется содержимое файла.

Второй особенностью файлопровода является минимизация или сжатие ассетов. Для файлов CSS это выполняется путем удаления пробелов и комментариев. Для JavaScript могут быть применены более сложные процессы. Можно выбирать из набора встроенных опций или определить свои.

Третьей особенностью файлопровода является то, что он позволяет писать ассеты на языке более высокого уровня с дальнейшей компиляцией до фактического ассета. Поддерживаемые языки по умолчанию включают Sass для CSS, CoffeeScript для JavaScript и ERB для обоих.[14]



## 4 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

Исходя из темы курсового, основным ресурсом веб-приложения являются инструкции. Приложение по структуре является Model-View-Controller (MVC, «Модель-Представление-Контроллер») приложением, поэтому первым делом было принято решение описать используемые классы модели, так как они являются основой приложения, от которой зависят все остальные компоненты разрабатываемого приложения.

Так как модель предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность, было решено привести структуру таблицы баз данных, а также методы, объявленные напрямую на модели представленной в приложении. Модель не зависит от представления — не знает как данные визуализировать — и контроллера — не имеет точек взаимодействия с пользователем — просто предоставляя доступ к данным и управлению ими. Модель предоставляет данные и реагирует на команды контроллера, изменяя свое состояние.

Представление отвечает за отображение данных модели пользователю, реагируя на изменения модели.

Контроллер интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Таблица 4.1 – Классы моделей и их краткое описание

Название класса	Краткое описание
Ability	Необходим для пакета CanCanCan. В данном классе задаются все права для всех доступных групп пользователей (включая гостей). Задание всех прав в одном классе позволяет не загружать другие модели данным функционалом.
ApplicationRecord	Абстрактный класс для всех контроллеров-обработчиков веб-приложения.

Продолжение таблицы 4.1

Название класса	Краткое описание
AverageCache	Используется пакетом ratyrate для обеспечения кешированием голосов пользователя.
Block	Абстрактный элемент, из наследников которого будет строится инструкция. Является наименьшим элементом инструкций или же является частью шага.
Category	Категории необходимы для объединения инструкций по определенной тематике.
Comment	Сущность, обозначающая комментарий авторизованного пользователя, который он может оставить к любым инструкциям в приложении.
CompletedStep	Сущность, обозначающая шаг инструкции, который пользователь прошел в ходе пошагового выполнения инструкции.
Image < Block	Обозначает картинку, которая может быть одним из блоков инструкции. Наследуется от абстрактной сущности Block.
Manual	Обозначает инструкцию – основная сущность приложения
OverallAverage	Сущность обозначающая кеш для использования в ratyrate пакете для простановки рейтинга к инструкциям.

Продолжение таблицы 4.1

Название класса	Краткое описание
Rate	Внутренняя сущность, используемая пакетом ratyrate.
RatingCache	Внутренняя сущность, используемая пакетом ratyrate.
Role	Сущность, обозначающая роль пользователя. Связана по связи многие-ко-многим с таблицей пользователей, чтобы дать возможность пользователям иметь много ролей. Используется пакетом rolify.
Step	Сущность, обозначающая шаг инструкции. Является одним из компонентов инструкции. Содержит вложенные блоки (наследники класса Block), которые и формируют содержимое шага.
Text < Block	Обозначает текст в формате Markdown, который может быть одним из блоков инструкции. Наследуется от абстрактной сущности Block.
User	Сущность, обозначающая пользователя в приложении. Используется пакетом devise, а также пакетом omniauth и его стратегиями. Играет ключевую роль в безопасности приложения.
Video < Block	Обозначает youtube видео, которое может быть одним из блоков инструкции. Наследуется от абстрактной сущности Block.

Таблица 4.2 – Описание методов и свойств класса Ability

Название элемента класса	Краткое описание
Конструктор initialize(user)	Конструктор класса, принимающий в качестве параметра пользователя, для которого будет производиться проверка прав доступа.
Подключаемый модуль CanCan::Ability	Подключаемый модуль поставляемого пакета CanCanCan для добавления в класс методов, связанных с авторизацией.

Таблица 4.3 – Описание методов и свойств класса ApplicationRecord

Название элемента класса	Описание
Поле abstract_class	Объявляет абстрактный класс.

Таблица 4.4 – Описание свойств и методов класса Block

Название элемента класса	Описание
Поле id	Первичный ключ для сущности.
Поле content	Содержимое блока, которое может быть текстом, ссылкой на картинку или ссылкой на видео на видеохостинге YouTube.
Поле priority	Устанавливает номер блока в списке для конкретного шага. Необходимо для того, чтобы пользователь мог изменить порядок следования блоков в инструкции.
Поле type	Поле, идентифицирующее название класса в иерархии наследования. Необходимо для реализации STI в Rails.
Поле step_id	Идентификатор шага, к которому принадлежит блок.
Поле created_at	Время создания записи.

Продолжение таблицы 4.4

Название элемента класса	Описание
Поле <code>updated_at</code>	Время последнего обновления записи.
<code>resourcify</code>	Метод, проверяющий права для данной сущности при помощи пакета <code>CanCanCan</code> .
<code>belongs_to :step</code>	Метод, обозначающий принадлежность к сущности <code>Step</code> .

Таблица 4.5 – Описание методов и свойств класса `Category`

Название элемента класса	Описание
Поле <code>id</code>	Первичный ключ для сущности.
Поле <code>slug</code>	Идентифицирует категорию при помощи текстового идентификатора.
Поле <code>created_at</code>	Время создания записи.
Поле <code>updated_at</code>	Время последнего обновления записи.
<code>resourcify</code>	Метод, проверяющий права для данной сущности при помощи пакета <code>CanCanCan</code> .
<code>has_many :manuals, dependent: :destroy</code>	Связанные сущности – инструкции. При удалении инструкции из базы данных, все связанные инструкции будут каскадно удалены.

Для того, чтобы иметь возможность переводить категории на другие языки было решено отказаться от идеи явно задавать название категории при помощи поля `name`. Был добавлен текстовый идентификатор `slug`, однозначно идентифицирующий категорию. Все соответствующие переводы были вынесены в конфигурационные файлы с локализацией, а для удобного доступа к нужному переводу (в том числе и в представлении) перевода был добавлен вспомогательный метод.

Таблица 4.6 – Описание методов и свойств класса Comment

Название элемента класса	Описание
Поле id	Первичный ключ для сущности.
Поле content	Содержимое комментария.
Поле user_id	Идентификатор пользователя, которому принадлежит комментарий.
Поле manual_id	Идентификатор инструкции, которой принадлежит комментарий.
Поле created_at	Время создания записи.
Поле updated_at	Время последнего обновления записи.
resourcify	Метод, проверяющий права для данной сущности при помощи пакета CanCanCan.
belongs_to :user, counter_cache: true	Обозначает принадлежность данной сущности к пользователю.
belongs_to :manual, counter_cache: true	Обозначает принадлежность данной сущности к инструкции.

Таблица 4.7 – Описание методов и свойств класса CompletedStep

Название элемента класса	Описание
Поле id	Первичный ключ для сущности.
Поле user_id	Идентификатор пользователя, которому принадлежит пройденный шаг.
Поле step_id	Идентификатор шага, который пользователь прошел.
Поле manual_id	Идентификатор инструкции шага.
Поле created_at	Время создания записи.
Поле updated_at	Время последнего обновления записи.

Таблица 4.8 – Описание методов и свойств класса Manual

Название элемента класса	Описание
Поле id	Первичный ключ для сущности.
Поле name	Обозначает название инструкции.
Поле preview	Обозначает картинку, которая в целом описывает данную инструкцию.
Поле user_id	Идентификатор пользователя – создателя инструкции.
Поле category_id	Идентификатор категории инструкции.
Поле comments_count	Закешированное количество комментариев, оставленных к данной инструкции. Пересчитывается при добавлении или удалении комментария.
Поле created_at	Время создания записи.
Поле updated_at	Время последнего обновления записи.

Таблица 4.9 – Описание методов и свойств класса Role

Название элемента класса	Описание
Поле id	Первичный ключ для сущности.
Поле name	Обозначает название роли на английском языке.
Поле resource_type	Обозначает тип ресурса, на котором определены роли.
Поле resource_id	Обозначает идентификатор ресурса. Связь с классом User осуществляется через промежуточную таблицу
Поле created_at	Время создания записи.
Поле updated_at	Время последнего обновления.

Таблица 4.10 – Описание методов и свойств класса Step

Название элемента класса	Описание
Поле id	Первичный ключ для сущности.
Поле name	Обозначает название шага в данной инструкции.
Поле priority	Обозначает номер шага в общем списке шагов для данной инструкции.
Поле manual_id	Обозначает идентификатор инструкции, к которой принадлежит шаг.
Поле user_id	Обозначает идентификатор пользователя, которому принадлежит данный шаг (пользователь, добавивший шаг к инструкции).
Поле created_at	Время создания записи.
Поле updated_at	Время последнего обновления.

Таблица 4.11 – Описание методов и свойств класса User

Название элемента класса	Описание
Поле id	Первичный ключ для сущности.
Поле email	Содержит название электронного ящика пользователя.
Поле encrypted_password	Содержит хешированный пароль пользователя. Необходим для работы пакета Devise.
Поле sign_in_count	Обозначает количество раз, сколько пользователь авторизовывался.
Поле current_sign_in_at	Обозначает время, когда пользователь аутентифицировался в текущую сессию.
Поле last_sign_in_at	Дата последнего входа пользователя.



Продолжение таблица 4.11

Название элемента класса	Описание
Поле <code>current_sign_in_ip</code>	IP адрес пользователя для текущей сессии.
Поле <code>provider</code>	Обозначает название стратегии <code>omniauth</code> , по которой пользователь был аутентифицирован.
Поле <code>uid</code>	Обозначает уникальный идентификатор пользователя для определенной стратегии аутентификации.
Поле <code>name</code>	Обозначает фамилию и имя пользователя, либо его второе имя. Заполняется самим пользователем.
Поле <code>about</code>	Обозначает информацию о пользователе, его биографию, факты из жизни. Заполняется самим пользователем.
Поле <code>interests</code>	Обозначает интересы пользователя. Заполняется самим пользователем.
Поле <code>manuals_count</code>	Обозначает количество инструкций, созданных пользователем. Данное поле обновляется и отслеживается при помощи встроенных в Rails средств.
Поле <code>comments_count</code>	Обозначает общее количество комментариев, созданных пользователем. Данное поле обновляется и отслеживается при помощи встроенных в Rails средств.
Поле <code>created_at</code>	Время создания записи.
Поле <code>updated_at</code>	Время последнего обновления записи.

Таблица 4.12 – Классы контроллеров и их краткое описание

Название класса	Краткое описание
ApplicationController	Главный контроллер приложения. Осуществляет перенаправление пользователя в случае попыток доступа неавторизованных пользователей. Хранит последнее местонахождение пользователя для перенаправления его на прежнюю страницу (а не на главную) после аутентификации через социальные сети. Устанавливает стандартную локаль для приложения. Сохраняет XSRF-токен в куки файлы для работы с AngularJS контроллерами без перезагрузки страниц редактирования.
BlocksController	Контроллер для осуществления управлением блоками, принадлежащих шагам.
CategoriesController	Контроллер для показа списка инструкций из выбранной категории.
CommentsController	Контроллер для управления комментариями к инструкциям.
ManualsController	Контроллер для управления инструкциями.
RaterController	Контроллер для управления рейтингом инструкций.
SearchesController	Контроллер для управления поисковыми запросами.
StepsController	Контроллер для управления шагами инструкций.
TagsController	Управление тегами к инструкциям.

Продолжение таблицы 4.12

Название класса	Краткое описание
UsersController	Управление пользователями.

Таблица 4.13 – Описание методов и свойств класса ApplicationController

Название элемента класса	Описание
store_current_location()	Сохраняет страницу, на которой находится пользователь в текущий момент. На нее будет перенаправлен пользователь после прохождения аутентификации.
after_sign_out_path_for()	Метод определяет, куда будет перенаправлен пользователь после выхода из своего аккаунта.
set_csrf_cookie_for_ng()	Сохраняет XSRF-TOKEN токен в куки. Метод необходим для осуществления запросов к API из AngularJS контроллеров.
set_locale()	Коллбэк, устанавливающий переданный либо стандартный язык для приложения.
verified_request?	Метод, осуществляющий проверку правильности установленного токена XSRF для запроса.

Таблица 4.14 – Описание методов и свойств класса BlocksController

Название элемента класса	Описание
index	Вывод списка всех блоков и рендерит их в необходимом формате.
show	Вывод и рендеринг в нужном формате блока.
new	Генерирует форму добавления нового блока.

Продолжение таблицы 4.14

Название элемента класса	Описание
create	Создает блок заданного типа (текст, картинка, видео) в соответствии с переданными параметром type.

Таблица 4.15 – Описание методов и свойств класса CategoriesController

Название элемента класса	Описание
index	Вывод списка всех блоков и рендерит их в формате json. Список используется для выбора категории при добавлении инструкции.
show	Вывод список инструкций по заданной категории из определенной страницы.

Таблица 4.16 – Описание методов и свойств класса CommentsController

Название элемента класса	Описание
new	Рендеринг формы для создания нового комментария.
create	Создание нового комментария и сделать его владельцем комментарием.
destroy	Удаление заданного комментария.

Таблица 4.17 – Описание методов и свойств класса ManualsController

Название элемента класса	Описание
index	Выводит список всех инструкций.
show	Рендерит и выводит запрошенную инструкцию.
new	Создает форму для создания новой инструкции.

Продолжение таблицы 4.17

Название элемента класса	Описание
create	Создает инструкцию и перенаправляет в зависимости от входных параметров.
update	Обновляет необходимую инструкцию, если пользователь имеет права на редактирование по отношению к инструкции.
destroy	Удаляет выбранную инструкцию из каталога.
set_manual	Метод необходим для поиска и установки нужного значения manual на основе входных параметров.
set_tags	Метод ищет и устанавливает теги для текущей инструкции.
set_manuels	В зависимости от того, что требуется: вывести инструкции по определенному тегу, вывести результаты поиска инструкций по страницам, либо просто вывести все инструкции с пагинацией.
set_comments	Устанавливает список комментариев перед тем как будет отрисована инструкция.
set_completed_steps	Для зарегистрированных пользователей метод устанавливает дополнительную переменную
manual_params	Проверка входных параметров и удаление ненужных.

Таблица 4.18 – Описание методов и свойств класса SearchesController

Название элемента класса	Описание
index	Выводит список всех инструкций.
set_manually	Ищет инструкции в базе данных на основании запроса query. Для индексации базы данных и полнотекстового поиска используется gem search_core.
query	Подготавливает запрос для его выполнения в set_manually методе.

Таблица 4.19 – Описание методов и свойств класса StepsController

Название элемента класса	Описание
index	Выводит список всех шагов.
show	Рендерит и выводит конкретный шаг.
new	Создает форму для создания шага
create	Создание инструкции с заданными параметрами для заданной инструкции для текущего пользователя.
update	Метод позволяет обновить запись. Используется после перетаскивания шагов для обновления их приоритета.
destroy	Удаление шага.
set_update_step	Установка шага для обновления.
set_show_step	Установка шага для просмотра.
set_completed_attributes	Если пользователь авторизован, то шаг помечается как пройденный в момент посещения страницы с данным шагом либо его API.
step_params	Проверка входных параметров и удаление лишних.

Таблица 4.20 – Описание методов и свойств класса TagsController

Название элемента класса	Описание
index	Выводит список всех добавленных тегов в формате json в случае если не передан параметр :query. Если передан параметр :query, то ищутся все искомые теги и отправляются клиенту. Во время набора тега, форма отправляет на сервер набранное пользователем и, в случае если найдены теги, предлагает их в качестве автоматического дополнения тегов. Со стороны клиента запрос делается через AngularJS.

Таблица 4.21 – Описание методов и свойств класса TagsController

Название элемента класса	Описание
index	Выводит список всех добавленных тегов в формате json в случае если не передан параметр :query. Если передан параметр :query, то ищутся все искомые теги и отправляются клиенту. Во время набора тега, форма отправляет на сервер набранное пользователем и, в случае если найдены теги, предлагает их в качестве автоматического дополнения тегов. Со стороны клиента запрос делается через AngularJS по мере набора пользователем текста для нового тега к инструкции.

Таблица 4.22 – Описание методов и свойств класса UsersController

Название элемента класса	Описание
show	Метод для получения информации о пользователе в формате json.
edit	Метод для создания формы редактирования пользователя на AngularJS.
update	Обновление информации о пользователе после ее редактирования.
destroy	Удаление пользователя.
set_user	Поиск пользователя по уникальному идентификатору.
user_params	Пользовательские параметры.
set_related	Связанные с пользователем сущности: инструкции и комментарии.
ManualsHelper	Содержит методы работы с инструкциями. В основном используются в представлении.

Для того, чтобы не писать в контроллерах и представлениях дополнительную логику был введен ряд вспомогательных методов (Rails Helpers).

Таблица 4.23 – Модули вспомогательных методов и их краткое описание

Название модуля	Краткое описание
ApplicationHelper	Основной модуль, содержащий общие вспомогательные методы.
CategoriesHelper	Модуль содержит методы для работы с названием категории.
StepsHelper	Модуль содержит методы для работы с шагами и их переключением при просмотре



Продолжение таблицы 4.23

Название элемента класса	Описание
ManualsHelper	Содержит методы работы с инструкциями. В основном используются в представлении.

Таблица 4.24 – Описание методов и свойств модуля ApplicationHelper

Название элемента класса	Описание
categories	Возвращает список всех категорий (для рендеринга их на главной странице).
tags	Возвращает топ-30 самых используемых тегов.
markdown(text)	Принимает текст в формате markdown и возвращает исходные коды html.
data_user_id	Получение id текущего пользователя (для логики работы с комментариями).

Таблица 4.25 – Описание методов и свойств модуля CategoriesHelper

Название элемента класса	Описание
translation_by_slug(slug, locale)	Принимает псевдо-имя категории slug и возвращает локализованное имя в соответствии с переданной локалью.

Таблица 4.26 – Описание методов и свойств модуля StepsHelper

Название элемента класса	Описание
previous_step	Добавляет ссылку на предыдущий шаг, если такой существует.
next_step	Ссылка на следующий шаг, если текущий не является последним.

## Продолжение таблицы 4.26

Название элемента класса	Описание
ordered_blocks(step)	Выводит блоки в необходимой последовательности в соответствии с полем priority, отвечающий за местоположение блока в списке.

В результате разработки мы получили следующую схему базы данных.

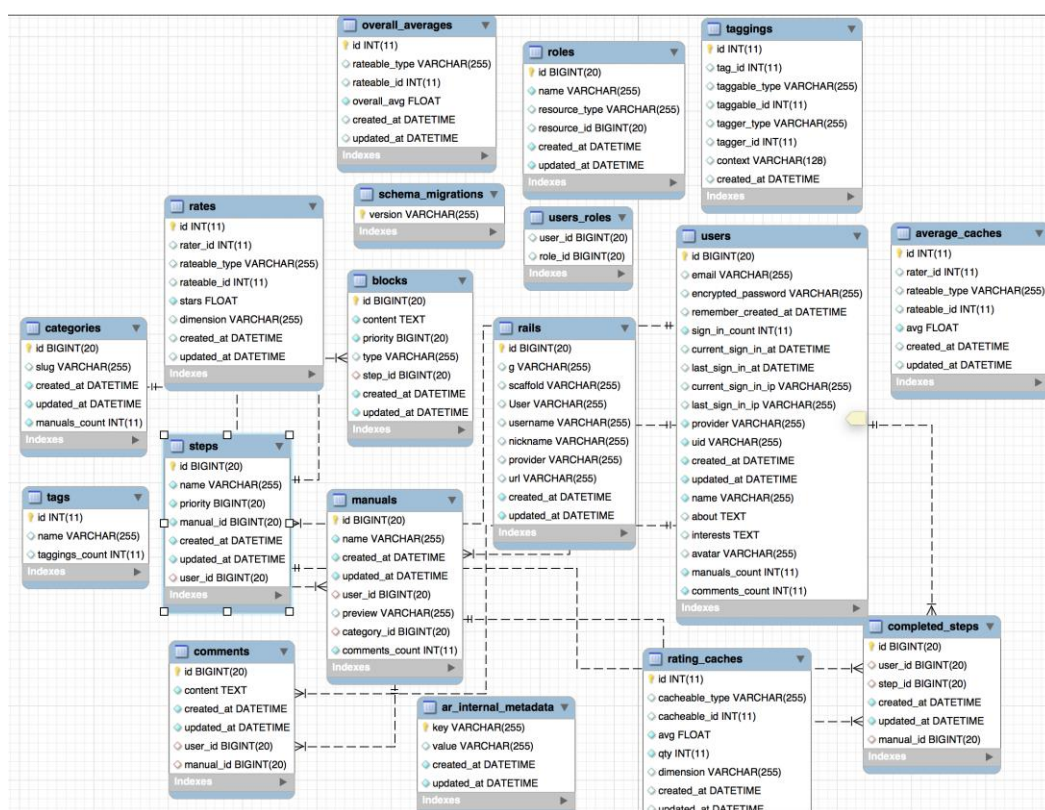


Рисунок 4.27 – Схема базы данных

## 5 ТЕСТИРОВАНИЕ

Было проведено тестирование приложения с точки зрения пользовательского интерфейса, возможности взлома приложения, а также проверок входных данных на уровне базы данных.

В ходе проверки работы интерфейса приложения при помощи браузера Safari были выявлены некоторые недочеты в дизайне приложения для мобильных устройств. Они были устранены путем использования элементов адаптивной верстки фреймворка Bootstrap.

Для проведения тестирования приложения с точки зрения безопасности была использована программа PostMan, позволяющая делать прямые HTTP запросы к приложению и тем самым определить уязвимости.

Таблица 5.1 – Тестирование безопасности программного средства

Запрос	Ответ	Ожидаемый ответ	Пояснение
GET /manuals/new	Cache-Control: no-cache Connection: keep-alive Content-Type: text/html Date: Sun, 17 Dec 2017 00:52:43 GMT Server: Cowboy Set-Cookie: _i_manual_session=; path=/; HttpOnly Transfer-Encoding: chunked Via: 1.1 vegur X-Content-Type-Options: nosniff X-Runtime: 0.0215	Cache-Control: no-cache Connection: keep-alive Content-Type: text/html Date: Sun, 17 Dec 2017 00:52:43 GMT Server: Cowboy Set-Cookie: _i_manual_session=; path=/; HttpOnly Transfer-Encoding: chunked Via: 1.1 vegur X-Content-Type-Options: nosniff X-Runtime: 0.0215	Неавторизованный пользователь не получил доступ к странице добавления инструкции.

Продолжение таблицы 5.1

Запрос	Ответ	Ожидаемый ответ	Пояснение
PUT manual.hero kuapp.com/ manuals/1	Cache-Control: no-cache Connection: keep-alive Content-Type: text/html Date: Sun, 17 Dec 2017 01:00:05 GMT Server: Cowboy Transfer-Encoding: chunked Via: 1.1 vegur X-Content-Type-Options: nosniff X-Frame-Options: SAMEORIGIN X-Request-Id: 17b1109e-10eb-4571-888b-9e668a945d52 X-Runtime: 0.021502 X-Xss-Protection: 1; mode=block	Cache-Control: no-cache Connection: keep-alive Content-Type: text/html Date: Sun, 17 Dec 2017 01:00:05 GMT Server: Cowboy Transfer-Encoding: chunked Via: 1.1 vegur X-Content-Type-Options: nosniff X-Frame-Options: SAMEORIGIN X-Request-Id: 17b1109e-10eb-4571-888b-9e668a945d52 X-Runtime: 0.021502 X-Xss-Protection: 1; mode=block	Неавторизированному пользователю не удалось обновить находящуюся на сайте инструкцию.

Для тестирования функции бесконечного прокручивания страниц был использован браузер Safari. Перед тестом был автоматически сгенерирован тестовый набор данных при помощи стандартной консоли Rails и небольшого скрипта. Данная функция была проверена и функционирует исправно.

Было проведено тестирования функциональности веб-приложения в целом при помощи инструмента Selenium IDE.

Selenium – это инструмент для автоматизированного управления браузерами. Наиболее популярной областью применения Selenium является автоматизация тестирования веб-приложений.

Разработка Selenium поддерживается производителями популярных браузеров. Они адаптируют браузеры для более тесной интеграции с Selenium,

а иногда даже реализуют встроенную поддержку Selenium в браузере. Selenium является центральным компонентом целого ряда других инструментов и фреймворков автоматизации.

Selenium поддерживает десктопные и мобильные браузеры. Selenium позволяет разрабатывать сценарии автоматизации практически на любом языке программирования. С помощью Selenium можно организовывать распределённые стенды, состоящие из сотен машин с разными операционными системами и браузерами, и даже выполнять сценарии в облаках.

Были созданы тест-кейсы при помощи Selenium IDE расширения для браузера Mozilla Firefox для тестирования основного функционала приложения.

Проверка доступности кнопок авторизации через социальные сети:

- open /;
- click `//*[@id="bs-example-navbar-collapse-1"]>ul[1]>li/a;`
- waitForElementPresent link=Sign in with Facebook;
- verifyElementPresent link=Sign in with Facebook;
- waitForElementPresent link=Sign in with Vk;
- verifyElementPresent link=Sign in with Vk;
- waitForElementPresent link=Sign in with Twitter;
- verifyElementPresent link=Sign in with Twitter;
- waitForElementPresent link=Sign in with Google;
- verifyElementPresent link=Sign in with Google.

Проверка авторизации через социальную сеть Твиттер:

- open /;
- click css=strong;
- clickAndWait link=Sign in with Twitter;
- type id=username\_or\_email AutoTes59161594;
- type id=password seleniumtest123456;
- clickAndWait id=allow;
- assertTextPresent Successfully authenticated from Twitter account;
- waitForVisible `//*[@id="bs-example-navbar-collapse-1"]>ul[1]>li/a;`
- click `//*[@id="bs-example-navbar-collapse-1"]>ul[1]>li/a;`
- clickAndWait link=Sign out;

- assertTextPresent Signed out successfully.

Проверка авторизации через социальную сеть Твиттер, если пользователь не разрешил вход через свой аккаунт:

- open /;
- click link=Sign in;
- clickAndWait link=Sign in with Twitter;
- clickAndWait id=cancel;
- clickAndWait link=Вернуться к iManual-Local;
- assertText `//*[@id='bs-example-navbar-collapse-1']/ul[1]/li/a/strong` Sign in.

Проверка возможности смены языка интерфейса:

- open /manuals?locale=en;
- click css=span.glyphicon.glyphicon-globe;
- clickAndWait link=Русский;
- assertLocation /manuals?locale=ru;
- assertTextPresent Все инструкции.

Проверка возможности создания инструкций с единственным шагом:

- open /manuals?locale=en;
- click `//*[@id='bs-example-navbar-collapse-1']/ul[1]/li/a`;
- clickAndWait link=Sign in with Twitter;
- clickAndWait id=allow;
- waitForVisible link=New;
- clickAndWait link=New;
- waitForVisible id>manual\_name;
- type id>manual\_name Auto Test;
- clickAndWait `css=div.col-lg-offset-2.col-lg-10` &gt;  
input[name='commit'];
- waitForVisible document.forms['form.state'].elements[2];
- select document.forms['form.state'].elements[2] label=Craft;
- pause 2000;
- type xpath=(//input[@type='text'])[4] First;
- click css=input.btn.btn-primary;
- clickAndWait link=Preview;
- assertTextPresent Auto Test;
- assertTextPresent Craft.

Проверка возможности удаления созданной инструкции:

- open /manuals?locale=en;
- clickAndWait xpath=(//a[contains(text(),'Read')])[1];
- click css=span.glyphicon.glyphicon-trash;
- assertConfirmation Are you sure?;
- waitForTextPresent Manual was successfully destroyed;
- assertTextPresent Manual was successfully destroyed.

На основании данных тест-кейсов был составлен следующий тестовый набор:

- проверка доступности ссылок для входа через социальные сети;
- проверка возможности успешной авторизации через социальную сеть;
- проверка появления ошибки при неуспешной авторизации;
- проверка смены языка;
- проверка создания инструкций;
- проверка удаление инструкций;
- проверка возможности выхода из аккаунта.

Все тесты в среде плагина Selenium IDE для браузера Mozilla Firefox версии 50.0 прошли успешно.

## 6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Регистрация и авторизация в приложении:

В приложении возможно зарегистрироваться или авторизоваться при помощи социальных сетей. Для этого с любой страницы приложения необходимо нажать в правом верхнем углу на кнопку «Sign in» (в русскоязычной версии приложения «Войти»).

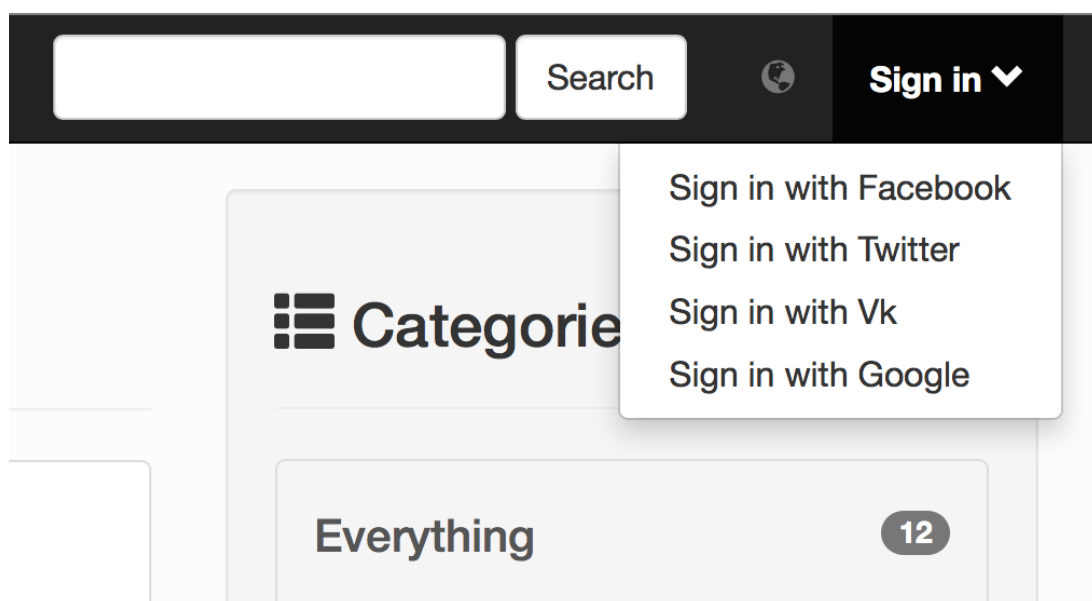


Рисунок 6.1 – Кнопки для авторизации в приложении

Затем выбираем, например, авторизацию через Твиттер нажатием на кнопку «Sign in with Twitter». Далее нас перенаправляет на страницу входа в Твиттер аккаунт как показано на рисунке 6.2

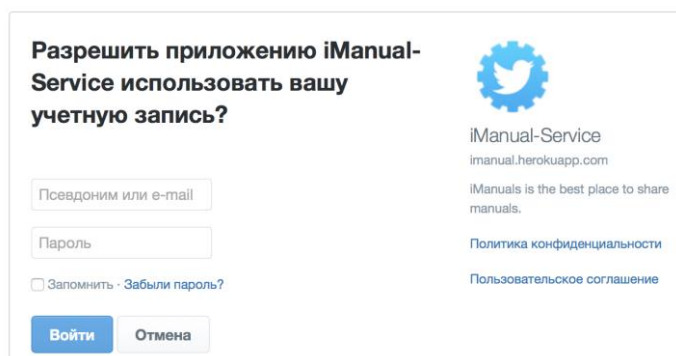


Рисунок 6.2 – Форма для авторизации в Твиттере



Вводим данные от своего аккаунта в социальной сети, нажимаем «Войти». Далее социальная сеть перенаправит нас обратно к приложению. Далее на сайте появится уведомление об успешной авторизации, как показано на рисунке 6.3

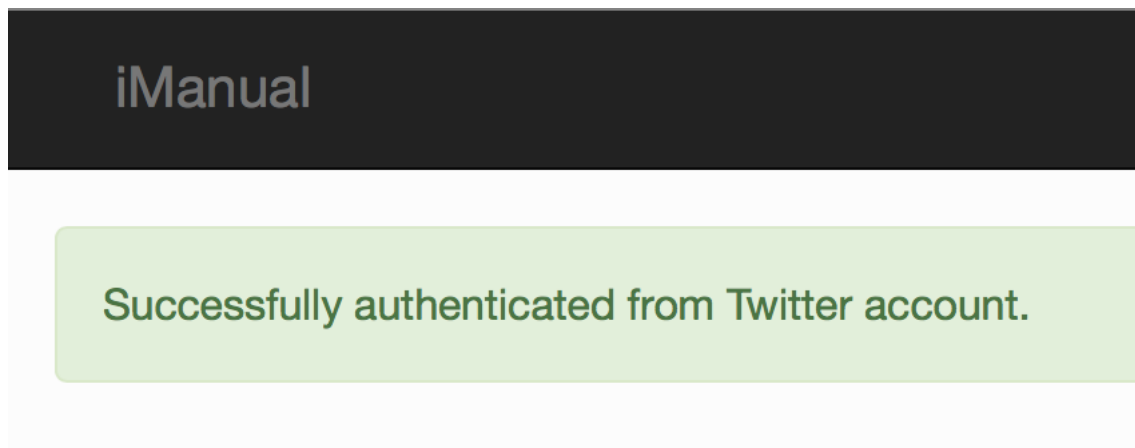


Рисунок 6.3 – Успешная авторизация в приложении при помощи Твиттера

Смена языка в приложении:

Для смены языка необходимо нажать на кнопку с глобусом, как показано на рисунке 6.4.

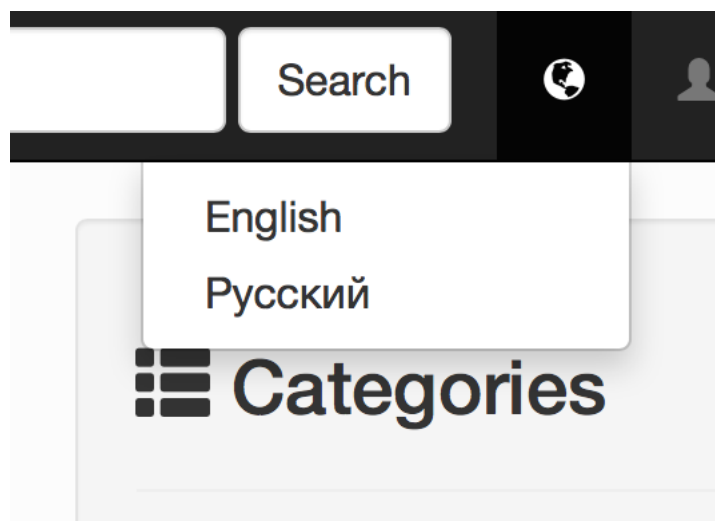


Рисунок 6.4 – Выбор языка приложения

Затем выбираем необходимый язык, после чего язык приложения сменится и в адресной строке можно увидеть изменение.

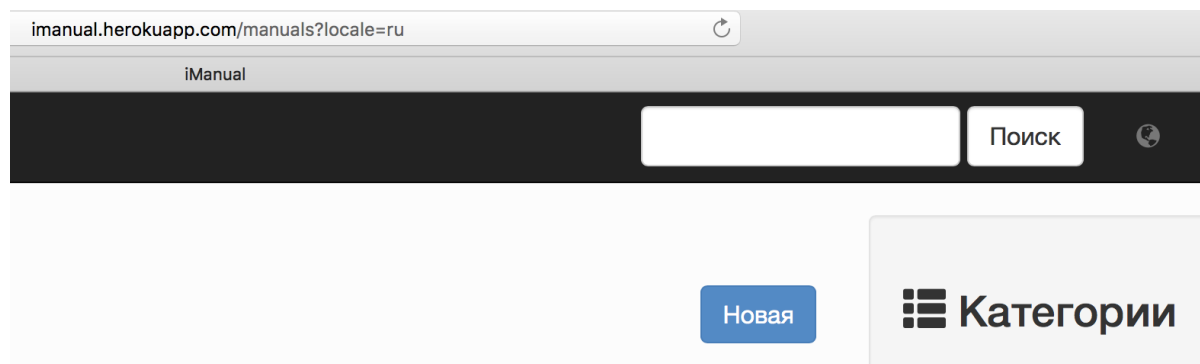


Рисунок 6.5 – Локализованное приложение

Как пройти инструкцию:

Для прохождения инструкции сначала необходимо выбрать инструкцию, затем нажать кнопку «Read» (в англоязычной версии сайта) как показано на рисунке 6.6. Также, в метаданных инструкции можно посмотреть прогресс, то есть количество пройденных шагов в инструкции.

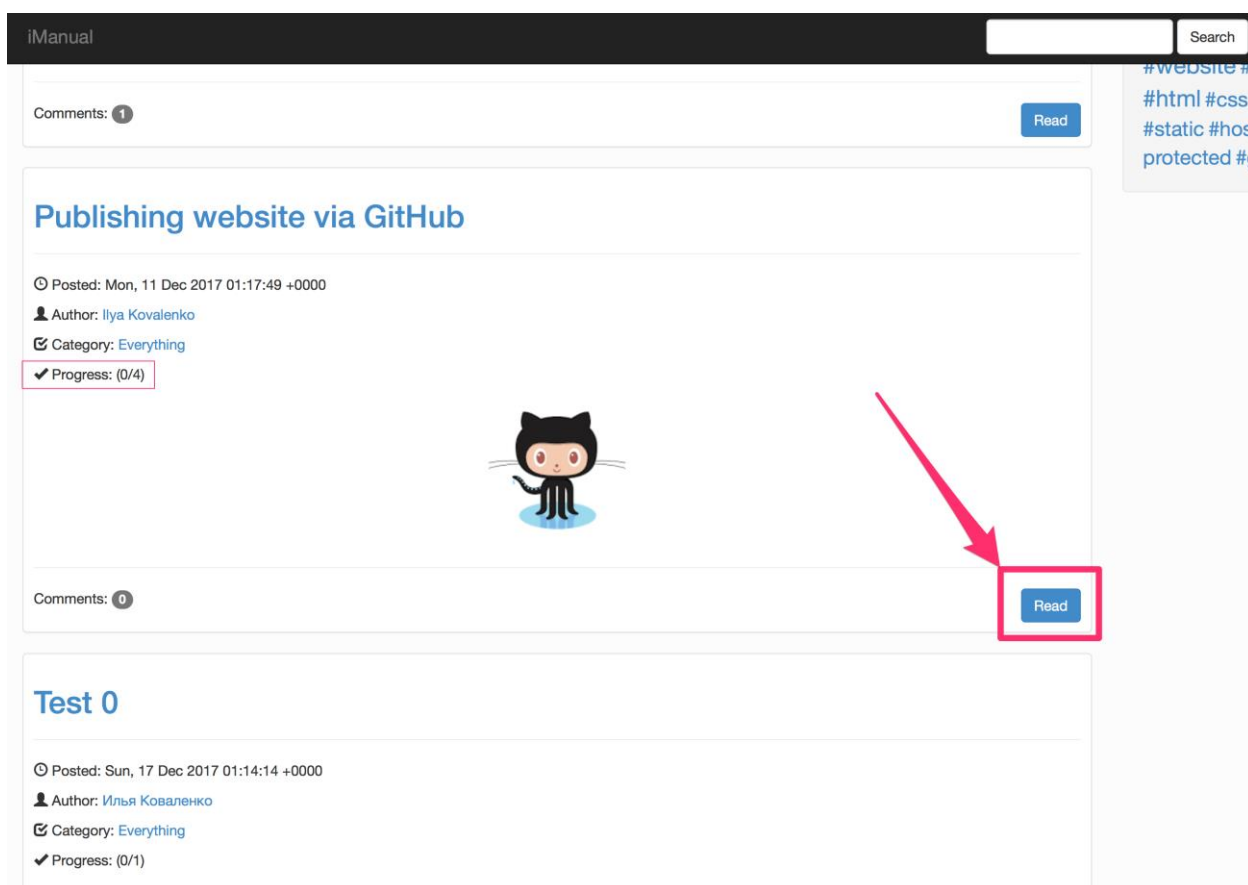


Рисунок 6.6 – Начало прохождения инструкции

После перехода к полной версии инструкции можно посмотреть на

список шагов, узнать пройденные, прочитать комментарии к инструкции, выставить рейтинг инструкции, как показано на рисунке 6.7.

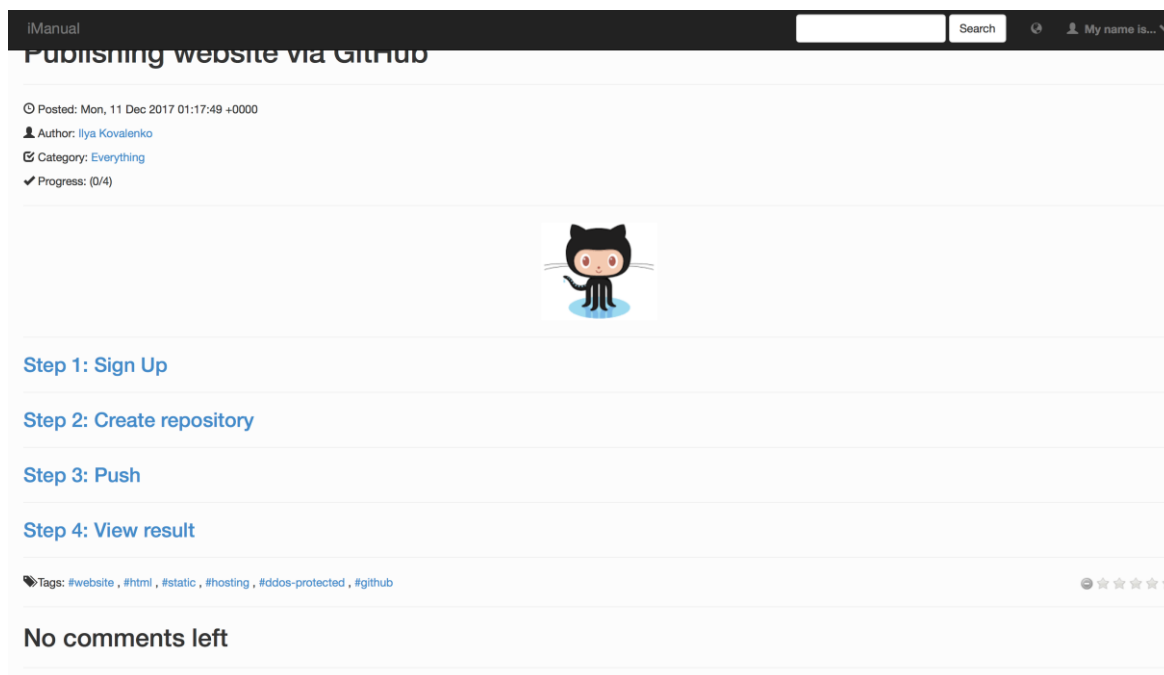


Рисунок 6.7 – Вид страницы с инструкцией

Для начала прохождения нажимаем на ссылку содержащую «Step 1:». Затем мы попадаем на страницу шага как показано на рисунке 6.8.



Рисунок 6.8 – Просмотр шага инструкции

При помощи кнопки «Next» можно перейти к следующему шагу инструкции. При помощи ссылки «Back to manual» можно вернуться на страницу с обобщенной информацией о данной инструкции. Нажмем еще раз на кнопку «Next», а затем выйдем обратно к общей информации об инструкции. Пройденные шаги выделены при помощи зеленой галочки, как

показано на рисунке 6.9. Это сделано для того, чтобы можно было вернуться к выполнению инструкции позже.

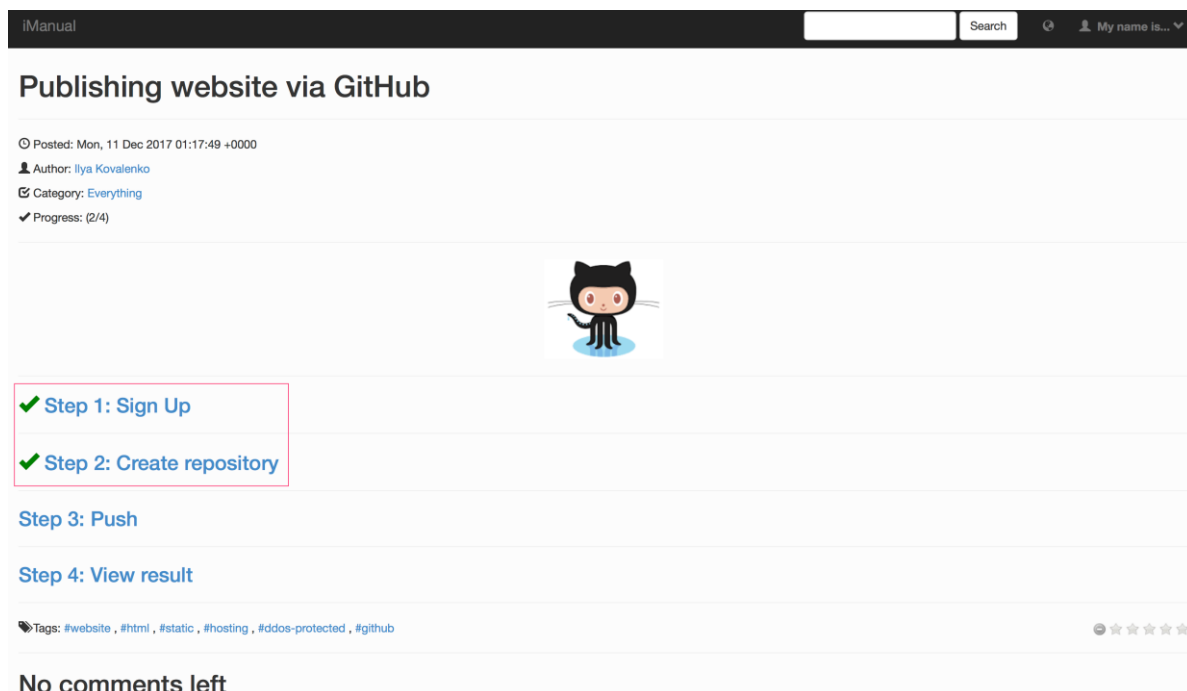


Рисунок 6.9 – Пройденные шаги в инструкции

Как написать комментарий к инструкции:

Сперва необходимо выбрать инструкцию, к которой нужно оставить комментарий. Переходим к полной версии инструкции как показано на рисунке 6.10.

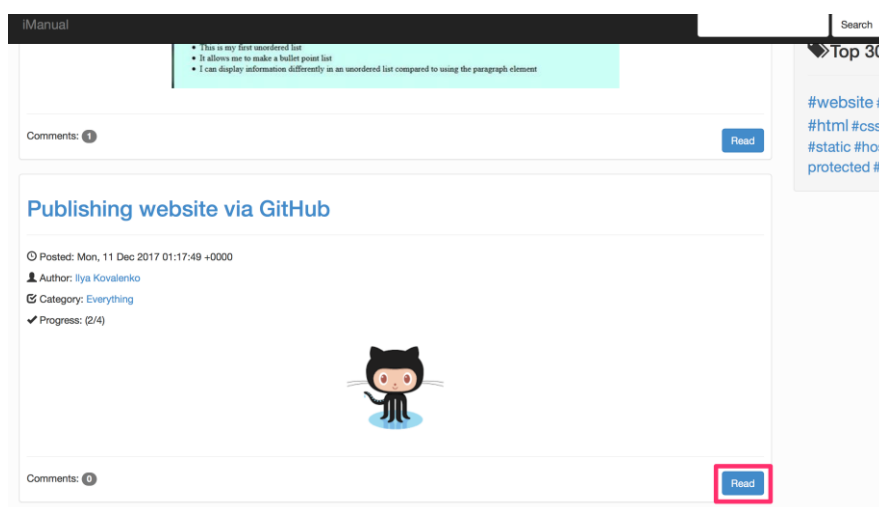


Рисунок 6.10 – Переход к полной версии инструкции

Затем в специальном поле вводим текст своего комментария и нажимаем «Send» как показано на рисунке 6.11.



The screenshot shows a web interface with the heading "No comments left" in large, bold, black text. Below this is a light gray rectangular box containing the text "Leave comment:". Underneath is a text input field with the placeholder text "Test comment 123". At the bottom left of this box is a blue button with the word "Send" in white text. The button is highlighted with a thick red rectangular border.

Рисунок 6.11 – Написание нового комментария

После нажатия по кнопке отправки, комментарий в режиме реального времени будет доставлен всем посетителям страницы, как показано на рисунке 6.12.

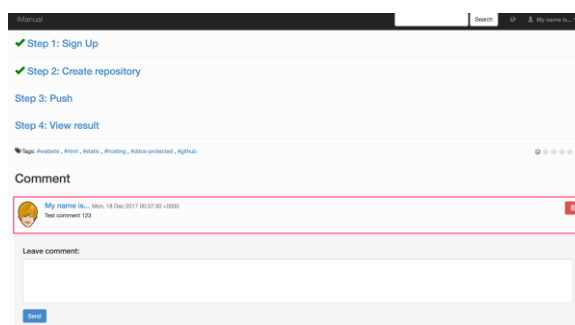


Рисунок 6.12 – добавленный комментарий к инструкции

## Установка рейтинга инструкции:

Сначала необходимо перейти к полной версии инструкции как показано на рисунке 6.10. Далее выставляем нужный рейтинг как показано на рисунке 6.13.

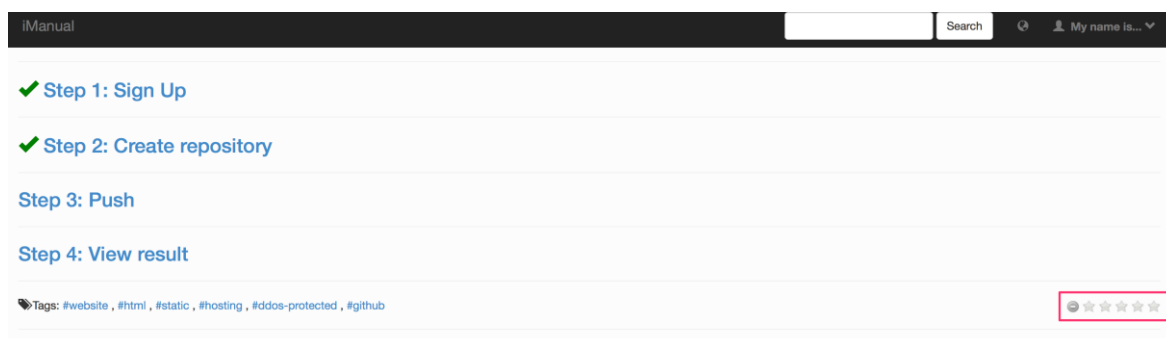


Рисунок 6.13 – Выставление рейтинга инструкции

После выставления рейтинга значок рейтинга примет вид как на рисунке 6.14.

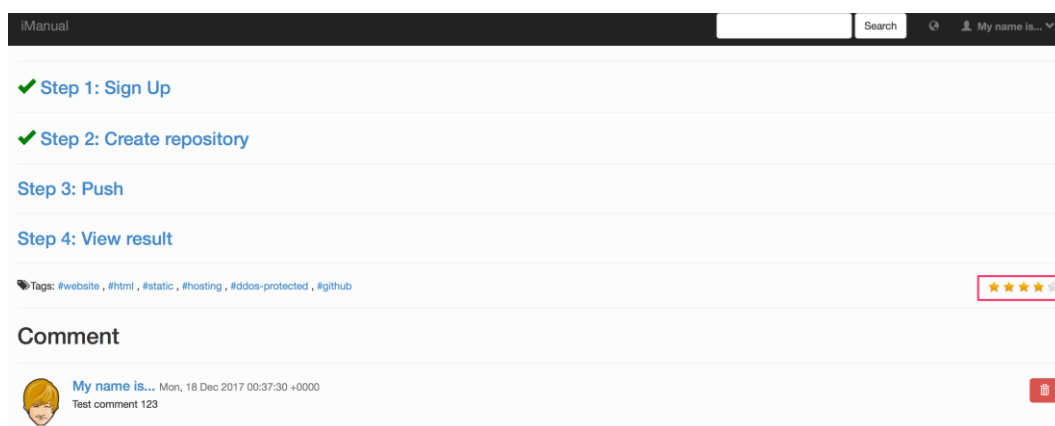


Рисунок 6.14 – Рейтинг инструкции

## Поиск инструкций:

Поиск инструкций является полнотекстовым и его можно осуществлять с любой страницы веб-приложения. Для осуществления поиска первым делом необходимо ввести искомую комбинацию символов в строку поискового запроса и затем нажать кнопку «Search» (в англоязычной версии веб-приложения) как показано на рисунке 6.15.

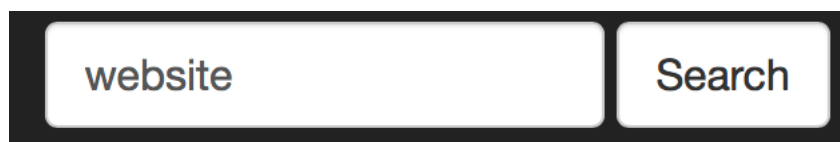


Рисунок 6.15 – Форма поиска инструкций

После нажатия на кнопку поиска приложение нас перенаправляет на страницу с результатами поиска, как показано на рисунке 6.16.

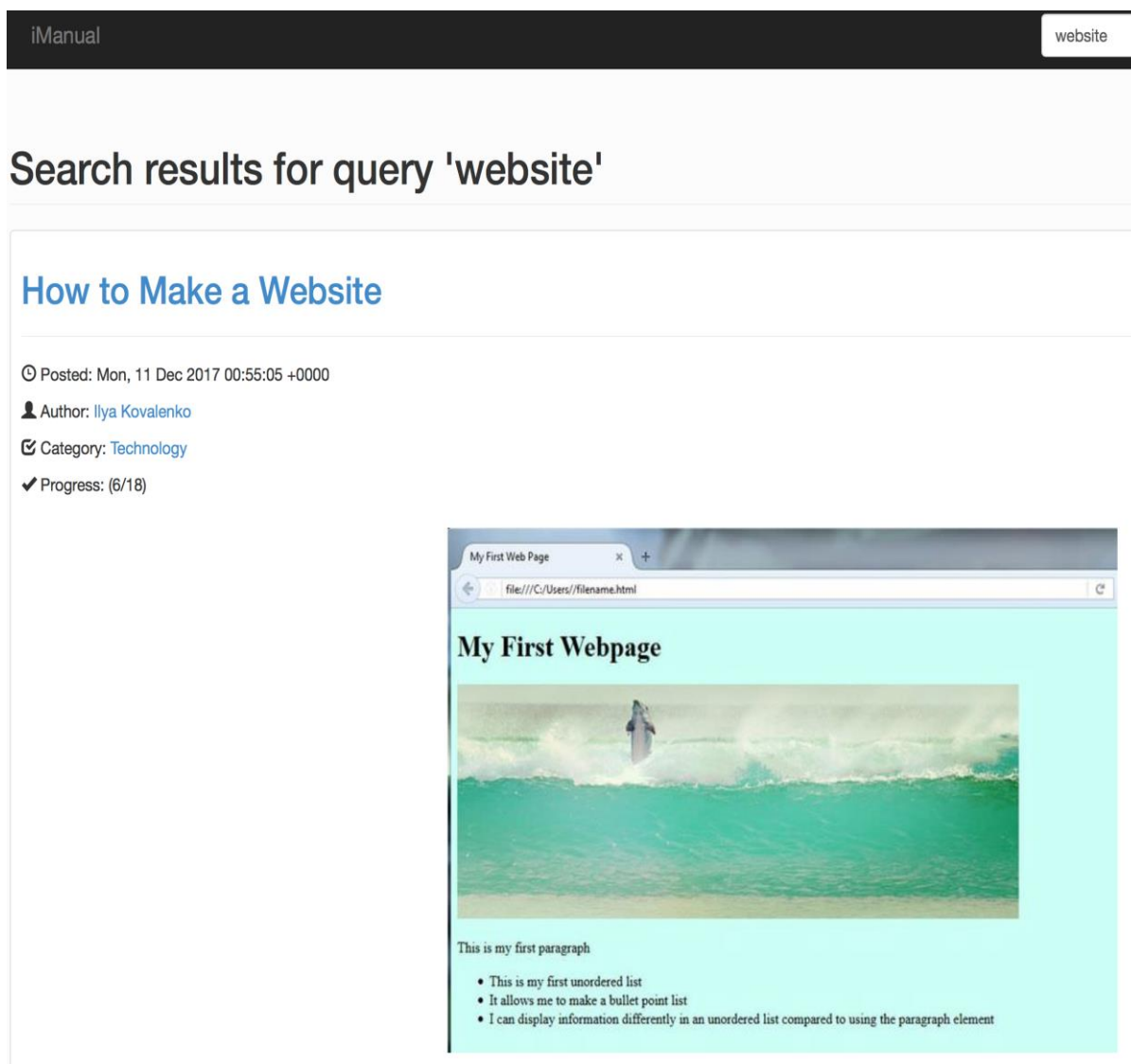


Рисунок 6.16 – Результаты полнотекстового поиска

Создание инструкции:

Для создания инструкции переходим на главную страницу приложения и нажимаем на кнопку «New» как показано на рисунке 6.17.

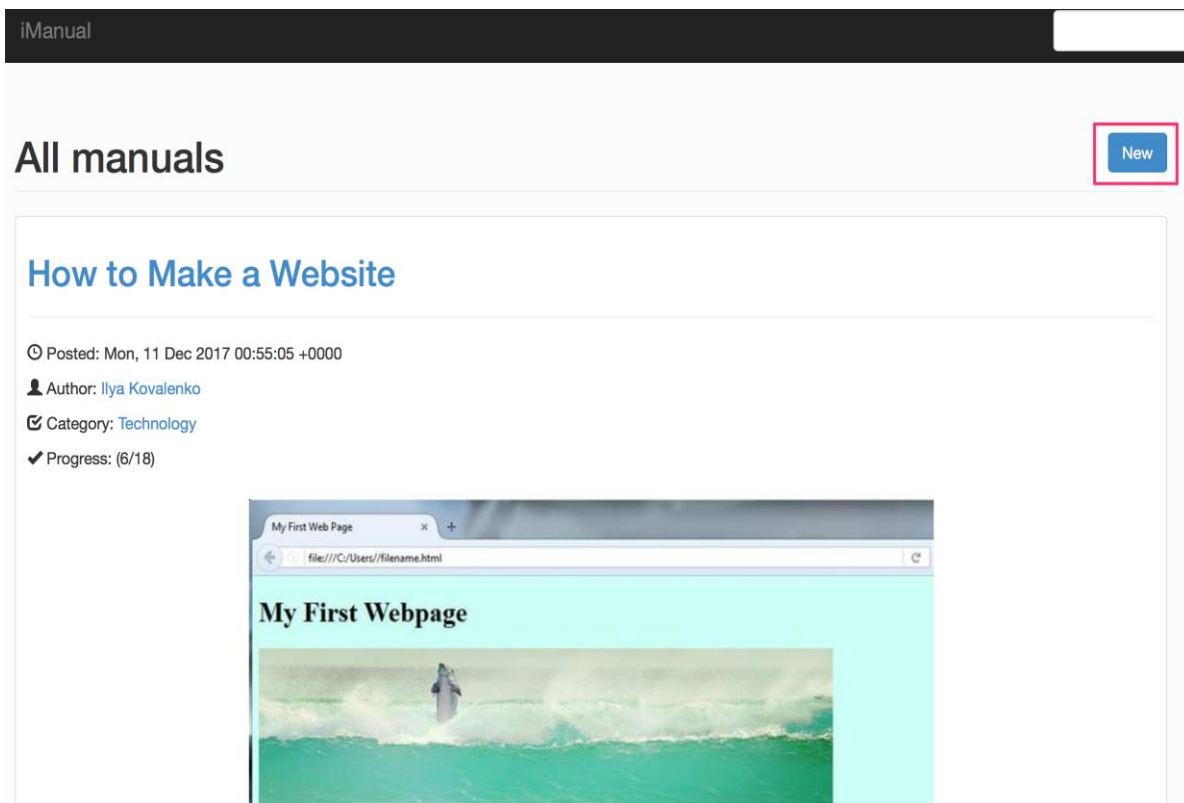


Рисунок 6.17 – Начало создания инструкции

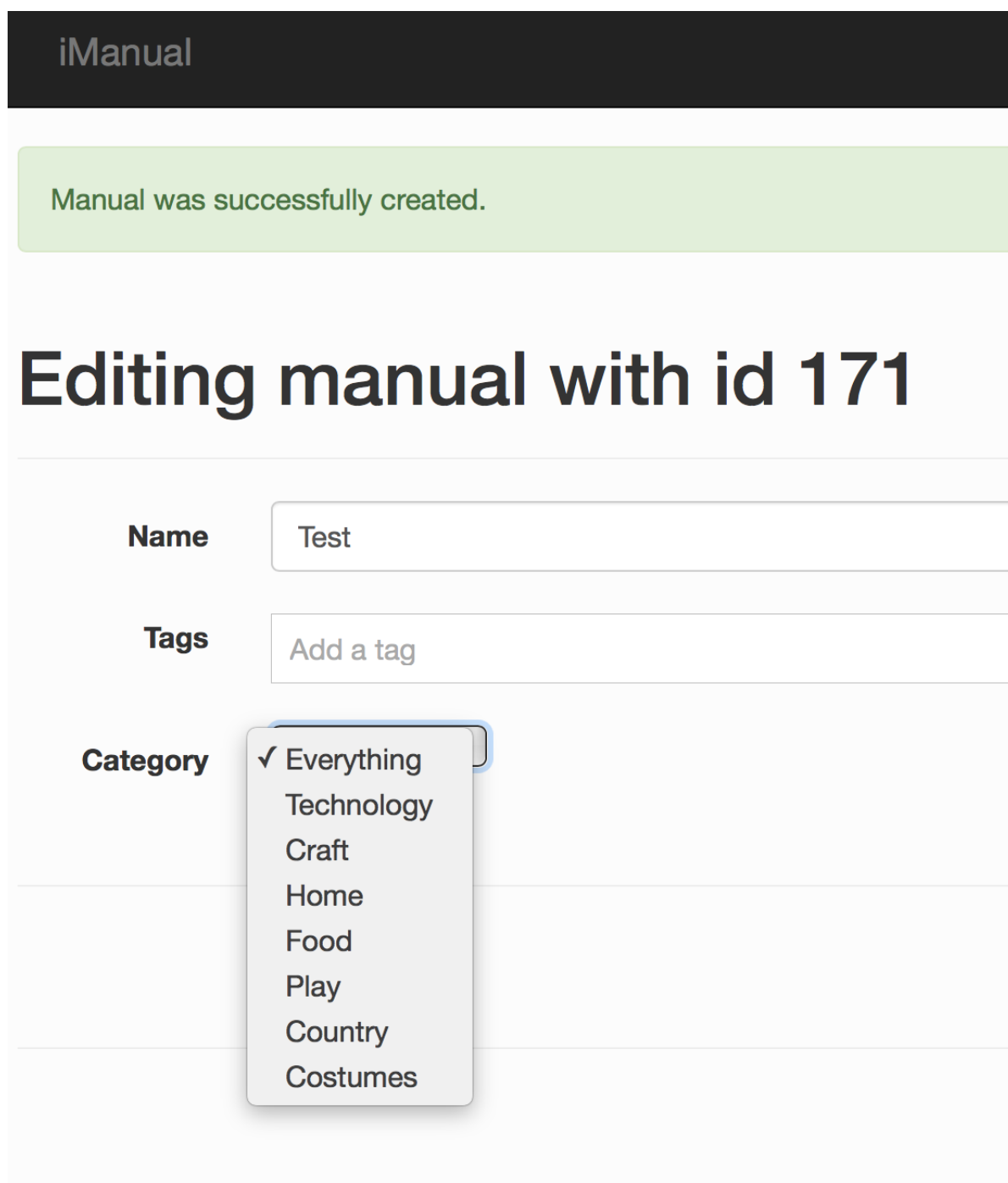
Далее вводим название новой инструкции, которую нужно добавить на сайт.

The screenshot shows the 'New manual' form in the iManual application. At the top, there is a dark header with the 'iManual' logo. Below the header, the main content area has the title 'New manual' in large, bold, black text. Below the title is a horizontal line. Underneath the line, there is a label 'Name' in bold black text. To the right of the label is a text input field containing the word 'Test'. Below the input field are two buttons: a blue 'Create' button and a white 'Cancel' button with a grey border.

Рисунок 6.18 – Ввод названия новой инструкции



После ввода названия нажимаем на кнопку «Create» и переходим к редактору инструкций. Сперва необходимо выбрать категорию будущей инструкции.



iManual

Manual was successfully created.

## Editing manual with id 171

**Name**

**Tags**

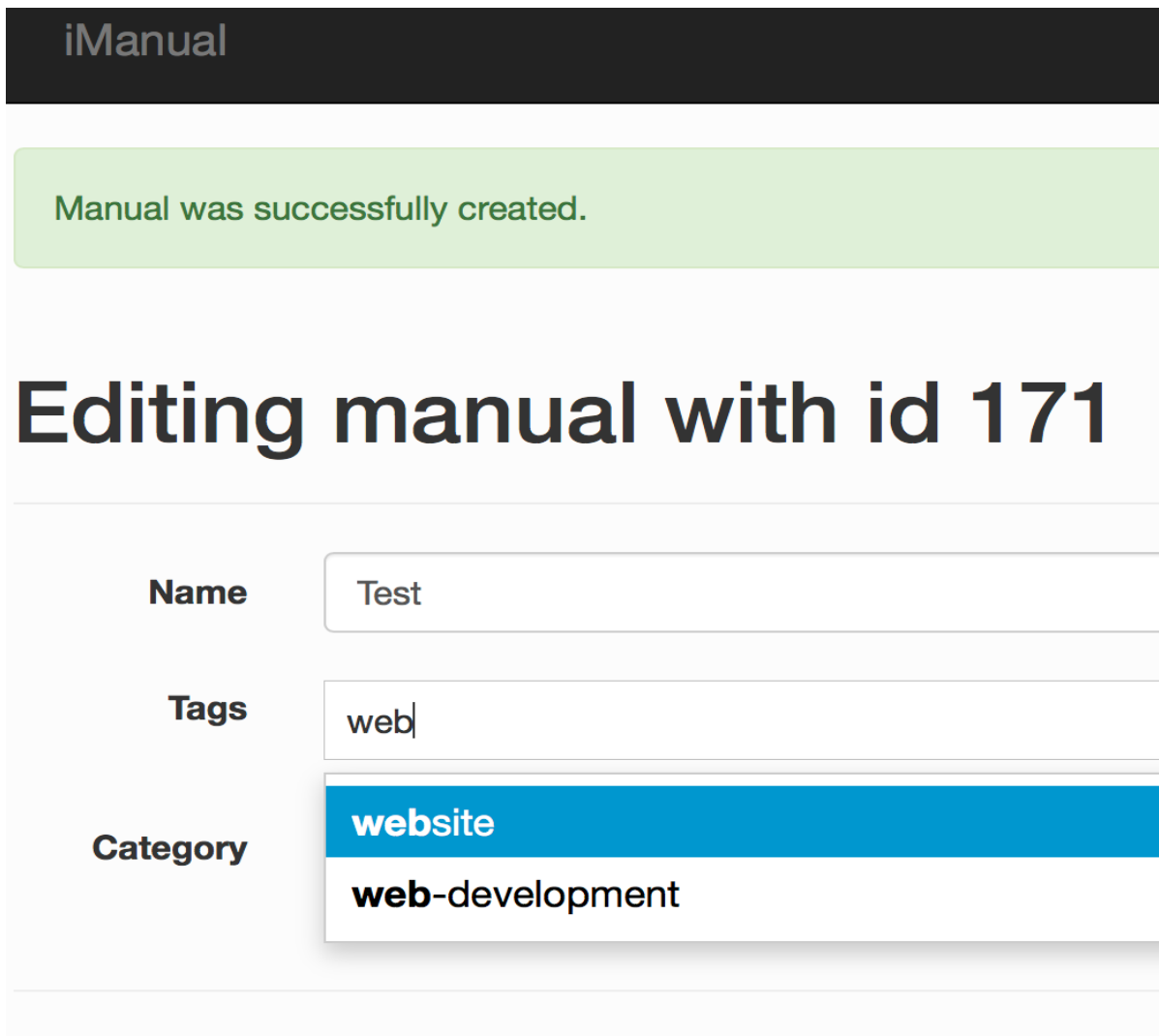
**Category**

- ✓ Everything
- Technology
- Craft
- Home
- Food
- Play
- Country
- Costumes

Рисунок 6.19 – Выбор категории для инструкции

После выбора категории для новой инструкции можно задать несколько тегов, по которым пользователи смогут найти нашу новую инструкцию. После

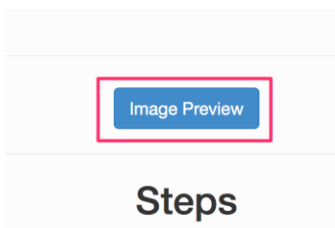
начала ввода тега появляется окно, предлагающее автоматически завершить ввод одним из уже известных системе тегов, как показано на рисунке 6.20.



The screenshot shows the iManual interface. At the top, a dark header contains the text "iManual". Below it, a green notification box states "Manual was successfully created." The main heading is "Editing manual with id 171". Below the heading is a form with three fields: "Name" with the value "Test", "Tags" with the value "web", and "Category". The "Category" field is open, showing a dropdown menu with two options: "website" (highlighted in blue) and "web-development".

Рисунок 6.20 – Автоматическое дополнение тегов

При помощи кнопки Image Preview (показанной на рисунке 6.21) можно загрузить заглавное изображение для инструкции, которое содержит краткую информацию о ней.



The screenshot shows a button labeled "Image Preview" in a blue box, which is highlighted by a red rectangle. Below the button is the text "Steps".

Рисунок 6.21 – Кнопка для загрузки превью

Для добавления нового шага к инструкции необходимо ввести его название и нажать кнопку New step, как показано на рисунке 6.22.

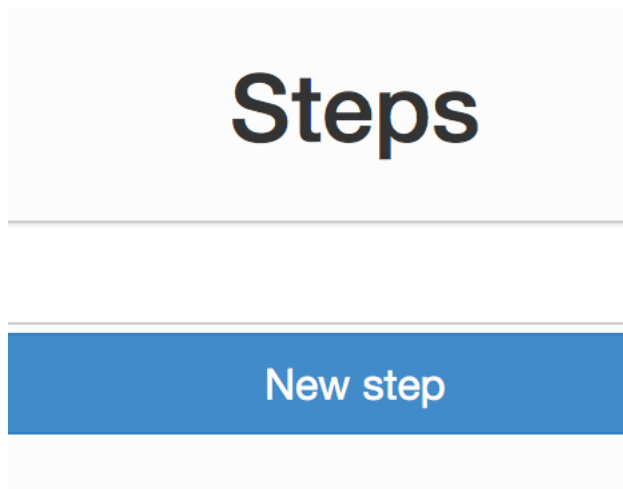


Рисунок 6.22 – Добавление нового шага

После добавления шаг можно либо удалить, либо наполнить информацией, необходимой для его успешного прохождения. Для добавления блоков внутрь шага нажимаем по шагу, как показано на рисунке 6.23 и переходим к редактированию шага.

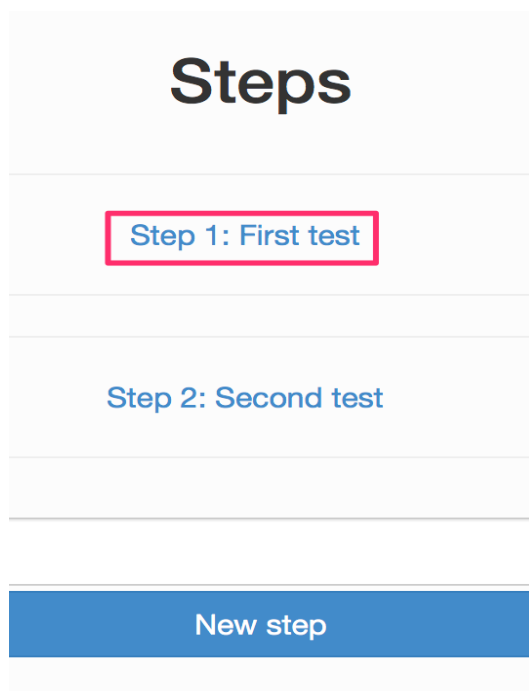


Рисунок 6.23 – Ссылка на редактирование шага

Далее можно видеть страницу редактирования блоков выбранного шага. Для добавления нового блока нужно нажать одну из кнопок, представленных на рисунке 6.24.

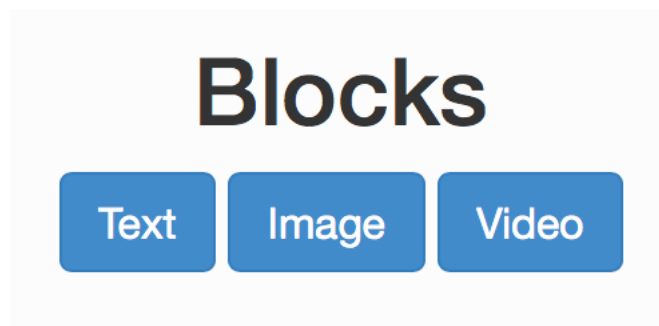


Рисунок 6.24 – Доступные блоки для добавления

Для добавления текстового блока выберем кнопку Text. Произошло добавление блока с текстом на страницу, куда можно добавить свой текст, как на рисунке 6.25.

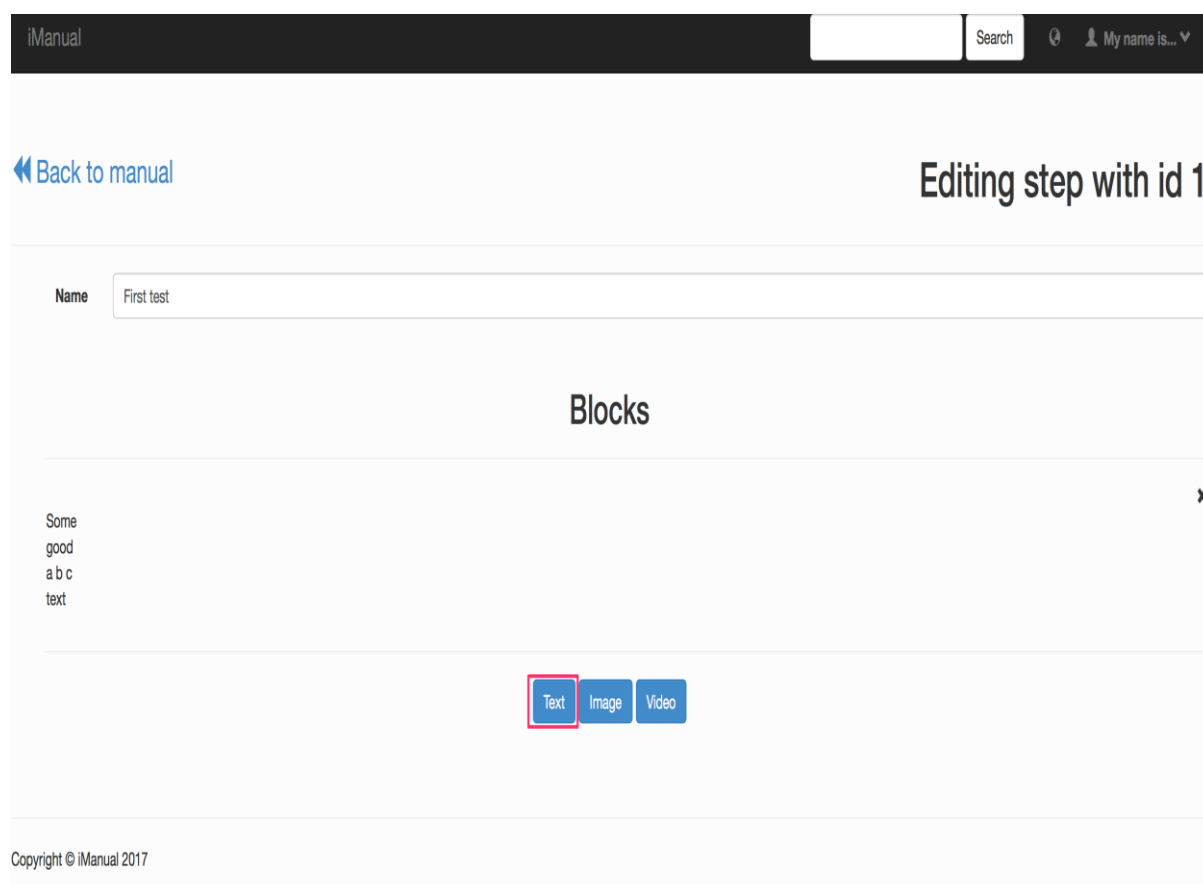


Рисунок 6.25 – Добавление текстового блока

Для добавления картинки выберем кнопку Image. Появится диалоговое окно с предложением выбрать нужную картинку (как показано на рисунке 6.26) для вставки в шаг.

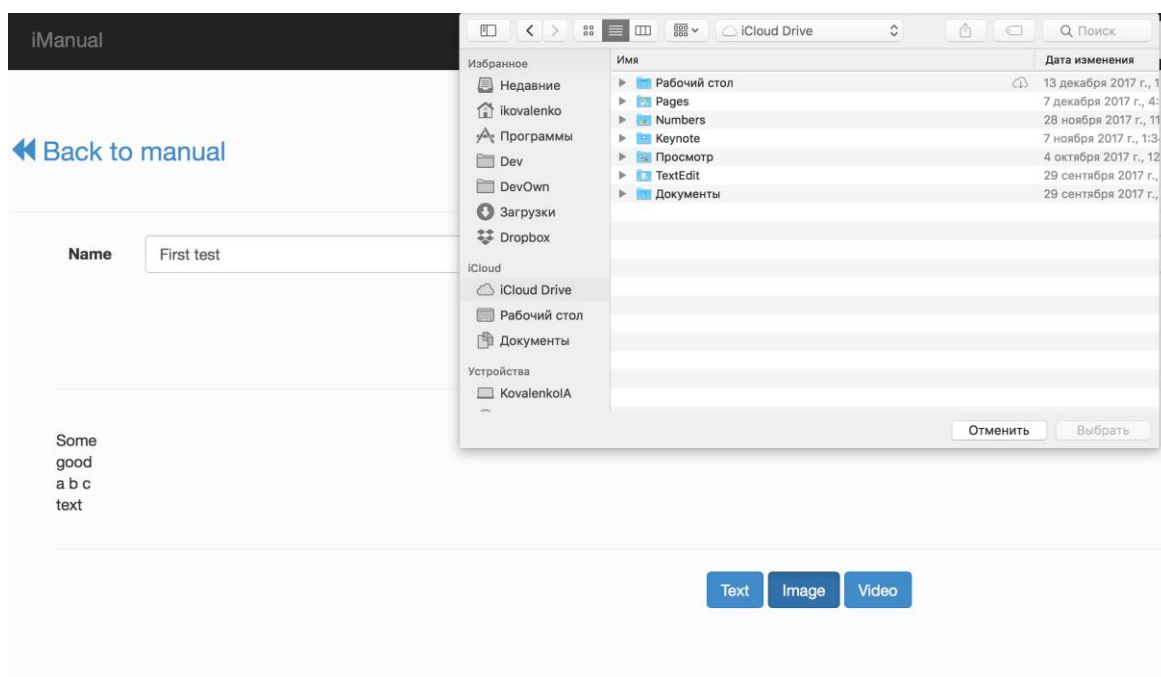


Рисунок 6.26 – Выбор картинки для загрузки в блок

После выборка картинки на компьютере появляется анимация загрузки, по окончании которой картинка загружена в блок. Теперь на странице можно наблюдать 2 блока: текст и картинка.

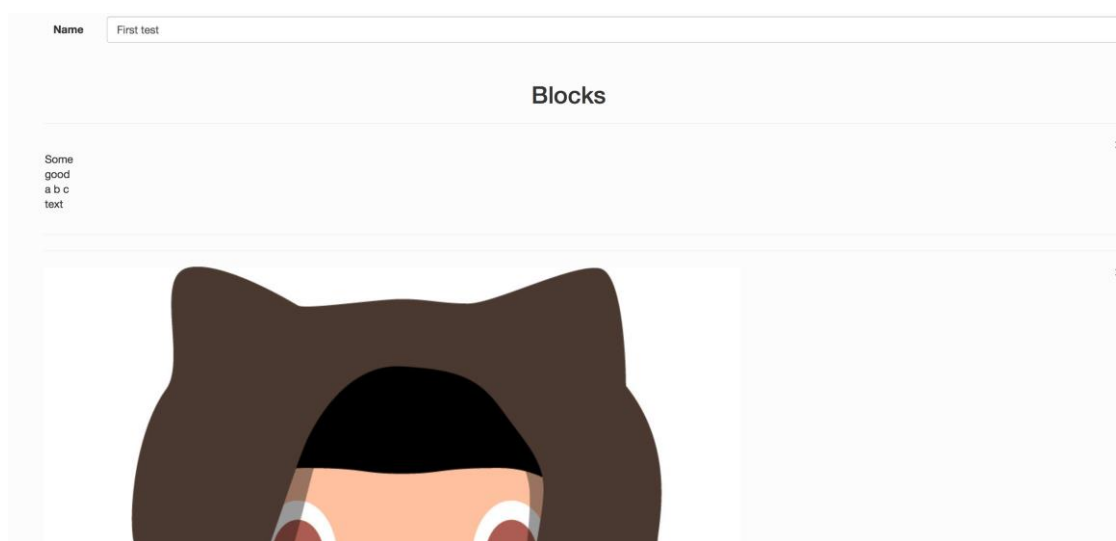


Рисунок 6.27 – Текст и картинка на странице шага

Также, возможно добавить видеоматериалы в качестве блока к шагу. Для этого необходимо выбрать кнопку Video, после чего указать ссылку на видео, размещающееся на видеохостинге YouTube, как показано на картинке 6.28.

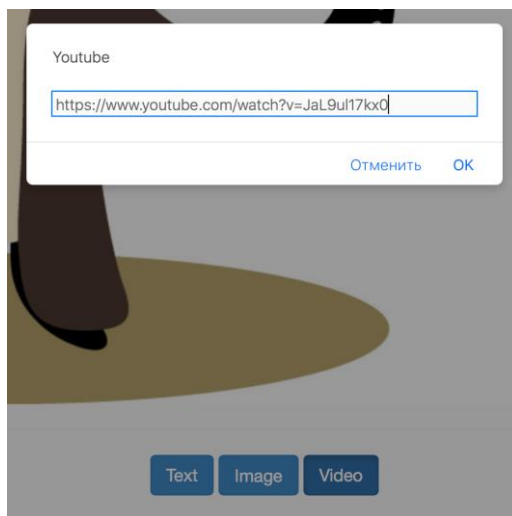


Рисунок 6.28 – Добавление видео блока

После загрузки ссылки на видео, оно появится в качестве одного из блоков шага, как можно видеть на картинке 6.29.

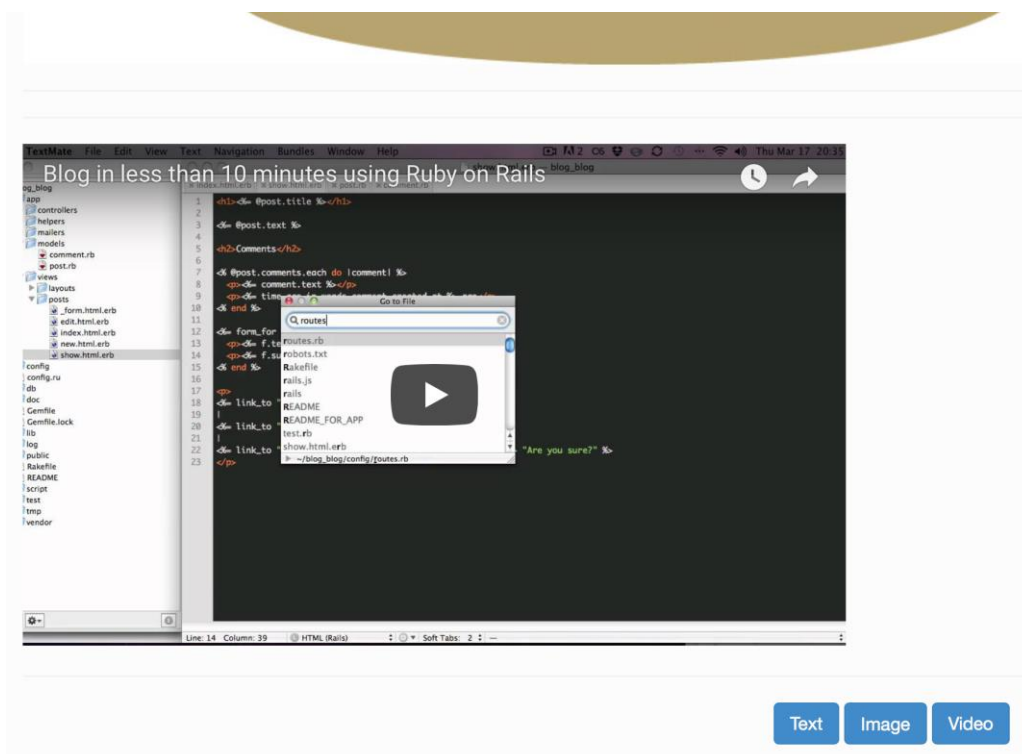


Рисунок 6.29 – Блок с видео

Развертывание приложения на реальном оборудовании:

Развернуть приложение на реальном хостинге можно при помощи автоматизированных средств, таких как heroku-cli, либо создать необходимую инфраструктуру, используя схему 6.30.

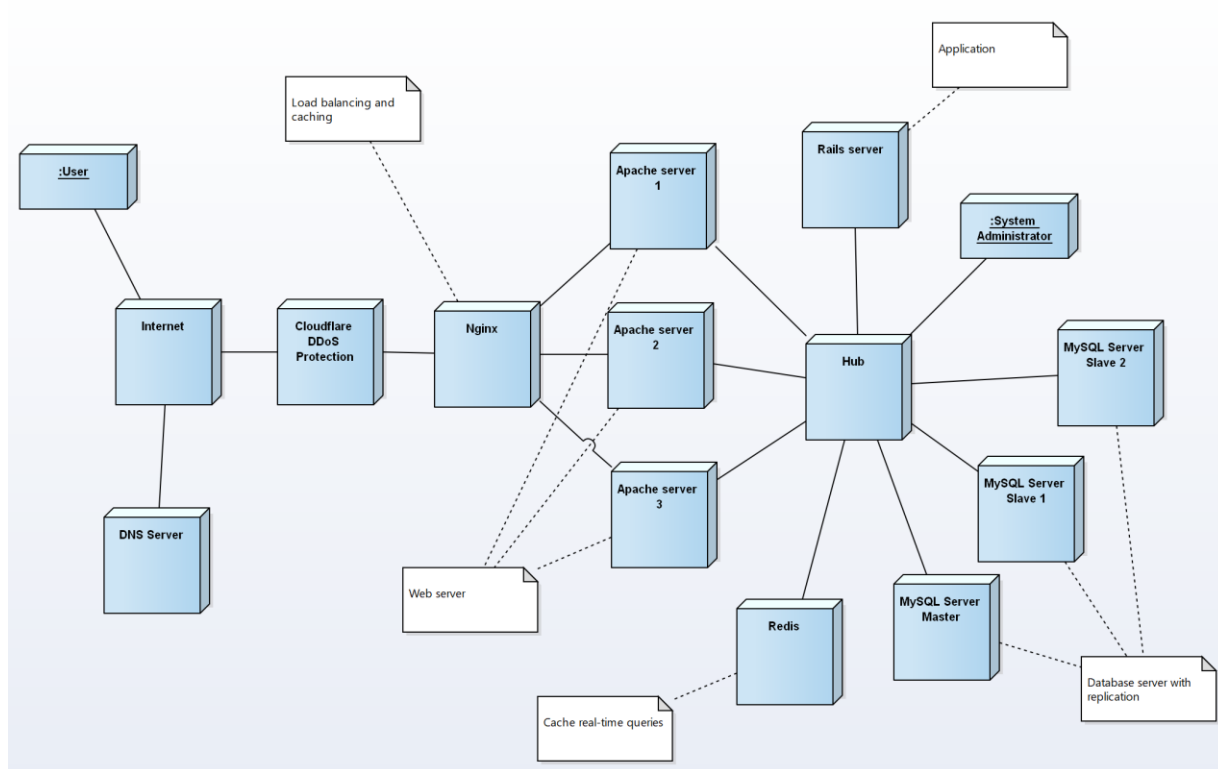


Рисунок 6.30 – Диаграмма Deployment

## ЗАКЛЮЧЕНИЕ

В ходе курсового проектирования было создано веб-приложение Каталог инструкций, позволяющее управлять инструкциями.

Разработанное приложение имеет следующие функции/возможности:

- создание, редактирование, удаление инструкций;
- возможность добавлять информацию в инструкции в виде текста, изображения либо видео;
- возможность проходить инструкции, а также смотреть свой прогресс для каждой конкретной инструкции;
- оставление отображающихся в реальном времени комментариев к инструкциям, проставление рейтинга инструкциям;
- возможность авторизации пользователей через социальные сети;
- редактирование, удаление профиля пользователя;
- возможность просмотра профиля пользователя;
- возможность смены языка пользовательского интерфейса.

В ходе работы фреймворк Rails зарекомендовал себя как отлично подходящий для современной гибкой разработки веб-приложений, имеющих поддержку всех современных технологий: coffeescript, javascript, erb, slim, websockets.

Так как созданное веб-приложение имеет REST-архитектуру, то дальнейшим шагом может стать создание соответствующих приложений для различных платформ (например, мобильных). По причине того, что в общем случае REST является очень простым интерфейсом управления информацией без использования каких-то дополнительных внутренних прослоек. Каждая единица информации однозначно определяется глобальным идентификатором, таким как URL. Каждая URL в свою очередь имеет строго заданный формат.

Еще одним из векторов дальнейшего развития является большая интеграция вебсокетов в виде ActionCable, представленного в Ruby On Rails фреймворке, которые позволяют создавать асинхронное, быстрое, с малыми накладными расходами на передачу данных по сети, приложение.

В настоящее время активно развиваются одностраничные приложения (англ. single page application, SPA) — это веб-приложение или веб-сайт, использующий единственный HTML-документ как оболочку для всех веб-



страниц и организующий взаимодействие с пользователем через динамически подгружаемые HTML, CSS, JavaScript, обычно посредством AJAX.

Одностраничные приложения напоминают родные (англ. native) приложения, с той лишь разницей, что исполняются в рамках браузера, а не в собственном процессе операционной системы.[15]

Основными элементами, используемыми при построении SPA, являются:

- фреймворки для JavaScript, в частности MVC и MVVM-фреймворки;
- роутинг (навигация между представлениями производится во фронтенде);
- шаблонизатор;
- HTML5;
- API для бэкенда, например, в стиле REST;
- Ajax.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Веб-приложение [Электронный ресурс]. – Режим доступа:  
<https://ru.wikipedia.org/wiki/Веб-приложение>
- [2] Инструкция [Электронный ресурс]. – Режим доступа:  
<https://ru.wikipedia.org/wiki/Инструкция>
- [3] Инструкции [Электронный ресурс]. – Режим доступа:  
<http://www.bibliotekar.ru/deloproizvodstvo-1/85.htm>
- [4] wikiHow [Электронный ресурс]. – Режим доступа:  
<https://ru.wikipedia.org/wiki/WikiHow>
- [5] Twitter Bootstrap Rails [Электронный ресурс]. – Режим доступа:  
<https://github.com/seyhunak/twitter-bootstrap-rails>
- [6] RedCarpet [Электронный ресурс]. – Режим доступа:  
<https://github.com/vmg/redcarpet>
- [7] Devise [Электронный ресурс]. – Режим доступа:  
<https://github.com/plataformatec/devise>
- [8] CanCanCan [Электронный ресурс]. – Режим доступа:  
<https://github.com/CanCanCommunity/cancancan>
- [9] Omniauth [Электронный ресурс]. – Режим доступа:  
<https://github.com/omniauth/omniauth>
- [10] RatyRate [Электронный ресурс]. – Режим доступа:  
<https://github.com/wazery/ratyrage>
- [11] Gravastic [Электронный ресурс]. – Режим доступа:  
<https://github.com/chrislloyd/gravtastic>
- [12] Search Cop [Электронный ресурс]. – Режим доступа:  
[https://github.com/mrkamel/search\\_cop](https://github.com/mrkamel/search_cop)
- [13] STI – одна таблица и много моделей [Электронный ресурс]. – Режим доступа: <https://habrahabr.ru/post/79630/>
- [14] Файлопровод Rails [Электронный ресурс]. – Режим доступа:  
<http://rusrails.ru/asset-pipeline>
- [15] Одностраничные приложения [Электронный ресурс]. – Режим доступа: [https://ru.wikipedia.org/wiki/Одностраничное\\_приложение](https://ru.wikipedia.org/wiki/Одностраничное_приложение)

## ПРИЛОЖЕНИЕ А – Исходный текст программы

### Текст класса ApplicationController

```
class ApplicationController < ActionController::Base
  protect_from_forgery
  before_action :set_locale
  before_action :store_current_location, unless: :devise_controller?
  after_action :set_csrf_cookie_for_ng

  rescue_from CanCan::AccessDenied do |exception|
    respond_to do |format|
      format.json { head :forbidden, content_type: 'text/html' }
      format.html { redirect_to main_app.root_url, alert: exception.message }
      format.js { head :forbidden, content_type: 'text/html' }
    end
  end

  def devise_mapping
    Devise.mappings[:user]
  end

  private

  def store_current_location
    store_location_for(:user, request.url)
  end

  def after_sign_out_path_for(resource)
    request.referrer || root_path
  end

  def set_csrf_cookie_for_ng
    cookies['XSRF-TOKEN'] = form_authenticity_token if protect_against_forgery?
  end

  def set_locale
    I18n.locale = params[:locale].blank? ? I18n.default_locale : params[:locale]
  end

  def default_url_options
    { locale: I18n.locale }
  end

  protected

  def verified_request?
    super || valid_authenticity_token?(session, request.headers['X-XSRF-TOKEN'])
  end
end
```

### Текст класса BlocksController

```
class BlocksController < ApplicationController
  before_action :set_block, only: [:show, :edit, :update, :destroy]
  load_resource :step
  load_and_authorize_resource :block, :through => :step

  # GET /blocks
  # GET /blocks.json
  def index
```

```

@blocks = Block.all
respond_to do |format|
  format.html
  format.json {
    render json: @blocks
  }
end
end

# GET /blocks/1
# GET /blocks/1.json
def show
  respond_to do |format|
    format.html
    format.json {
      render json: @block
    }
  end
end

# GET /blocks/new
def new
  @block = Block.new
end

# GET /blocks/1/edit
def edit
end

# POST /blocks
# POST /blocks.json
def create
  @block = Block.new(block_params)
  @block.type = params[:type].capitalize

  respond_to do |format|
    if @block.save
      format.html { redirect_to @block, notice: 'Block was successfully created.' }
      format.json { render json: @block }
    else
      format.html { render :new }
      format.json { render json: @block.errors, status: :unprocessable_entity }
    end
  end
end

# PATCH/PUT /blocks/1
# PATCH/PUT /blocks/1.json
def update
  respond_to do |format|
    if @block.update(block_params)
      format.html { redirect_to @block, notice: 'Block was successfully updated.' }
      format.json { render json: @block }
    end
  end
end

```

```

    else
      format.html { render :edit }
      format.json { render json: @block.errors, status: :unprocessable_entity }
    end
  end
end

# DELETE /blocks/1
# DELETE /blocks/1.json
def destroy
  @block.destroy
  respond_to do |format|
    format.html { redirect_to blocks_url, notice: 'Block was successfully destroyed.' }
    format.json { head :no_content }
  end
end

private

def set_block
  @block = Block.find(params[:id])
end

def block_params
  params.require(:block).permit(:content, :priority, :step_id, :type, :priority)
end
end

```

## Текст класса CategoriesController

```

class CategoriesController < ApplicationController
  before_action :set_category, only: [:show]

  def index
    respond_to do |format|
      format.json
    end
  end

  # GET /categories/1
  def show
    @manuals = @category.manuals.page(params[:page])
  end

  private

  def set_category
    @category = Category.find(params[:id])
  end
end

```

## Текст класса CommentsController

```

class CommentsController < ApplicationController
  before_action :set_comment, only: [:destroy]
  load_and_authorize_resource

```

```

# GET /manuals/:manual_id/new
def new
  @comment = Comment.new
end

# POST /manuals/:manual_id/comments
# POST /manuals/:manual_id/comments.json
def create
  @comment = Comment.new(comment_params)
  @comment.user = current_user
  @comment.save
end

# DELETE /manuals/:manual_id/comments/1
# DELETE /manuals/:manual_id/comments/1.json
def destroy
  @comment.destroy
  respond_to do |format|
    format.html { redirect_to manual_path(params[:manual_id]), notice: 'Comment was successfully destroyed.' }
    format.json { head :no_content }
  end
end

private

def set_comment
  @comment = Comment.find(params[:id])
end

def comment_params
  params.require(:comment).permit(:content, :manual_id)
end
end

```

## Текст класса ManualsController

```

class ManualsController < ApplicationController
  before_action :set_manual, only: [:show, :edit, :update, :destroy]
  before_action :set_comments, only: [:show]
  before_action :set_completed_steps, only: [:show]
  before_action :set_manuals, only: [:index]
  before_action :set_tags
  load_and_authorize_resource

  # GET /manuals
  # GET /manuals.json
  def index
    respond_to do |format|
      format.html
      format.js { render partial: 'manuals/more_manuals' }
      format.json { render json: @manuals }
    end
  end

  # GET /manuals/1
  # GET /manuals/1.json
  def show
    @comment = Comment.new

```

```

respond_to do |format|
  format.html
  format.json { render json: @manual, include: %i(steps tags) }
end
end

# GET /manuals/new
def new
  @manual = Manual.new
  @category_id = params[:category_id] || Category.first.id
end

# GET /manuals/1/edit
def edit
end

# POST /manuals
# POST /manuals.json
def create
  @manual = Manual.new(manual_params)
  @manual.user = current_user

  respond_to do |format|
    if @manual.save
      format.html { redirect_to edit_manual_path(@manual), notice: 'Manual was successfully created.' }
      format.json { render :show, status: :created, location: @manual }
    else
      format.html { redirect_to new_manual_path, :flash => { :error => @manual.errors.full_messages.join(', ') } }
      format.json { render json: @manual.errors, status: :unprocessable_entity }
    end
  end
end

# PATCH/PUT /manuals/1
# PATCH/PUT /manuals/1.json
def update
  @manual.tag_list = params[:tags].map { |tag| tag[:name] }
  respond_to do |format|
    if @manual.update(manual_params)
      format.html { redirect_to @manual, notice: 'Manual was successfully updated.' }
      format.json { render :show, status: :ok, location: @manual }
    else
      format.html { render :edit }
      format.json { render json: @manual.errors, status: :unprocessable_entity }
    end
  end
end
end

```

Обозначение					Наименование					Дополнительные сведения				
					<u>Текстовые документы</u>									
БГУИР КП 1–40 01 01 551005 012 ПЗ					Пояснительная записка					72 с.				
					<u>Графические документы</u>									
ГУИР 551005 012 СП					Схема связей между элементами базы данных					Формат А1				
										</				



