

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение высшего
образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет «Вычислительная техника» (наименование)	Кафедра «Системы автоматизированного проектирования» (наименование)
Направление подготовки	09.03.01 «Информатика и вычислительная техника» (код и наименование)
Профиль	«Системы автоматизированного проектирования» (наименование)

БАКАЛАВРСКАЯ РАБОТА
на тему

Разработка компонентов интегрированной библиотечной системы

Студент

АК 27.05.17
(подпись, дата)

Акимеев М.А.
(фамилия, инициалы)

Руководитель

Бур- 30.05.17
(подпись, дата)

Бурдakov И.И.
(фамилия, инициалы)

Нормоконтролёр

инф. каф. САПР ПГУ
(должность, место работы)

29.05.17
(подпись, дата)

Гонимов А.С.
(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

Березин
(подпись)

Березинский Х.М.
(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

(фамилия, инициалы)

Пенза 2017

Реферат

Пояснительная записка _ листа, _ рисунков, _ источников, _ приложение.

БИБЛИОТЕКА, ОНЛАЙН БРОНИРОВАНИЕ, RUBY, RUBY ON RAILS, WEB-ПРИЛОЖЕНИЕ, POSTGRESQL, NGINX, HTML5, CSS.

Объектом разработки является WEB-приложение для поиска книг в библиотеке и онлайн бронирования их.

Цель работы - разработать информационно-справочное WEB-приложение, которое позволит бронировать литературу в библиотеке онлайн.

В процессе разработки проводилось изучение принципов построения WEB-приложений, в частности технологии создания приложений с использованием фреймворка Ruby on Rails.

В результате выполнения дипломной работы создано WEB-приложение для поиска книг и последующего их резервирования в библиотеке.

Разработанный сервис следует применять для автоматизации работы библиотеки.

					ВКР 090301.03 17 81 01						
Изм.	Лист	№ докум.	Подпись	Дата							
Разраб.		Акинтеев М.А.			Разработка интегрированной библиотечной системы Пояснительная записка			Лит.	Лист	Листов	
Провер.		Бурукина И.П.								3	
Н. Контр.		Бождай А.С.						ФВТ, каф. САПР, группа 13BB2			
.											
Утверд.		Бершадский А.М.									

СОДЕРЖАНИЕ

ПЕРЧЕНЬ СОКРАЩЕНИЙ	6
ВВЕДЕНИЕ	6
1. ПОСТАНОВКА ЗАДАЧИ.....	9
2. ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ	10
2.1. Язык программирования Ruby.....	10
2.2. Фреймворк Ruby on Rails	11
2.3. Среда разработки RubyMine	12
2.4. SQLite3	12
2.5. PostgreSQL	13
2.6. Slim.....	14
2.7. Scss	14
2.8. Ubuntu.....	14
2.9. Proху-сервер.....	14
3. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ.....	16
3.1. Описание метода разработки.....	16
3.2. Предварительный этап	17
3.2.1. Установка Ruby.....	17
3.2.2. Установка RoR.....	18
3.2.3. Установка RubyMine	18
3.2.4. Создание каркаса приложения.....	18
3.2.5. Подключение необходимых библиотек	19
3.3. Проектирование системы.....	20
3.3.1. Разработка структуры БД.....	20
3.3.2. Разработка моделей данных.....	23
3.3.3. Проектирование контроллеров	24
3.3.4. Проектирование средств защиты информации.....	25
3.4. Программирование приложения.....	25
3.4.1. Создание модуля управления регистрацией	25
3.4.2. Обработка URL.....	29

3.4.3. Контроллеры	30
3.4.4. Файлы представления.....	33
3.4.5. Организация поиска книг	34
3.4.6. Реализация почтовой рассылки	38
3.4.7. Подключение формы обратной связи.....	41
3.5. Разработка дизайна приложения	41
3.5.1. Общий стиль	41
3.5.2. Страница показа информации о книге	42
3.5.3. Страница регистрации пользователя.....	44
3.5.4. Панель администратора.....	45
3.5.5. Адаптивная верстка	46
3.6. Тестирование	48
3.7. Размещение на сервере.....	49
4. ФУНКЦИОНИРОВАНИЕ ПРОГРАММЫ	50
4.1. Диаграмма использования	50
4.2. Описание схемы работы системы.....	50
4.3. Описание схемы взаимодействия программ.....	51
4.4. Описание схемы программы.....	53
5. ТЕХНОЛОГИЧЕСКАЯ ИНСТРУКЦИЯ ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ.....	55
5.1. Описание программы	55
5.2. Описание возможных ошибок и способы их устранения	56
6. ЗАКЛЮЧЕНИЕ	60
7. СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ.....	61
ПРИЛОЖЕНИЕ А. ЛИСТИНГ	63

ПЕРЧЕНЬ СОКРАЩЕНИЙ

ORM – (Object relative mapping) объектно-реляционное отображение;

IDE – (Integrated Development Environment) интегрированная среда разработки;

ROR – (Ruby on Rails) фреймворк для создания web-приложений на языке ruby;

CRUD – (Create Read Update Delete) создать прочитать обновить удалить;

REST – (Representational State Transfer) архитектурный стиль взаимодействия компонентов распределённого приложения в сети;

СУБД – система управления базой данных

БД – база данных.

MVC – (Model-View-Controller) модель-представление-контроллер;

ОС – операционная система;

ПО – программное обеспечение;

ПГУ – Пензенский государственный университет;

ЕСПД – единая система программной документации;

					ВКР 090301.03 17 81	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

ВВЕДЕНИЕ

Разработанная программа представляет из себя систему электронной библиотеки, в которой пользователь может не только выбрать из каталога интересующую его книгу, но и узнать: свободна она или занята. Если книга свободна, пользователь может ее зарезервировать, в противном случае – встать в очередь.

Одним из важных направлений развития современного общества является его глобальная информатизация.

На фоне интенсивной информатизации всех сфер жизни и деятельности общества вопросы использования информационно-коммуникационных технологий в работе библиотек приобретает особую значимость.

От работы библиотекарей ждут умения работать с электронными документами, с современной техникой, владения компьютерной грамотностью, навыками аналитической и проектной работы.

Все это выводит на первый план следующие задачи:

- накопление, обработка, распространение информационных ресурсов и материалов в различных форматах;
- внедрение автоматизированных информационно-поисковых систем, позволяющих оперативно и полно находить необходимую информацию;
- квалифицированные консультации и рекомендации для пользователей;
- обучение пользователей поисково-аналитической деятельности.

Исходя из этих задач, можно выделить главные направления интеграции информационно-коммуникационных технологий в деятельности библиотек:

- расширение информационного пространства библиотеки;
- автоматизация библиотечных процессов;

- использование ИКТ для повышения эффективности педагогической деятельности библиотекарей.

Таким образом, разработка данного программного продукта является актуальной задачей, способной решить достаточно важную проблему.

					<i>ВКР 090301.03 17 81</i>	<i>Лист</i>
						8
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		

1. ПОСТАНОВКА ЗАДАЧИ

Задачей дипломного проектирования является написание и отладка электронной библиотеки, в которой пользователь должен иметь возможность поиска и бронирования книг.

Система должна быть представлена WEB-сервисом, в котором сотрудники библиотеки имеют возможность добавления, изменения и удаления информации о книгах, управлять очередью бронирования книг, и выдавать книги согласно ей. Пользователь должен иметь возможность просмотра всех имеющихся книг в библиотеке, их поиска и бронирования. Также пользователь должен иметь возможность узнать свободна ли книга, в противном случае он может узнать дату, когда книга освободится.

В программе должна быть панель администратора, в которой реализован функционал, доступный только для сотрудников библиотеки. К просмотру книг библиотеки должны допускаться все пользователи, а узнать свободна ли книга или зарезервировать ее могут только зарегистрированные пользователи. Также немаловажной функцией электронного каталога является учет библиотечных фондов.

При входе в систему, посетитель должен видеть список всех книг с возможностью поиска необходимой книги из всего библиотечного фонда, а также иметь возможность просмотреть подробную информацию о них.

					<i>ВКР 090301.03 17 81</i>	<i>Лист</i>
						9
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		

2. ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ

2.1. Язык программирования Ruby

Ruby (англ. *Ruby* — рубин) — динамический, рефлексивный, интерпретируемый высокоуровневый язык программирования для быстрого и удобного объектно-ориентированного программирования. Язык обладает независимой от операционной системы реализацией многопоточности, строгой динамической типизацией, сборщиком мусора и многими другими возможностями. По особенностям синтаксиса он близок к языкам Perl и Eiffel, по объектно-ориентированному подходу — к Smalltalk. Также некоторые черты языка взяты из Python, Lisp, Dylan и Клу [10].

К возможностям данного языка программирования можно отнести:

- Имеет лаконичный и простой синтаксис, частично разработанный под влиянием Ада, Eiffel и Python.
- Позволяет обрабатывать исключения в стиле Java, Python, C#.
- Позволяет переопределять операторы, которые на самом деле являются методами.
- Полностью объектно-ориентированный язык программирования. Все данные в Ruby являются объектами в понимании Smalltalk. Единственное исключение — управляющие конструкции, которые в Ruby, в отличие от Smalltalk, не являются объектами.
- Не поддерживает множественное наследование, но вместо него может использоваться концепция «примесей», основанная в данном языке на механизме модулей.
- Содержит автоматический сборщик мусора. Он работает для всех объектов Ruby, в том числе для внешних библиотек.
- Поддерживает замыкания с полной привязкой к переменным.

					ВКР 090301.03 17 81	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

- Поддерживает блоки кода (код заключается в { ... } или do ... end).
- Целые переменные в Ruby автоматически конвертируются между типами Fixnum (32-разрядные) и Bignum (больше 32 разрядов) в зависимости от их значения, что позволяет производить целочисленные математические расчёты со сколь угодно большой точностью.
- Не требует предварительного объявления переменных. Язык использует простые соглашения для обозначения области видимости.
- В Ruby непосредственно в языке реализованы многие шаблоны проектирования.
- Может динамически загружать расширения, если это позволяет операционная система.
- Имеет независимую от ОС поддержку невытесняющей многопоточности.
- Перенесён на множество платформ.

2.2. Фреймворк Ruby on Rails

Для разработки был выбран фреймворк Ruby on Rails.

Ruby on Rails (RoR) — фреймворк, 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.

Причины выбора:

1. Большая распространенность. Существует большое количество библиотек и готовых решений, позволяющих значительно ускорить процесс разработки. Также на stackoverflow.com можно найти ответ на практически любой вопрос, возникающий при разработке.

					ВКР 090301.03 17 81	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

2. Основанность на MVC. Данный шаблон проектирования позволяет разграничить объекты, отвечающие за визуальное отображение, обработку web запросов и реализацию модели данных.
3. Поддержка ORM. Каждая таблица в БД может автоматически быть преобразована в класс, а каждая запись в экземпляр класса. Данный функционал реализует стандартная библиотека ActiveRecord. Также все классы имеют встроенные функции работы с БД
4. Язык программирования Ruby. На данный момент язык считается наиболее объектно-ориентированным. Множество специфичных операторов и синтаксических конструкций позволяют сократить размер кода, а также вероятность появления ошибок в коде.

2.3. Среда разработки RubyMine

На сегодняшний день RubyMine самая распространенная IDE для разработки с использованием фреймворка ROR. На втором месте по популярности стоит связка текстовый редактор – консоль.

Причины выбора данной IDE:

1. Удобный пользовательский интерфейс;
2. Уникальные средства создания и оптимизации приложений;
3. Своевременные обновления, позволяющие IDE не отставать от быстрого развития фреймворка.

2.4. SQLite3

При разработке приложения использовалась СУБД SQLite3. Ruby on Rails работает с ней «из коробки», она быстроедействительна и легко переносится с компьютера на компьютер. Так же фреймворк абстрагируется от используемой базы данных и не имеет значения какую БД ты используешь при разработке, а какая используется в релизе.

					<i>ВКР 090301.03 17 81</i>	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

2.5. PostgreSQL

Для размещения приложения на сервере было решено выбрать СУБД PostgreSQL. PostgreSQL – объектно-ориентированная СУБД с открытым исходным кодом. Основной причиной выбора послужила высокая безопасность и защищенность. Данная СУБД широко распространена и Ruby on Rails легко конфигурируется для работы с ней. На «Википедии» [12] также указаны дополнительные возможности выбранной СУБД:

- Соблюдение принципов ACID;
- Соответствие стандартам ANSI SQL-92 и SQL-99;
- Поддержка запросов с OUTER JOIN, UNION, UNION ALL, EXCEPT, INTERSECT и подзапросов;
- Последовательности;
- Контроль целостности;
- Репликация;
- Общие табличные выражения и рекурсивные запросы;
- Аналитические функции;
- Поддержка Юникода (UTF-8);
- Поддержка регулярных выражений в стиле Perl;
- Встроенная поддержка SSL, SELinux и Kerberos;
- Протокол разделяемых блокировок;
- Подгружаемые расширения, поддерживающие SHA1, MD5, XML;
- Расширения для написания сложных выборок, отчётов и т.д. (API открыт);
- Средства для генерации совместимого с другими системами SQL-кода и импорта из других систем;
- Автономные блоки на доступных языках, а не только SQL.

2.6. Slim

Для создания HTML кода было решено выбрать шаблонизатор Slim. Разметка, написанная с его использованием, легко читается и понятна программисту, значительно компактнее HTML и быстро транслируется. При показе страницы, Ruby on Rails кеширует страницу, следовательно не будет проблем с производительностью [3].

2.7. Scss

Scss — это метаязык на основе CSS, предназначенный для увеличения уровня абстракции CSS кода и упрощения файлов каскадных таблиц стилей.

Причиной выбора данного препроцессора стилей послужила сложность создания иерархической структуры при использовании обычного CSS [3].

2.8. Ubuntu

Сайт было решено размещать на сервере с ОС Ubuntu. Ruby on Rails работают лучше всего под данной операционной системой. Некоторые библиотеки зависят от операционной системы и не работают под Windows. Так же сервера с данной операционной системой значительно дешевле [5].

2.9. Proxy-сервер

При размещении сайта на сервере было решено использовать два вида серверов: Backend и Frontend. Frontend-сервер получает web-запрос от сети и перенаправляет его на Backend-сервер, который в свою очередь обрабатывает его и возвращает ответ. В качестве Frontend-сервера был выбран Nginx. Отличительными особенностями которого являются легкая конфигурируемость, быстрое действие и, в отличие от Apache, он не создает дополнительных нитей для каждого запроса. Вопросы в выборе Backend-

					<i>ВКР 090301.03 17 81</i>	Лист
						14
Изм.	Лист	№ докум.	Подпись	Дата		

сервера не стояло. Для целей создаваемого приложения подходил любой. Выбран был Рунта, так как в отличие от остальных поддерживает web-сокеты «из коробки», незначительно быстрее остальных и имеет возможность распараллеливать запросы, а не выполнять поочередно (что важно только для высоконагруженных приложений).

					ВКР 090301.03 17 81	Лист
						15
Изм.	Лист	№ докум.	Подпись	Дата		

3. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ

3.1. Описание метода разработки

Проектирование системы осуществлялось методом нисходящего программирования. При этом методе проектирование осуществляется сверху вниз. Сначала пишется головной модуль, а нижележащие функции обработки данных заменяются заглушками. Далее отлаживается головной модуль и заглушки заменяются конкретными модулями для обработки данных. Этот процесс повторяется до тех пор, пока не будут отлажены модули самых низших уровней. Преимуществом такого подхода при написании программ является то, что процесс написания программы совпадает с процессом отладки. При разработке компилятора процесс компиляции был разбит на несколько подзадач, таких как обработка текста, дешифрация команд, генерация кода команд, обработка данных. Такая декомпозиция задачи на отдельные подзадачи облегчает процесс проектирования и соответствует идеологии проектирования методом нисходящего программирования.

Работу по созданию приложения, условно можно разбить на несколько этапов:

1. Предварительный этап;
2. Разработка архитектуры проекта;
3. Программирование;
4. Разработка дизайна приложения;
5. Тестирование;
6. Размещение;
7. Администрирование (поддержка).

3.2. Предварительный этап

На предварительном этапе устанавливаются все необходимые программные средства и создается каркас приложения. Также подключаются все необходимые библиотеки. Разработка велась под ОС Ubuntu 16.04, все последующие действия применимы к ней.

3.2.1. Установка Ruby

Установить Ruby на компьютер можно разными способами, правильным считается способ, использующий RVM (Ruby Version Manager). Данная программа позволяет использовать одновременно разные версии языка программирования, легко переключаться между ними, а также загружать другие версии. Для установки менеджера версий необходимо выполнить команду:

```
\curl -sSL https://get.rvm.io | bash -s stable
```

Ruby использует несколько библиотек и их необходимо установить предварительно:

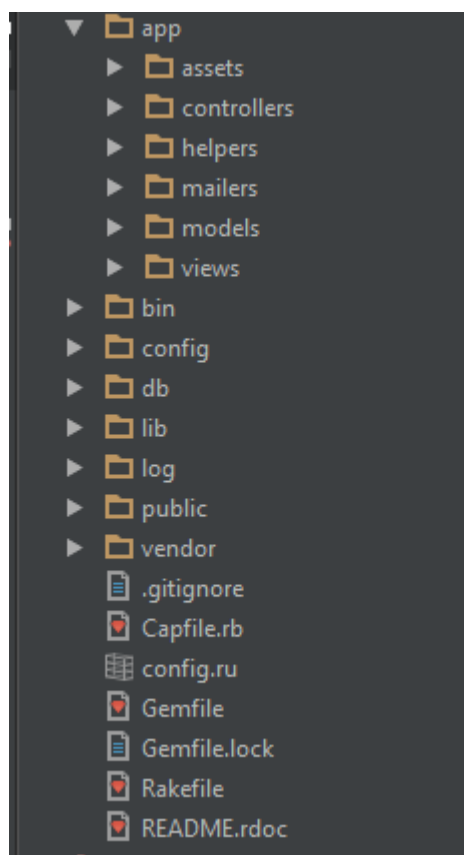
```
sudo apt-get install git-core curl zlib1g-dev build-essential libssl-dev libreadline-dev libyaml-dev libsqlite3-dev sqlite3 libxml2-dev libxslt1-dev libcurl4-openssl-dev python-software-properties libffi-dev
```

Затем нужно установить сам язык Ruby с использованием команды, где цифра обозначает версию языка:

```
rvm install 2.3.0  
rvm use 2.3.0 --default  
ruby -v
```

После установки языка программирования необходимо установить Bundler, являющийся незаменимой частью Ruby on Rails.

```
gem install bundler
```



					<i>BKP 090301.03 17 81</i>	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

3.2.5. Подключение необходимых библиотек

Подключение библиотек в Rails-приложениях просто. Для этого необходимо прописать название библиотеки в Gemfile и вызвать команду «bundle install».

Список необходимых для данного приложения библиотек:

```
source 'https://rubygems.org'
gem 'rails'
gem 'uglifier', '>= 1.3.0'
# Templates
gem 'sass-rails'
gem 'slim-rails'
gem 'materialize-sass'
gem 'font-awesome-rails'
gem 'jquery-rails'
gem 'turbolinks'
gem 'jbuilder', '~> 2.0'
gem 'russian'
gem 'activeadmin', github: 'activeadmin'
gem 'devise'
gem 'simple_form'
gem 'puma'
group :production, :test do
  gem 'rails_12factor', '0.0.2'
  gem 'pg'
end
gem 'ckeditor'
gem 'carrierwave'
gem 'kaminari'
gem 'sprockets'
group :development, :test do
  gem 'byebug'
  gem 'web-console', '~> 2.0'
  gem 'spring'
  gem 'quiet_assets'
  gem 'better_errors'
  gem 'binding_of_caller'
  gem 'sqlite3'
  gem 'capistrano', '~> 3.4.0'
  gem 'capistrano-rvm'
  gem 'capistrano3-puma'
  gem 'capistrano-rails', '~> 1.1', require: false
  gem 'capistrano-bundler', '~> 1.1', require: false
end
```

					ВКР 090301.03 17 81	Лист
Изм.	Лист	№ докум.	Подпись	Дата		19

3.3. Проектирование системы

3.3.1. Разработка структуры БД

Структура базы данных формируется при помощи миграций. В миграции указываются изменения, которые необходимо произвести. Миграции модно накатывать и откатывать. По файлам миграции создается файл схемы базы данных.

Модель пользователя создается в первую очередь. Скрипт, представленный ниже создает таблицу в базе данных со всеми необходимыми данными:

```
class AddDeviseToUsers < ActiveRecord::Migration
  def self.up
    change_table(:users) do |t|
      ## Database authenticatable
      t.string :email,              null: false, default: ''
      t.string :encrypted_password, null: false, default: ''

      ## Recoverable
      t.string   :reset_password_token
      t.datetime :reset_password_sent_at

      ## Rememberable
      t.datetime :remember_created_at

      ## Trackable
      t.integer  :sign_in_count, default: 0, null: false
      t.datetime :current_sign_in_at
      t.datetime :last_sign_in_at
      t.string   :current_sign_in_ip
      t.string   :last_sign_in_ip
      t.string   :name
      t.string   :telephone
      t.boolean  :admin, null: false, default: false
    end

    add_index :users, :email,                unique: true
    add_index :users, :reset_password_token, unique: true
  end
end
```

					ВКР 090301.03 17 81	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

Для пользователя создаются поля с информацией об адресе электронной почты, хешем пароля, датой регистрации, номером телефона, именем и дополнительной служебной информацией, необходимой для восстановления и защиты аккаунта. Для того, чтобы отделить пользователя от администратора был добавлен специальный флаг. По умолчанию, он выставлен в отрицательное значение.

Затем создается таблица с информацией о категориях книг. Таблица называется «Categories». Данная таблица создается раньше таблицы с информацией о книгах для того, чтобы можно было сделать ссылку на таблицу. Каждая категория имеет: идентификатор (создается неявно), имя и информацию о дате создания и последнего изменения.

```
class CreateCategories < ActiveRecord::Migration[ 5.0]
  def change
    create_table :categoryes do |t|
      t.string :name, null: false
      t.timestamps
    end
  end
end
```

Далее создается таблица с информацией о книге. Это самая главная таблица в приложении. В ней хранится информация о книжном фонде библиотеки. Каждая запись состоит из: имени книги, автора произведения, категории книги, дате издания, издательстве. Также для каждой записи создается поле описание, в котором можно разместить любую информацию о книге в свободном формате.

```
class CreateBooks < ActiveRecord::Migration[ 5.0]
  def change
    create_table :books do |t|
      t.references :category, null: false
      t.string :name, null: false
      t.string :author, null: false
      t.string :year, null: false
      t.string :publishing, null: false
      t.string :description, null: false
      t.timestamps
    end
  end
end
```

```
end
end
```

Информация о записи в очередь на книги хранится в таблице «Relationship». Эта таблица реализует отношение многие ко многим между читателем и книгой. Дополнительным полем является дата возврата книги. Данная информация будет использоваться для создания очереди и информирования участником бизнес-процесса. Датой, когда книгу читатель взял, является максимальная дата между датой создания записи в таблице и датой, когда книга освободится.

```
class CreateRelationships < ActiveRecord::Migration[ 5.0]
  def change
    create_table :relationships do |t|
      t.references :user, null: false
      t.references :book, null: false
      t.timestamp :release_date, null: false
      t.timestamps
    end
  end
end
```

Ограничение, что одновременно книгу не может зарезервировать на чтение больше одного человека будет реализовано программно, а не на уровне базы данных. Если в библиотеке больше одного экземпляра книги, то необходимо создать по одной записи для каждого экземпляра, так как книги имеют свой уникальный регистрационный номер и учитываются отдельно. Возможность реализации с добавлением поля «количество экземпляров» необходимо будет рассмотреть после первых испытаний системы.

Также создадим таблицу «Articles», в которой будут храниться объявления и статьи, написанные для информирования посетителей библиотеки.

Также создадим таблицу «Notifications», в которой будут храниться различные уведомления: о том, что кто-то встал в очередь на книгу, подошла очередь на чтение книги или человек вернул книгу досрочно.

Для каждой таблицы скрыто будет добавлен идентификатор, дата создания и последнего изменения записи.

					ВКР 090301.03 17 81	Лист
						22
Изм.	Лист	№ докум.	Подпись	Дата		

После создания миграций, необходимо выполнить команду «rails db:migrate»

3.3.2. Разработка моделей данных

К каждой созданной таблице необходимо создать модель, установить ограничения по данным, отношения между моделями.

В разработке данной квалификационной работы используется принцип объектно-реляционного отображения. Он подразумевает соотношение класса с таблицей в базе данных, где запись — это экземпляр класса, а поле — это свойство класса. Для реализации используется стандартная библиотека фреймворка Ruby on Rails – ActiveRecord. Данная библиотека инкапсулирует SQL-запросы и предоставляет их в форме методов и свойств класса. Библиотека имеет встроенные механизмы валидации данных и создания реляционных отношений между классами.

```
class User < ApplicationRecord
  has_many :relationships

  devise :database_authenticatable, :registerable,
:recoverable, :rememberable, :trackable, :validatable
  validates_presence_of :email, :name, :encrypted_password,
:telephone
  validates_uniqueness_of :email
  VALID_MOBILE_PHONE_REGEX = /\+?\d+([\ \(\s\ -]?\d+[\ \)\s\ -
]?[\d\s\ -]+)?/
  validates :mobile_phone, format: {with:
VALID_MOBILE_PHONE_REGEX, allow_blank: false }
End
```

Данный класс реализует логику, связанную с моделью записи на очередь чтения книги. Дополнительно добавлен метод информирования библиотекаря о том, что пользователь библиотеки хочет взять почитать книги на электронную почту.

```
class Relationship < ActiveRecord::Base
  belongs_to :user
  belongs_to :book
```

```

after_save :send_notification
validates_presence_of :user, :book, :release_date
validate :validate_release_date

private

def validate_release_date
  #проверим не попадает ли дата возврата в тот момент,
  когда книга занята уже кем-то.
  if Relationship.where('created_at < ? AND release_date >
?', DateTime.now, self.release_date).count > 0 &&
self.release_date > DateTime.now
    errors.add(:release_date, 'вы не сможете вернуть книгу
в этот момент, так как вы ее еще не успеете взять почитать')
  end
end

def send_notification
  s = 'Читатель встал на очередь к книге: ' +
self.book.name
  Notification.create!(user: task.user, text: s,
notifiable: self)
  NotificationMailer.new_relationship(self).deliver
end
end
end

```

Остальные классы аналогичны представленным выше. В них продублирована валидация данных, наложенная ограничениями в базе данных.

3.3.3. Проектирование контроллеров

Дальше необходимо создать контроллеры для моделей. Контроллеры описывают все действия, которые можно произвести с моделью. В Ruby on Rails используется модель CRUD, согласно которой необходимо реализовать методы:

1. Index – просмотр всех объектов
2. Show – показ одного объекта;

3. New – создание нового экземпляра объекта;
4. Create – сохранение объекта в БД;
5. Update – обновление записи в базе данных;
6. Edit – редактирование записи.

3.3.4. Проектирование средств защиты информации

Для защиты от межбраузерных запросов к каждой странице будет добавляться токен, валидный только в рамках данного сайта. В Ruby on Rails данный функционал реализуется добавлением в контроллер приложения записи:

```
# Prevent CSRF attacks by raising an exception.
protect_from_forgery with: :exception
```

Для защиты от отправки дополнительных параметров или выполнении SQL-инъекции, будут проверяться все параметры, получаемые из интернета и отфильтровываться лишние.

Для разграничения уровней доступа будут использована библиотека «can»», которая позволяет в удобной форме расписать возможности каждого типа пользователей. Так же к сервису будет добавлен модуль аутентификации и регистрации «devise». Указанные библиотеки имеют открытый исходный код, что позволяет убедиться в отсутствии «бекдоров», так же библиотеки полностью покрыты тестами, что исключает возможность появления уязвимостей.

3.4. Программирование приложения

3.4.1. Создание модуля управления регистрацией

Для регистрации было решено использовать библиотеку «devise». Для конфигурации библиотеки, после ее установки, необходимо выполнить

					ВКР 090301.03 17 81	Лист
						25
Изм.	Лист	№ докум.	Подпись	Дата		

команду «rails g devise:install». Для того, чтобы она создала файлы разметки, необходимо выполнить команду «rails g devise:views»

Так же решено было позволить пользователям регистрироваться через социальные сети. Выбраны были Facebook, ВКонтакте и Одноклассники. Для этого в каждой социальной сети была произведена регистрация приложения, после чего были получены идентификаторы приложений, а также ключи доступа к API. После этого был создан файл конфигурации, представленный ниже.

```
Devise.setup do |config|
  require 'devise/orm/active_record'
  config.secret_key = Rails.env[:devise_key]
  config.mailer_sender = 'diplom.library@yandex.ru'
  config.mailer = ApplicationMailer
  #config.mailer = Devise::Mailer
  config.authentication_keys = [ :email ]
  config.case_insensitive_keys = [ :email ]
  config.strip_whitespace_keys = [ :email ]
  config.skip_session_storage = [ :http_auth ]
  config.stretches = Rails.env.test? ? 1 : 10
  config.allow_unconfirmed_access_for = 2.days
  config.reconfirmable = true
  config.password_length = 4..128
  config.reset_password_within = 6.hours
  config.scoped_views = true
  config.sign_out_via = :delete
  require 'omniauth-facebook'
  config.omniauth :facebook, '483193208507051',
    Rails.env[:fb_key] ,
    scope: 'email', info_fields: 'email, name, birthday,
gender, location, first_name, last_name',
    provider_ignores_state: true, display: 'page', image_size:
    'large', secure_image_url: true

  require 'omniauth-vkontakte'
  config.omniauth :vkontakte, '5022257', Rails.env[:vk_key] ,
    scope: 'email', display: 'page'

  require 'omniauth-odnoklassniki'
  config.omniauth :odnoklassniki, '1150420480',
    Rails.env[:ok_key] , :public_key => 'CBAHADJFEBABABABA',
```

					ВКР 090301.03 17 81	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

```

        :scope => 'GET_EMAIL, VALUABLE_ACCESS',
provider: 'odnoklassniki'
end

```

Контроллеры работы с регистрацией уже реализованы в библиотеке, в приложении необходимо только реализовать обработку возвращаемых значений с сервера. Для этого в модель User были добавлены следующие методы: см. Приложение А. Листинг. «Контроллеры».

Для реализации панели администратора использовалась библиотека «activeadmin». Библиотека проста в использовании и для ее настройки необходимо заполнить несколько файлов с описанием.

Когда администратор сервиса заходит на сайт, ему сразу отображается информация о статистике и положении дел в библиотеке. Для реализации данного функционала был создан файл «dashboard.rb» с содержимым:

```

ActiveAdmin.register_page 'Dashboard' do
  menu priority: 1, label: proc{
I18n.t('active_admin.dashboard') }
  content title: proc{ I18n.t('active_admin.dashboard') } do
    columns do
      column do
        panel 'Сводка' do
          div "Всего пользователей: #{ User.all.count} "
          div "Пользователей в данный момент на сайте:
#{ User.online_count} "
          div "Всего книг: #{ Book.count} "
          div "Книг занято:
#{ Relationship.where('release_date ? >',
DateTime.now).distinct.pluck(:book_id)} "
        end
      end
    end
  end
end

```

Для отображения статистики в графическом виде, использовалась библиотека «Google charts». Для использования в фреймворка RoR, была разработана библиотека «Chartkick», являющаяся оберткой вокруг Java Script кодом для удобного инстанцирования в приложение. Данный график позволяет отобразить моменты возврата книг.

```

columns do

```

					ВКР 090301.03 17 81	Лист
						27
Изм.	Лист	№ докум.	Подпись	Дата		

```

        column do
          div do
            text_node line_chart
Relationship.group_by_day(:release_date).count, ytitle:
'DВозвратов в день'
          end
        end
      end
    end
  end
end

```

Администратор имеет доступ к редактированию всей информации на сайте. Он может создавать, удалять, изменять информацию о книжном фонде библиотеки. Дополнительно он может не только просматривать очередь, но и управлять ей.

Все файлы конфигураций состоят из трех необязательных блоков: конфигурирования списка, конфигурирование показа и конфигурирование редактирования. Так же имеются дополнительные блоки, но они не использовались в разработке приложения.

```

ActiveAdmin.register Book do
  menu label: 'Книги'
  permit: :category :name :author :year :publishing,
:description

```

```

index do
  selectable_column
  id_column
  column 'Категория' :category
  column 'Название' :name
  column 'Автор' :author
  actions
end

```

```

show do
  default_main_content do
    row :category do |c|
      c.name
    end
    row :name
    row :author
    row :year
    row :publishing
    row :description

```

					ВКР 090301.03 17 81	Лист
						28
Изм.	Лист	№ докум.	Подпись	Дата		

```

end

form do |f|
  categories = Category.all.map { |cat| [cat.name,
cat.id]}
  f.inputs 'Исполнитель' do
    f.input :name, label: 'Название'
    f.input :author, label: 'Автор'
    f.input :year, label: 'Год издания'
    f.input :publisher, label: 'Издательство'
    f.input :categories, as: :select, label: 'Категория',
      collection: categories, include_blank: false
    f.input :description, as: :ckeditor, label: 'Описание'
  f.actions
  end
end
end

```

В данном файле конфигурируется показ списка книг, форма создания или изменения информации о книге, а также показ подробной информации о книге. Для ввода информации о книге используется текстовый редактор «ckeditor», который позволяет добавить форматирование к тексту.

Аналогично конфигурации административной панели для работы с таблицей «Book», конфигурируется управление остальными таблицами.

3.4.2. Обработка URL

Для обработки различных URL и перенаправления их методам контроллера в ROR имеется файл «Routes». Для работы с моделями, реализующими методы CRUD, существует метод «resources» с указанием имени класса, для которого необходимо реализовать пути. Данный метод можно расширить указав, какие методы не реализует контроллер модели, а какие реализует дополнительно [7].

Метод «root» описывает куда направлять запрос без указания пути. Первым параметром идет имя контроллера, вторым через символ «#» имя метода.

Методы контроллера, которые не относятся к моделям, описываются при помощи метода «match», где первым параметром идет URL, вторым метод контроллера, аналогично методу «root», тип запроса и псевдоним к данному пути.

```
Rails.application.routes.draw do
  resources :book
  resources :relationship, only: [:create, :destroy]
  root 'static_pages#landing'

  namespace :users do
    get 'omniauth_callbacks/facebook'
    get 'omniauth_callbacks/vkontakte'
    get 'omniauth_callbacks/odnoklassniki'
  end

  ActiveAdmin.routes(self)
  devise_for :users, controllers: { omniauth_callbacks:
'users/omniauth_callbacks'}
  match '/about', to:'static_pages#about', via: :get, as:
'about'
end
```

3.4.3. Контроллеры

Главный контроллер приложения был расширен дополнительным функционалом. В него приходят все запросы до того, как они будут обработаны требуемым.

см. Приложение

Для контроллера модели Vacansy были реализованы все методы модели CRUD:

1. Index – подготавливает к выводу список всех номеров в отеле;
2. Show – подготавливает к отображению номер с информацией о занятости по датам;
3. Show – подготавливает к отображению номер с информацией о занятости по датам;
4. New – создает экземпляр вакансии и строит по нему форму ввода информации;

					ВКР 090301.03 17 81	Лист
						30
Изм.	Лист	№ докум.	Подпись	Дата		

5. Create – получив данные от клиента, метод проверяет их валидность и создает новый экземпляр вакансии с последующей записью его в базу данных;
6. Update – метод обновляет уже имеющуюся вакансию;
7. Delete – в данном методе происходит удаление вакансии.

Аналогичные контроллеры реализованы и для других моделей: Worker, Employer и Standart. Исключением являются модели User и Relationship. Для модели User контроллер реализован в библиотеке Devise. В разработанном приложении данный контроллер только расширен методами аутентификации через социальные сети.

```
class Users::OmniauthCallbacksController <
  Devise::OmniauthCallbacksController
    def facebook
      @user =
User.find_for_facebook_oauth(request.env[ 'omniauth.auth' ] ,
current_user)
      after_login(@user)
    end

    def vkontakte
      @user =
User.find_for_vkontakte_oauth(request.env[ 'omniauth.auth' ] ,
current_user)
      after_login(@user)
    end

    def odnoklassniki
      @user =
User.find_for_odnoklassniki_oauth(request.env[ 'omniauth.auth' ] ,
current_user)
      after_login(@user)
    end

    def after_login(user)
      if user.persisted?
        sign_in_and_redirect user, event: :authentication
        set_flash_message(:notice, :success, kind:
"#{ request.env[ 'omniauth.auth' ].provider }.capitalize) if
is_navigational_format?
      else

```

```

        session[ "devise.#{ provider} _data" ] =
env[ 'omniauth.auth' ]
        redirect_to new_user_registration_path
    end
end

```

```

def passthru

```

```

end
end

```

Модель Relationship имеет только два метода: create и destroy

```

class RelationshipsController < ApplicationController

```

```

    def create
        @book =
Relationship.find( params[ :relationship ][ :book_id ] )
        @release_date = params[ :relationship ][ :release_date ]
        @relationship = Relationship.new( book: @book, user:
@current_user
        if @relationship.save
            respond_to do |format|
                format.html { redirect_to @task }
                format.js
            end
        else
            redirect_to root_path
        end
    end
end

```

```

    def destroy
        rel = Relationship.find( params[ :id ] )
        if rel.delete
            respond_to do |format|
                format.html { redirect_to @task }
                format.js
            end
        else
            redirect_to root_path
        end
    end
end
end

```

Проверка доступности даты для записи происходит в методе валидации, описанном в предыдущей главе. Также, уведомление человеку о том, что книга освободилась досрочно произойдет в обратном вызове при удалении записи

3.4.4. Файлы представления

После обрабатывания каждого метода контроллера происходит генерация файлов отображения. Контроллеры инициализируют глобальные переменные, которые затем использует представления для генерации представления [8].

В приложении был создан главный файл «application.html.slim», в котором хранится информация, общая для каждой страницы и отведено место для прорисовки содержимого страниц. На месте ключевого слова «yield» происходит инстанцирование информации, уникальной для каждой страницы. См Приложение. А. Листинг «Файлы представления».

Для отображения коллекций необходимо вызвать функцию «render» с параметром коллекции элементов. Она автоматически сгенерирует каждый элемент. Необходимо только создать файл с правилами отображения элемента. Если не указано имя файла, файл должен начинаться с подчеркивания и иметь именем название модели, которой принадлежит файл. Строка «render @books», приведет к генерации для каждого элемента массива разметки из файла «_book.html.slim»(представлен ниже), в который будет передан параметр с именем в единственном числе. В нашем случае «book».

```
= link_to book, class: 'collection-item avatar'
  .vacancy
    p
      b Название:
      = ' ' + book.name
    p
      b Автор:
      = ' ' + book.author
```

Поиск

Выберите категорию книги

Должна быть свободна

До

НАЙТИ

СБРОС

					ВКР 090301.03 17 81	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Понедельник

АПР
23
2017

◀

Апрель ▼

2017 ▼

▶

S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

СЕГОДНЯ ОЧИСТИТЬ x

```

$slider = $("#slider-range");
min = parseInt($start_cost.val());
max = parseInt($last_cost.val());
$slider.slider({
    range: true,
    min: min,
    max: max,
    values: [ min, max] ,
    slide: function (event, ui) {
        $start_cost.val(ui.values[ 0] );
        $last_cost.val(ui.values[ 1] );
    }
});

$start_cost.on('input keyUp', function(){
    if(+$start_cost.val() > +$last_cost.val()){
        $last_cost.val(+$start_cost.val());
        $slider.slider("values", 1, [ +$start_cost.val()]);
    }
    $slider.slider("values", 0, [ +$start_cost.val()]);
});

$last_cost.on('input keyUp', function () {
    if (+$last_cost.val() < +$start_cost.val()) {
        $start_cost.val(+$last_cost.val());
        $slider.slider("values", 0, [ +$last_cost.val()]);
    }
    $slider.slider("values", 1, [ +$last_cost.val()]);
});
});

```

Для внутренней индексации вакансий было решено использовать поисковой движок «ElasticSearch», написанный на языке Java. Для интеграции движка с Rails приложением существует библиотека «searchkick». Библиотека имеет удобный интерфейс и легко интегрируется с приложением. Рассмотрим алгоритм интеграции подробнее.

Для указания того, что модель должна индексироваться поисковиком, необходимо добавить в исходный код класса строку:

```
searchkick language: 'Russian'
```

Данный метод принимает параметром хеш, одним из ключей которого является язык, на котором написан текст для индексации. Для поиска

					ВКР 090301.03 17 81	Лист
Изм.	Лист	№ докум.	Подпись	Дата		36

необходимо вызывать статичный метод «search», в котором необходимо передать параметром хеш «where» с параметрами поиска.

Заполнив форму, пользователь отправляет ее на сервер. При получении запроса подготавливаются параметры для организации поиска.

```
query = params[:index]
query[:query] = '*' if query[:query].blank?
@vacancies = Book.joins(:relationships).search
query[:query], where: {
  active: true,
  category_id: query[:category],
  'release_date > ?' => query[:date]
}, page: params[:page], per_page: 20
```

Далее полученные данные передаются клиенту асинхронно, а в браузере данные на странице обновляются.

```
respond_to do |format|
  format.js
  format.html
end
```

Асинхронное обновление страниц в Rails приложениях реализуется просто. Для этих целей необходимо написать код, представленный выше, у формы указать ключ «remote: true». При получении запроса, приложение будет отрисовывать файл с именем, совпадающим с именем метода контроллера, если не указано другого, и с расширением «*.js.erb».

Содержимое файла для отображения результатов поиска:

```
$('#books').html("<%= escape_javascript(render @books) %>");
$('#paginator').html("<%= escape_javascript(paginate(@books,
remote: true)) %>");
```

Данный код будет транслирован в код на языке JavaScript, который браузер воспроизведет при получении, что в итоге приведет к выводу списка номеров, найденных по запросу пользователя.

3.4.6. Реализация почтовой рассылки

Было решено после отклика на вакансию, отправлять библиотекаря уведомление на почту. Почтовая рассылка в Rails приложениях реализуется просто. В папке с приложением по умолчанию создается папка «mailers», в которую будут размещаться классы, отвечающие за почтовую рассылку. Базовым классом для почтовых рассылок является класс «ActionMailer::Base». Данный класс реализует протокол SMTP. Программисту необходимо только пронаследоваться от данного класса и реализовать необходимые методы. В разрабатываемом приложении было реализовано два класса: один для рассылки уведомлений, другой для работы с аккаунтами, в частности для восстановления пароля.

```
class ApplicationMailer < ActionMailer::Base
  default from: 'diplom.library@yandex.ru'

  def send_relationship_notification(relationship)
    @book = relationship.book
    @user = relationship.user
    mail(to: 'diplom.library@yandex.ru', subject: 'Книгу
зарезервировали для прочтения')
  end

end
```

В методе «send_relationship_notification» параметры инициализируют глобальные переменные, которые впоследствии будут использоваться в представлении для создания текста письма на языке гипертекстовой разметки HTML.

Второй класс почтовой рассылки пронаследован не напрямую от «ActionMailer::Base», а от «DeviceMailer», который, в свою очередь, имеет в родителях необходимый класс. Данный почтовый класс используется библиотекой «Device» для рассылки писем для работы с аккаунтом пользователя. Для регистрации данного класса в качестве целевого для библиотеки, необходимо в конфигураторе указать имя класса.

```
Devise.setup do |config|
  config.mailer = ApplicationMailer
  ...
end
```

В указанном классе необходимо переопределить при необходимости методы, для которых необходимо применить собственную реализацию. В данном дипломной проекте были перегружено 4 метода:

```
class ApplicationMailer < MandrillMailer::TemplateMailer
  include Devise::Controllers::UrlHelpers
  include Rails.application.routes.url_helpers
  default from: 'diplom.libriary@yandex.ru'

  def main_app
    Rails.application.class.routes.url_helpers
  end

  def confirmation_instructions(record, token, opts = {})
    vars = {
      USER_USERNAME: record.username,
      UNLOCK_URL: confirmation_url(record,
confirmation_token: token)
    }
    invite(record.email, 'Подверите Ваш электронный адрес',
'confirmation_instructions', vars)
  end

  def reset_password_instructions(record, token, opts = {})
    vars = {
      USER_USERNAME: record.username,
      UNLOCK_URL: edit_password_url(record,
reset_password_token: token)
    }
    invite(record.email, 'Инструкции по восстановлению
пароля', 'reset_password_instructions', vars)
  end

  def unlock_instructions(record, token, opts = {})
    vars = {
      USER_USERNAME: record.username,
      UNLOCK_URL: unlock_url(record, unlock_token: token)
    }
    invite(record.email, 'Инструкции по разблокировке
аккаунта', 'unlock_instructions', vars)
  end
end
```

```

protected
def invite(email, subject, template, vars)
  mandrill_mail(
    template: template,
    subject: subject,
    to: email,
    vars: vars,
    important: true,
    inline_css: true)
end
end

```

Для каждого метода необходимо создавать файл разметки, аналогично любому другому файлу. Данное решение позволяет создавать красивые и информативные письма для почтовой рассылки, ведь программист в праве использовать все стили, созданные в приложении, любые переменные и объекты, а также средства фреймворка Ruby on Rails.

Но без указания SMTP сервера рассылка не будет работать. Для этого необходимо зарегистрировать почту на любом почтовом сервере, в данном случае почта была зарегистрирована на сервере Яндекса. Далее необходимо узнать адрес, порт и другие данные сервера и на основе их создать конфигурационный файл.

```

ActionMailer::Base.smtp_settings = {
  address:          'smtp.yandex.ru',
  port:             587,
  user_name:        'diplom.library@yandex.ru',
  password:         12345678,
  domain:           'library.findprofy.ru',
  authentication:   'plain',
  enable_starttls_auto: true
}
ActionMailer::Base.delivery_method = :smtp

```

Для отправки письма необходимо в любом месте программы написать код, указанный ниже:

```

ApplicationMailer.send_relationship_notification(@r).deliver_now

```

Указанный код отправляет письмо синхронно, что может повлиять на скорость отклика сервера. Для асинхронной отправки письма необходимо вызвать метод «deliver_later», который сразу же вернет управление, но в другой нити начнет асинхронную отставку письма. В протоколе сервера можно увидеть содержимое письма, отправленного по протоколу SMTP. Пройдя через сервер Яндекса, оно будет отправлено пользователю.

3.4.7. Подключение формы обратный связи

Для связи с библиотекарем или для ответа на вопрос, возникший у пользователя, в приложение была подключена форма обратной связи. Используемая в приложении форма создана компанией «Jivosite» и имеет бесплатную версию. Для создания собственной формы необходимо зарегистрироваться на сайте компании и создать в конструкторе собственную форму. В результате создается JavaScript файл, который необходимо подключить к сайту, установить мобильное приложение и привязать к нему данную форму.

```
<!-- BEGIN JIVOSITE CODE {literal} -->
<script type='text/javascript'>
    (function(){ var widget_id = 'LK1MumUCRg';
        var s = document.createElement('script'); s.type =
'text/javascript'; s.async = true; s.src =
'//code.jivosite.com/script/widget/'+widget_id; var ss =
document.getElementsByTagName('script')[ 0] ;
ss.parentNode.insertBefore(s, ss);} ) ();</script>
<!-- {/literal} END JIVOSITE CODE -->
```

3.5. Разработка дизайна приложения

3.5.1. Общий стиль

Для создания современного дизайна приложения использовалась библиотека «materialize». Данная библиотека содержит в себе набор классов

					ВКР 090301.03 17 81	Лист
						41
Изм.	Лист	№ докум.	Подпись	Дата		

«Ромео и Джульетта» — Уильям Шекспир

Название: Ромео и Джульетта
Автор: Уильяма Шекспира
Год: 1594-1595

Описание

Непосредственной сюжетной основой трагедии Шекспира послужила, по всей видимости, поэма Артура Брука «Трагическая история Ромеуса и Джульетты» (Arthur Brooke. The tragicall Historie of Romeus and Juliet, 1562). Упомянув о некой предшествующей пьесе, Брук, тем не менее, разработал сюжет, воспринятый им из переведённой на французский язык новеллы итальянского литератора Маттео Банделло, в оригинале вышедшей в 1554 году в сборнике «Четыре книги новелл» (Часть вторая, новелла IX, изд. в Лукке). У Брука также присутствуют отголоски поэзии Джефри Чосера («Troilus and Criseyde», 1581)[1]. Повествование Банделло представляло собой расширенный, детализированный пересказ более компактного произведения Луиджи Да Порто (Luigi Da Porto, 1485—1529) «Новонайденная история двух благородных влюблённых и их печальной смерти, произошедшей в Вероне во времена синьора Бартоломео дела Скала» (Historia novellamente ritrovata di due nobili amanti, 1524), в котором впервые в литературе появились образы Ромео и Джульетты (Romeo Montecchi e Giulietta Cappelletti) и некоторые другие персонажи (монах Лоренцо, Маркуччо, Тебальдо, граф ди Лодроне — жених Джульетты), получившие развитие в пьесе Шекспира. Новелла Да Порто неоднократно (в 1531 и 1535 годах) печаталась в Венеции (в 1539 году была издана под названием «Джульетта»/Giulietta) и пользовалась большим успехом[1]. Произведение Да

Статистика

Статус: свободна

Количество прочтений: 1

[ЗАБРОНИРОВАТЬ КНИГУ](#)

КНИГИ

Программирование на С или C++

Автор: Линдемманн

Предмет: Программирование

База данных SQL

Автор: Максвелл

Предмет: Программирование

Читает в данный момент



Наталья

До: 10.05.17

 ВСТАТЬ В ОЧЕРЕДЬ

Очередь



Виктор

До: 17.05.17



Ирина

До: 22.05.17



Боромир

До: 01.06.17

					ВКР 090301.03 17 81	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Регистрация



Email



Имя



Фамилия



Birthday



Мобильный телефон



Пароль



Подтверждение пароля

ЗАРЕГИСТРИРОВАТЬСЯ

					ВКР 090301.03 17 81	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Leaderator

Панель управления

Users

Книги

Очередь

Категории

Панель управления

Статистика

Всего пользователей: 1854

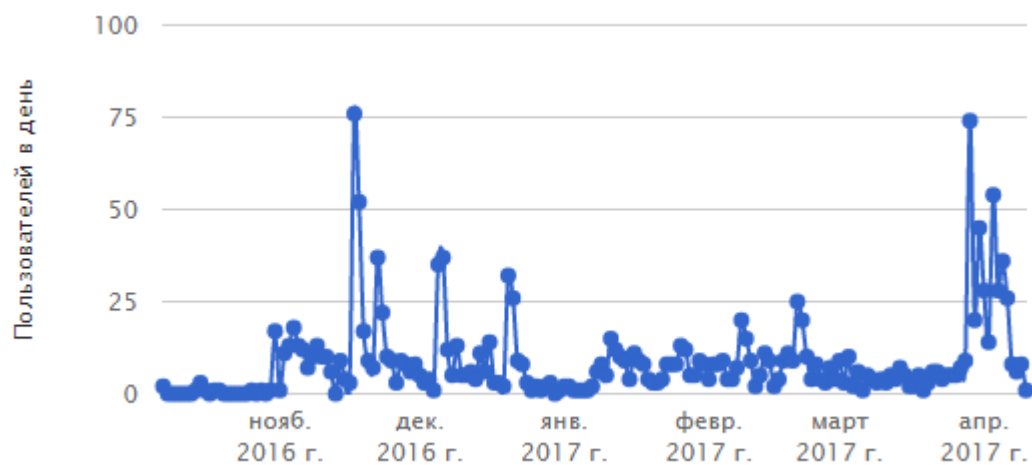
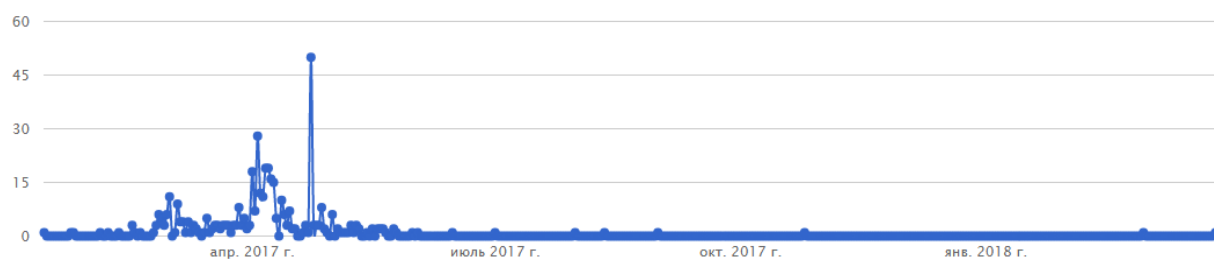
Зарегистрированных пользователей за 24 часа: 141

Книг: 14

Книг на руках: 10

Людей ожидает: 2

					ВКР 090301.03 17 81	Лист
Изм.	Лист	№ докум.	Подпись	Дата		



средними экранами в промежутке между 600 и 992 пикселями, маленькими с разрешением менее 600 пикселей. [6].

Для целей данного дипломного проекта было недостаточно использования данных классов и были разработаны собственные с использованием медиазапросов.

Представленный ниже исходный код демонстрирует использование медиазапросов. В нем, в зависимости от ширины экрана, устанавливаются различные отступы от краев экрана.

```
@media (max-width: 1992px) {  
  .container {  
    width: 80% !important;  
  }  
}  
  
@media (max-width: 992px) {  
  .container {  
    width: 85% !important;  
  }  
}  
  
@media (max-width: 600px) {  
  .container {  
    width: 100% !important;  
  }  
}
```

Библиотеке «Materialize» позволяет использовать различные типы меню для различных размеров экрана. На экране настольного компьютера меню отображается так же, как на рисунке 4. В мобильной же версии, меню преобразуется к боковому меню, что изображен на рисунках 9 и 11 (на рисунке 9 меню в скрытом состоянии).

➔ Войти	ханике
Книги	ке
Бронирование	ал: Скачать файл
Архив	

3.7. Размещение на сервере

Для размещения на сервере необходимо установить Nginx, а также Ruby on Rails, аналогично пунктам 3.2.2 и 3.2.3.

Nginx устанавливается на сервер при помощи двух команд:

```
sudo apt-get update  
sudo apt-get install nginx
```

Затем его необходимо настроить. Настройка прокси-сервера происходит путем соотнесения DNS-адреса, на запросы к которому необходимо отвечать сайту, а также указанию пути да файлов с номером процесса приложения и номером сокета, через который общаются Backend и Frontend сервера.

Для размещения самого приложения воспользовались библиотекой «Capistrano». Для корректной работы библиотеки необходимо подключить все необходимые модули, а также написать скрипт «деплой».

Дополнительно необходимо указать в приложении IP-адреса, по которым будет происходить размещение, а также конфигурацию приложения. Поддерживается указание различных серверов для различных конфигураций, для того чтобы исключить возможность повреждения данных при тестировании приложения. См. Приложение А. Листинг. «Размещение на сервере».

4. ФУНКЦИОНИРОВАНИЕ ПРОГРАММЫ

4.1. Диаграмма использования

На рисунке 12 представлена диаграмма использования. На диаграмме представлено два субъекта: библиотекарь и читатель. Приложение решает проблемы из обоих. Соискатель может найти нужную библиотеку из библиотечного фонда и забронировать на прочтение, если книга занята, то читатель может встать в очередь. Сервис упрощает работу библиотекаря, систематизирует ее, а также позволяет ему избавиться от бюрократии и работы с карточками. Оба субъекта использования системы имеют возможность просмотра статистики.

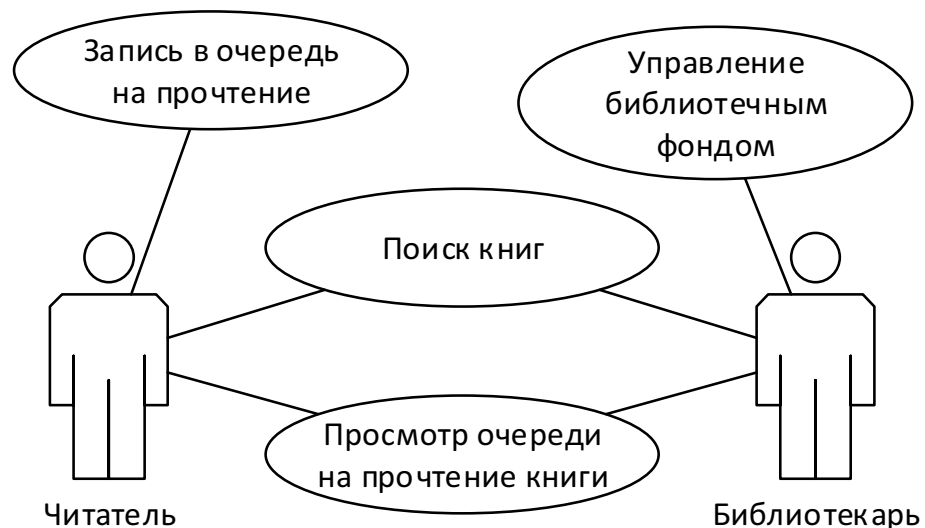
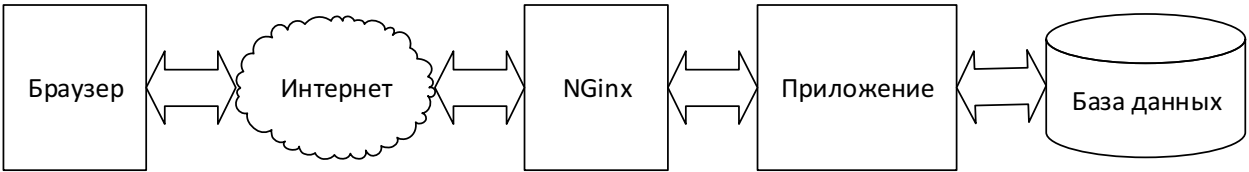
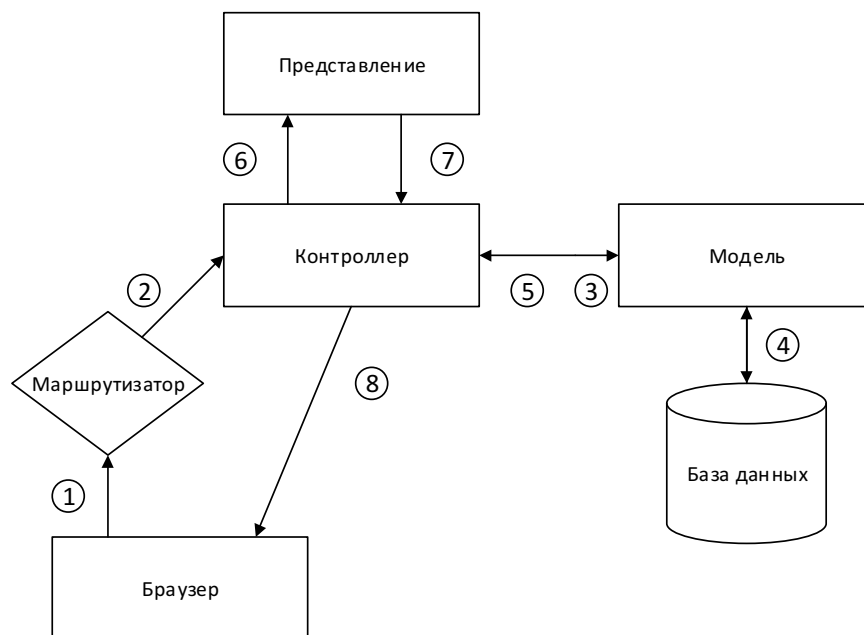


Рисунок 12 – Диаграмма использования

4.2. Описание схемы работы системы

На рисунке 13 представлена схема работы системы. Пользователь отправляет запросы к приложению через сеть интернет, прокси-сервер Nginx получает их и перенаправляет приложению. Приложение в свою очередь





5. Получив необходимые данные, контроллер начинает подготовку HTML-страницы для отображения, для этого он передает управление представлению;
6. Представление, получив управление имеет в распоряжении глобальные переменные, которые создал контроллер. С их использованием происходит создание HTML-страницы. Применительно для рассматриваемого случая, представление с использованием метода «render» отрисовывает для каждого объекта массива свою шаблонную часть страницы;
7. Подготовив все необходимые данные для отображения, представление возвращает управление контроллеру;
8. Контроллер возвращает браузеру HTML-страницу. Браузер при необходимости напрямую получает файлы стилей и изображения, в обход данной схемы. В итоге пользователю отображен список отелей.

Данная схема является стандартной не только для любого Ruby on Rails приложения, но и с небольшими корректировками может подойти для любого web-приложения, созданного с использованием паттерна проектирования MVC.

4.4. Описание схемы программы

На рисунке 15 представлена схема работы программы при записи на прочтение книги. Представленная схема укрупнена и абстрагирована от деталей реализации.

					ВКР 090301.03 17 81	Лист
						53
Изм.	Лист	№ докум.	Подпись	Дата		



Рисунок 15 – Схема резервирования книги

Резервирование книги состоит из нескольких последовательных этапов. Предварительно читателю необходимо зарегистрироваться на сервисе. Далее пользователь уже может выбрать понравившуюся книгу. Далее пользователь может узнать, свободна книга или нет. Если книга свободна, то он сразу может ее зарезервировать для прочтения. Если книга занята, пользователь может увидеть очередь и, если время ожидания его устраивает, встать в очередь и дожидаться возврата книги. После прочтения книги в установленный срок пользователь обязан вернуть книгу в библиотеку.

5. ТЕХНОЛОГИЧЕСКАЯ ИНСТРУКЦИЯ ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ

5.1. Описание программы

Разработанная система представлена WEB-сервисом, в котором читатель может найти книгу и забронировать ее для прочтения. Если книга занята, пользователь может встать в очередь. Когда книга освободится, пользователь получит уведомление на электронную почту.

Система имеет административную панель, в которой библиотекарь может работать с книгами и очередью. Вся информация о заявках на прочтение книг дублируется письмом на электронный адрес.

При входе на сайт, посетитель видит список всех книг с возможностью фильтрации, а также имеет возможность просмотреть подробную информацию о них.

Разработанная программа написана на языке Ruby с использованием фреймворка Ruby on Rails. Разработанное приложение реализует паттерн MVC, а также модель CRUD. Архитектурный стиль взаимодействия компонентов – REST.

					ВКР 090301.03 17 81	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

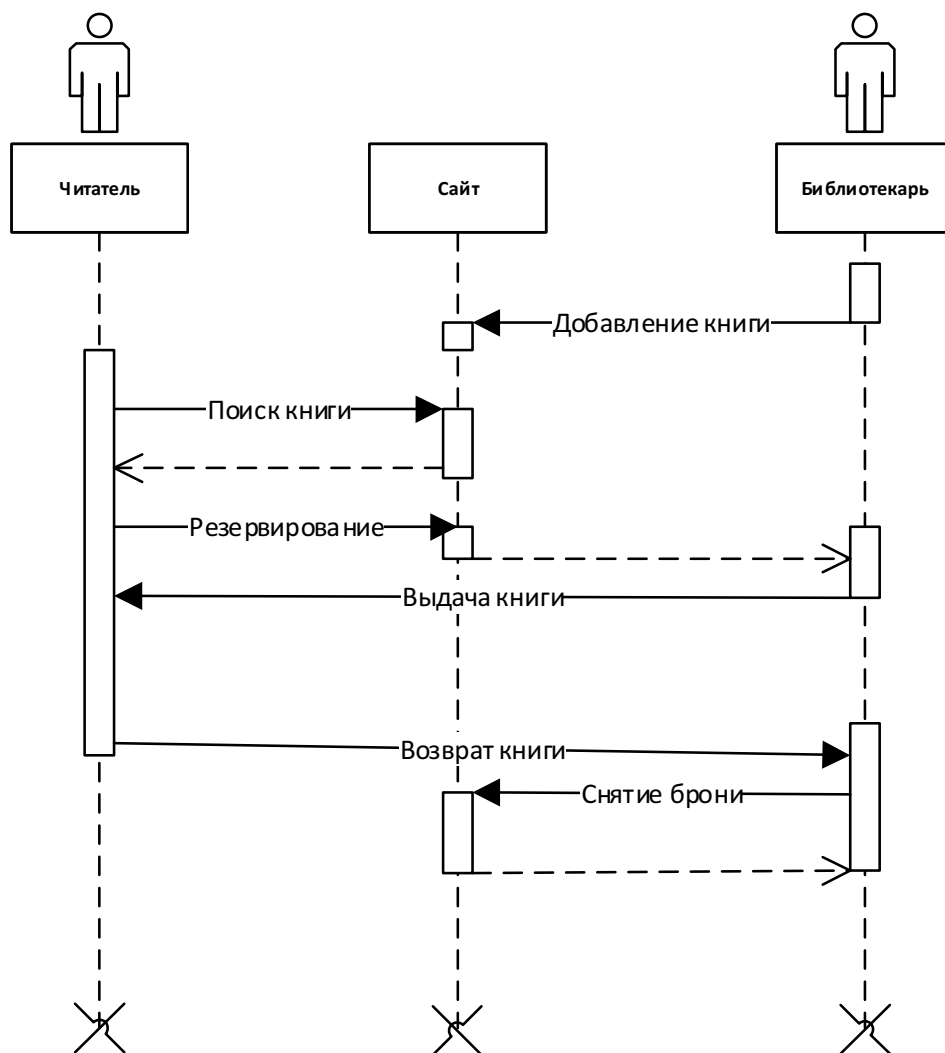


Рисунок 16 – Диаграмма последовательности

Как видно по диаграмме использования, система автоматизирует работу библиотекаря, но не заменяет его. Библиотекарь выдает книги пользователям, зарезервировавшим книгу, а также подтверждает возврат книги на сайте и выставляет фактическую дату возврата книги. Оцифровка книг позволит исключить из данной схемы библиотекаря и очередь, так как книги станут электронными и появится возможность неограниченно увеличивать количество экземпляров любой книги.

5.2. Описание возможных ошибок и способы их устранения

В ходе работы программы возможно возникновение ошибок. С наибольшей вероятностью возможно появление ошибок с кодами: 400, 500,

Страница, которую вы искали не существует.

Вы, возможно, опечатались при вводе адреса или страница была перемещена.

502 Bad Gateway

nginx/1.2.6

Нам очень жаль, но что-то пошло не так.

Вы можете обратиться к разработчикам для скорейшего
исправления ошибки

					BKP 090301.03 17 81	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

можно найти все происходящие ошибки, все запросы, приходящие к приложению.

Данная ошибка требует вмешательства разработчика для устранения. Если проводить параллели с языками программирования, данная ошибка означает необработанное исключение времени исполнения. При отправлении запросов к серверу асинхронно с использованием JavaScript, пользователь не увидит возникновения ошибки, что делает ошибку в данной ситуации сложноопределяемой.

					<i>ВКР 090301.03 17 81</i>	<i>Лист</i>
						59
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		

6. ЗАКЛЮЧЕНИЕ

Задачей дипломного проектирования являлось написание и отладка электронной библиотеки, в которой пользователь должен иметь возможность поиска и бронирования книг. Весь функционал выполнен в полной мере и в качестве, соответствующем требованиям, предъявляемыми к современному программному обеспечению. Разработанная система представлена WEB-сервисом, в котором сотрудники библиотеки имеют возможность добавления, изменения и удаления информации о книгах, управлять очередью бронирования книг, и выдавать книги согласно ей. Пользователь имеет возможность просмотра всех имеющихся книг в библиотеке, их поиска и бронирования. Также пользователь имеет возможность узнать свободна ли книга, в противном случае он может узнать дату, когда книга освободится.

В программе реализована панель администратора, в которой реализован функционал, доступный только для сотрудников библиотеки. К просмотру книг библиотеки допускаются все пользователи, а узнать свободна ли книга или зарезервировать ее могут только зарегистрированные пользователи. Также немаловажной функцией электронного каталога является учет библиотечных фондов.

При входе на сайт, посетитель видит список всех книг с возможностью фильтрации, а также имеет возможность просмотреть подробную информацию о них.

В ходе выполнения дипломного проектирования были продемонстрированы навыки, полученные в ходе обучения в ПГУ. Такие как: умение программировать на объектно-ориентированном языке программирования, умение создавать web-приложения, умение проектировать базы-данных, умение тестировать программные продукты, умение писать документацию, соответствующую требованиям ЕСПД.

					<i>ВКР 090301.03 17 81</i>	<i>Лист</i>
						60
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		

7. СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Д. Румянцев. Сам себе Web-программист. Практикум создания качественного Web-сайта. – М.: ИНФРА-М, 2001.
2. Фримен Э., Робсон Э. Изучаем программирование на JavaScript. – СПб.: Питер, 2015. – 640 с.
3. Закас Н. JavaScript для профессиональных веб-разработчиков / [Пер. с англ. А. Лютича]. – СПб.: Питер, 2015. – 960 с.
4. Бенедетти Р., Крэнли Р. Изучаем работу с jQuery. – СПб.: Питер, 2012. – 528 с.
5. Коннолли Т., Бегг К., Страчан А. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание.: Пер. с англ. – М.: Издательский дом "Вильямс", 2003. – 1440 с.
6. Фултон Х., The Ruby Way. – М.: Питер, 2015 – 348 с.
7. HTML5, CSS3 и JavaScript. Исчерпывающее руководство — Дженнифер Нидерст Роббинс. [Пер. с англ. А. Лютича]. – СПб.: Питер, 2015. – 160 с.
8. Хартл М, Ruby on Rails Tutorial. – М.: Питер, 2011 – 514 с.
9. Бахтизин, В.В. Технология разработки программного обеспечения / В.В Бахтизин, Л. А. Глухова, – Минск БГУИР, 2010. – 408с.
10. Ruby on Rails по-русски [Электронный ресурс]. – Электрон.дан. – URL: <http://rusrails.ru/> (дата обращения 16.04.2017).
11. Ruby on Rails [Электронный ресурс]. – Электрон.дан. – URL: https://ru.wikipedia.org/wiki/Ruby_on_Rails (дата обращения 20.04.2017);
12. PostgreSQL [Электронный ресурс]. – Электрон.дан. – URL: <https://ru.wikipedia.org/wiki/PostgreSQL> (дата обращения 20.04.2017);
13. Ruby [Электронный ресурс]. – Электрон.дан. – URL: <https://ru.wikipedia.org/wiki/Ruby> (дата обращения 20.04.2017)

14. Орлов, С. Технологии разработки программного обеспечения / С. Орлов, Б. Цилькер. – СПб.: Питер, 2012. - 464 с.

ПРИЛОЖЕНИЯ

					<i>ВКР 090301.03 17 81</i>	<i>Лист</i>
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		62

ПРИЛОЖЕНИЕ А. ЛИСТИНГ

Регистрация через соц. сети

```
def self.find_for_facebook_oauth(auth, signed_in_resource)
  identity = Identity.find_for_oauth(auth)
  user = signed_in_resource ? signed_in_resource :
identity.user
  if user.nil?
    email = auth.extra.raw_info.email == nil ?
'blanck@blanck.ru' : auth.extra.raw_info.email
    user = User.with_deleted.where(email: email).first if
email

    if user.nil?
      user = User.new(name:
auth.extra.raw_info.first_name,
                      surname: auth.extra.raw_info.last_name,
                      sex: auth.extra.raw_info.gender == 'male' ?
2 : 1,
                      email: email,
                      remote_avatar_url: auth.info.image,
                      password: Devise.friendly_token[0,20])
      user.save!
    else
      user.restore
    end
  end
  if identity.user != user
    identity.user = user
    identity.save!
  end
  user
end

def self.find_for_vkontakte_oauth(auth, signed_in_resource
= nil)
  identity = Identity.find_for_oauth(auth)
  user = signed_in_resource ? signed_in_resource :
identity.user
  if user.nil?
```

					ВКР 090301.03 17 81	Лист
Изм.	Лист	№ докум.	Подпись	Дата		63

```

        email = auth.info.email == nil ? 'blanck@blanck.ru' :
auth.info.email
        user = User.with_deleted.where(email: email).first
if email

        # Если пользователь удалил аккаунт, но потом решил
зайти еще раз
        if user.nil?
            user = User.new(name:
auth.extra.raw_info.first_name,
            surname: auth.extra.raw_info.last_name,
            sex: auth.extra.raw_info.sex,
            email: email,
            vkontakte: auth.info.urls.Vkontakte,
            birthday: auth.extra.raw_info.bdate,
            remote_avatar_url: auth.extra.raw_info.photo_200_orig,
            password: Devise.friendly_token[0,20])
            user.save!
        else
            user.restore
        end
    end
    if identity.user != user
        identity.user = user
        identity.save!
    end
    user
end

    def self.find_for_odnoklassniki_oauth(auth,
signed_in_resource = nil)
        identity = Identity.find_for_oauth(auth)
        user = signed_in_resource ? signed_in_resource :
identity.user
        if user.nil?
            email = auth.extra.raw_info.email == nil ?
'blanck@blanck.ru' : auth.extra.raw_info.email
            user = User.with_deleted.where(email: email).first if
email

            if user.nil?
                user = User.new(name:
auth.extra.raw_info.first_name,
                    surname: auth.extra.raw_info.last_name,
                    sex: auth.extra.raw_info.gender == 'male' ?
2 : 1,

```

```

        email: email,
        remote_avatar_url: auth.info.image,
        password: Devise.friendly_token[0,20])
      user.save!
    else
      user.restore
    end
  end
  if identity.user != user
    identity.user = user
    identity.save!
  end
  user
end

def update_with_password(params={})
  if params[:password].blank?
    params.delete(:password)
    params.delete(:password_confirmation) if
params[:password_confirmation].blank?
  elsif params[:password] !=
params[:password_confirmation]
    errors.add(:password_confirmation, 'Пароли не
совпадают')
    errors.add(:password)
    return false
  end
  unless params[:avatar].blank?
    self.avatar = params[:avatar]
    params.delete(:avatar)
    save
  end
  update_attributes(params)
end
end

```

Контроллеры

```
class ApplicationController < ActionController::Base
  protect_from_forgery with: :exception

  before_filter :configure_permitted_parameters, if:
:devise_controller?
  before_filter :miniprofiler
  before_action :start_event_machine

  USER_CHANNEL_KEY = 'user_channel_key'

  def authenticate_admin_user!
    authenticate_user!
    unless current_user.has_role? :admin
      flash[:alert] = 'У вас недостаточно прав для доступа к
данному ресурсу!'
      redirect_to root_path
    end
  end

  def after_sign_in_path_for(resource)
    if resource.is_a? User
      set_user_channel_cookie
    end
    super
  end

  def after_sign_out_path_for(resource)
    if resource.is_a? User
      clear_user_channel_cookie
    end
    super
  end

  protected

  def miniprofiler
    if signed_in? && current_user.admin? ||
!Rails.env.production?
      Rack::MiniProfiler.authorize_request
    end
  end
end
```

					BKP 090301.03 17 81	Лист
Изм.	Лист	№ докум.	Подпись	Дата		66

```

        end
    end

    def configure_permitted_parameters
        devise_parameter_sanitizer.for(:sign_up) do |u|
            u.permit(:name, :email, :password, :sex, :birthday,
                    :name, :surname, :mobile_phone)
        end
        devise_parameter_sanitizer.for(:account_update) do |u|
            u.permit( :name, :email, :password,
:password_confirmation, :sex, :birthday,
                    :name, :surname, :mobile_phone, :avatar,
:mobile_phone)
        end
    end

end
end
end

```

					<i>BKP 090301.03 17 81</i>	Лист
						67
Изм.	Лист	№ докум.	Подпись	Дата		

Регистрация

```
$(document).ready(function () {
    $('#user_birthday').pickadate('picker').set('max', new
Date());

    $("#user_email, #user_name, #user_surname, #user_birthday,
#user_password,
#user_password_confirmation").keypress(function(){
        hideErrorMessage($(this))
    });

    var $birthday = $('#user_birthday');
    $birthday.click(function(){
        hideErrorMessage($(this))
    });

    $("#send").on('click', function() {
        removeErrorMessages();
        var focused = false;
        var success = true;

        var $email = $('#user_email');
        if($email.val().length == 0){
            addErrorMessage($email, 'Введите Email', focused);
            success = false;
            focused = true;
        }
        var $name = $('#user_name');
        if($name.val().length == 0){
            addErrorMessage($name, 'Введите имя', focused);
            success = false;
            focused = true;
        }
        var $surname = $('#user_surname');
        if($surname.val().length == 0){
            addErrorMessage($surname, 'Введите
фамилию', focused);
            success = false;
            focused = true;
        }
        var error = false;
        var $user_password = $('#user_password');
        if($user_password.val().length == 0){
```

```

        addErrorMessage($user_password, 'Введите
пароль',focused);
        error = true;
        success = false;
        focused = true;
    }
    var $user_password_confirmation =
$('#user_password_confirmation');
    if($user_password_confirmation.val().length == 0){
        addErrorMessage($user_password_confirmation,
'Повторите введенный пароль',focused);
        error = true;
        success = false;
        focused = true;
    }

    if (!error && $user_password.val() !=
$user_password_confirmation.val()) {
        addErrorMessage($user_password_confirmation, 'Пароли
не совпадают',focused);
        success = false;
    }
    if(success == true) {
        $('#new_task').submit();
        return true;
    }
    return false;
});
});

```

Файлы представления

```
doctype html
html lang="ru"
  head
    meta charset="utf-8"
    meta http-equiv="X-UA-Compatible" content="IE=edge"
    meta name="viewport" content="width=device-width,
initial-scale=1.0"
    title= content_for?(:title) ? yield(:title) :
'Электронная библиотека'
    meta name="description"
content="#{ content_for?(:description) ? yield(:description) :
'информационно-справочное web-приложение
для онлайн вронирования книг' }"
    = csrf_meta_tags
    = stylesheet_link_tag 'books'
    = javascript_include_tag 'application', 'data-
turbolinks-track' => true
    = stylesheet_link_tag 'application', media: 'all',
'data-turbolinks-track' => true
    = stylesheet_link_tag 'font-awesome'
    = stylesheet_link_tag 'fancybox'
    = stylesheet_link_tag
'https://fonts.googleapis.com/icon?family=Material+Icons'
    = favicon_link_tag 'favicon-144x144.png', :rel =>
'apple-touch-icon-precomposed', :type => 'image/png', :sizes =>
'144x144'
    = favicon_link_tag 'favicon-114x114.png', :rel =>
'apple-touch-icon-precomposed', :type => 'image/png', :sizes =>
'114x114'
    = favicon_link_tag 'favicon-72x72.png', :rel =>
'apple-touch-icon-precomposed', :type => 'image/png', :sizes
=> '72x72'
    = favicon_link_tag 'favicon-144x144.png', :rel =>
'apple-touch-icon-precomposed', :type => 'image/png'
    = favicon_link_tag 'favicon-64x64.png', :rel =>
'shortcut icon'
    = javascript_include_tag '//yastatic.net/share/share.js'
/= javascript_include_tag 'jivosite.js'
/! Le HTML5 shim, for IE6-8 support of HTML elements
/[ if lt IE 9]
```

					ВКР 090301.03 17 81	Лист
Изм.	Лист	№ докум.	Подпись	Дата		70

```
      = javascript_include_tag
'//cdnjs.cloudflare.com/ajax/libs/html5shiv/3.7.2/html5shiv.min.
js'
```

```
body
  #flash
    = bootstrap_flash
  =render 'layouts/header'
main
  .row
    = yield
  =render 'layouts/footer'
  =render 'layouts/jivo'
  - if Rails.env.production?
    =render 'layouts/google_analytics'
    =render 'layouts/yandex_metrika'
  javascript:
    $(document).ready(function () {
      $alert = $('.alert');
      var text = $alert.first().text();
      $alert.remove();
      if (jQuery.trim(text).length > 0) {
        Materialize.toast(text, 5000);
      }
    });
```

					BKP 090301.03 17 81	Лист
Изм.	Лист	№ докум.	Подпись	Дата		71

Регистрация

```
$(document).ready(function () {
    $('#user_birthday').pickadate('picker').set('max', new
Date());

    $("#user_email, #user_name, #user_surname, #user_birthday,
#user_password,
#user_password_confirmation").keypress(function(){
    hideErrorMessage($(this))
});

var $birthday = $('#user_birthday');
$birthday.click(function(){
    hideErrorMessage($(this))
});

$("#send").on('click', function() {
    removeErrorMessages();
    var focused = false;
    var success = true;

    var $email = $('#user_email');
    if($email.val().length == 0){
        addErrorMessage($email, 'Введите Email', focused);
        success = false;
        focused = true;
    }
    var $name = $('#user_name');
    if($name.val().length == 0){
        addErrorMessage($name, 'Введите имя', focused);
        success = false;
        focused = true;
    }
    var $surname = $('#user_surname');
    if($surname.val().length == 0){
        addErrorMessage($surname, 'Введите
фамилию', focused);
        success = false;
        focused = true;
    }
    var error = false;
    var $user_password = $('#user_password');
```

```

        if($user_password.val().length == 0){
            addErrorMessage($user_password, 'Введите
пароль',focused);
            error = true;
            success = false;
            focused = true;
        }
        var $user_password_confirmation =
$('#user_password_confirmation');
        if($user_password_confirmation.val().length == 0){
            addErrorMessage($user_password_confirmation,
'Повторите введенный пароль',focused);
            error = true;
            success = false;
            focused = true;
        }

        if (!error && $user_password.val() !=
$user_password_confirmation.val()) {
            addErrorMessage($user_password_confirmation, 'Пароли
не совпадают',focused);
            success = false;
        }
        if(success == true) {
            $('#new_task').submit();
            return true;
        }
        return false;
    });
});

```

Размещение на сервере

```
lock '3.4.0'
set :application, 'hotels'
set :repo_url, 'git@bitbucket.org:hotels/hotels.git'
set :shared_path, "#{ deploy_to } /shared"
set :linked_files, %w{ config/database.yml config/secrets.yml }
set :linked_dirs, %w{ log tmp/pids tmp/cache tmp/sockets
vendor/bundle public/uploads uploads}
set :deploy_to, '/work/www/vacancies'
set :puma_threads, [ 0, 16]
set :puma_workers, 0
set :use_sudo, false
set :puma_bind, 'unix:/tmp/vacancies.sock'
set :puma_state, "#{ shared_path } /tmp/pids/puma.state"
set :puma_pid, "#{ shared_path } /tmp/pids/puma.pid"
set :puma_access_log, "#{ shared_path } /log/puma.error.log"
set :puma_error_log, "#{ shared_path } /log/puma.access.log"
set :puma_preload_app, true
set :puma_worker_timeout, nil
set :ssh_options, { forward_agent: true, user: fetch(:user),
keys: %w(~/.ssh/id_rsa.pub) }
set :puma_init_active_record, true

namespace :puma do
  desc 'Create Directories for Puma Pids and Socket'
  task :make_dirs do
    on roles(:app) do
      execute "mkdir #{ shared_path } /tmp/sockets -p"
      execute "mkdir #{ shared_path } /tmp/pids -p"
    end
  end

  before :start, :make_dirs
end

namespace :deploy do
  after :publishing, :restart
  desc 'Initial Deploy'
  task :initial do
    on roles(:app) do
      before 'deploy:restart', 'puma:start'
      invoke 'deploy'
    end
  end
end
```

```

end

desc 'Restart application'
task :restart do
  on roles(:app), in: :sequence, wait: 5 do
    invoke 'puma:restart'
  end
end

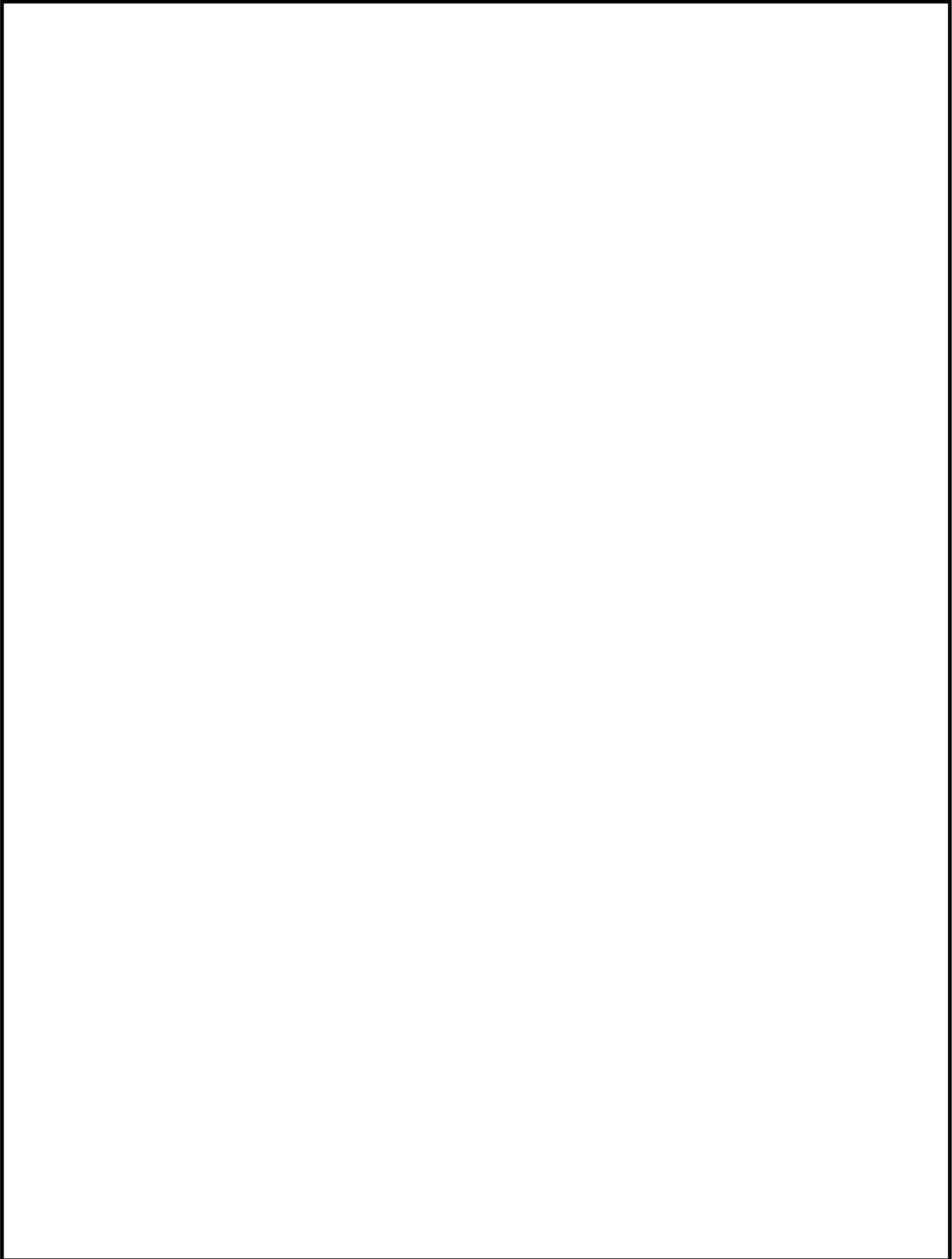
desc 'Stop application'
task :stop do
  on roles(:app), in: :sequence, wait: 5 do
    invoke 'puma:stop'
  end
end

desc 'Start application'
task :start do
  on roles(:app), in: :sequence, wait: 5 do
    invoke 'puma:start'
  end
end

after :finishing, :compile_assets
after :finishing, :cleanup
after :finishing, :restart
end

```


							Лист
					БКР 090301.03 17 81		77
Изм.	Лист	№ докум.	Подпись	Дата			



					<i>ПГУ 1.09.05.01.54.008</i>			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Акинтеев М.А.			Разработка web-прилож...(см. тит. лист) Физическая модель БД.	Лит.	Лист	Листов
Провер.		Бурукина И.П.						
Н. Контр		Бождай А.С.						78
						ФВТ		
Утверд.		Бершадский А.М.				гр.13ВВ2		