АВТОМАТИЗАЦИЯ ПРОЦЕССА УЧЕТА ВЗРЫВНЫХ РАБОТ НА РАЗРЕЗАХ

Загидуллин Д.Р студент гр. ИТб-161, IV курс
Научный руководитель Ванеев О.Н
Кузбасский государственный технический университет
имени Т.Ф. Горбачева.

Проблема

В рамках решения проблемы по теме «Автоматизация процесса учета взрывных работ на разрезах» был выбран язык Ruby на платформе web-приложения с использованием технологии Ruby on Rails (ROR). Данное программное обеспечение удовлетворяет всем требованиям, необходимым для разработки моего дипломного проекта.

Решения

Среди всего множества CMS и программных обеспечений для web разработки можно выделить следующие, конкурирующие с ROR, приложения: Jango WordPress и 1C-bitrix

Преимущества и недостатки

ROR Преимущества:

- Занимает меньше времени на разработку.
- Модульная конструкция Ruby On Rails и приложения приводит не только к более быстрому развитию, но и в гибкости этих решений.
- Ruby является понятием последовательности в структуре и методологии при написании кода с последующим разработчиками RoR.
- На самом деле, все предыдущие 3 пункта приводят к окончательному снижению затрат и повышение рентабельности инвестиций (ROI).

Недостатки:

- RoR приложения не так быстры, как приложения , написанные на языках Java или C.
- Как правило цена на серверы для начинающих, выше. Сложно запустить RoR сайт на хостинге за 70 рублей.

Jango Преимущества:

- Быстрота разработки
- Фреймворк Django наилучшим образом подходит для работы с самыми высокими трафиками.
- Менеджмент контента, научные вычислительные платформы, даже крупные организации – со всем этим можно эффективно справляться при помощи Django.

Недостатки:

- Использование шаблона маршрутизации с указанием URL
- Django слишком монолитный
- Все базируется на ORM Django
- Компоненты развертываются совместно
- Необходимо умение владеть всей системой для работы

WordPress Преимущества:

- Главное преимущество – система бесплатная. Например, минимальное решение на 1С-Битрикс стоит 15 900 рублей.
- Открытый код, что позволяет дорабатывать веб-ресурс.
- Большой набор дополнений, расширений, шаблонов, причем как коммерческих, так и бесплатных.
- Понятный интерфейс админки и работать в ней легко.

Недостатки:

- Нет официальной техподдержки. Приходится гуглить или придумывать решение самостоятельно.
- Без подходящего плагина нельзя создать даже банальный sitemap или править robots.txt.
- Система изначально не предназначена для создания интернет-магазина и функционал уступает тому же Битриксу.
- Нельзя делать резервные копии из стандартного решения.
- Из-за открытости кода система уязвима – сайт на WordPress легче взломать, чем сайт на CMS с закрытым кодом.

1C-bitrix Преимущества:

- Поддержку визуального редактора для управления основными модулями сайта.
- Разграничение прав пользователей.
- Постоянное обновление и улучшение системы.
- Высокий уровень безопасности.
- Возможность создания модулей на PHP и ASP.NET.
- Интеграцию с продуктами 1С.

Недостатки:

- Громоздкий и неудобный код, недружественный к оптимизации
- Отсутствие подробной документации.
- 1С-Битрикс — высокая стоимость лицензии.

Проектирование

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.

Рассмотрим реализацию проекта на примере части схемы БД о «Взрывных работах», отвечающая за хранение данных о персонале.

Таблица people с параметрами (Фамилия Имя Отчество ID-профессии ID-сертификата), где ID-профессии и ID-сертификата играют роль внешнего ключа.

Таблица Professiya с параметрами (Наиманование профессии)

Таблица Sertifikat с параметрами (Наиманование сертификата)

Первичные ключи во всех таблицах создаются автоматически.

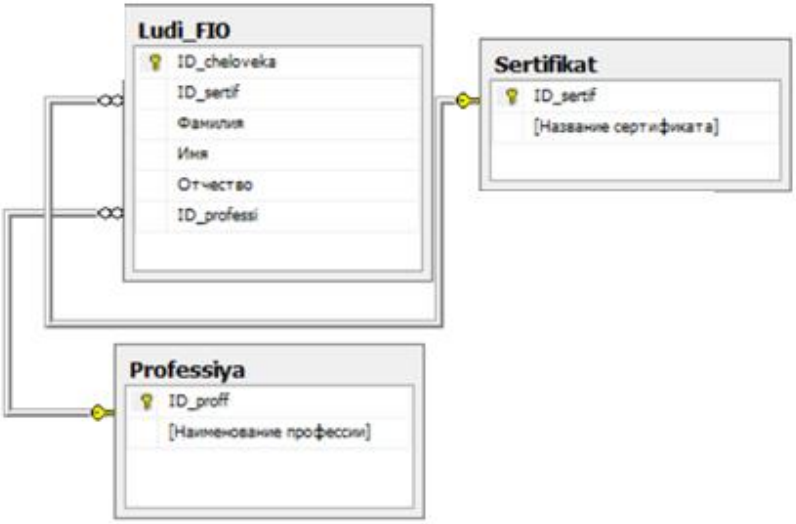


Рис.1 Компонент схемы, отвечающий за персонал

Основные элементы проекта:

- **C**ontroller получает данные от пользователя и передаёт их в Модель. Кроме того, он получает сообщения от Модели и передаёт их в Вид. (Рис.2)

- **M**odel является «сутью» приложения и отвечает за непосредственные алгоритмы, расчёты и тому подобное внутреннее устройство приложения. Также предоставляет линк к хранилищу данных. (Рис.3)
- **V**iew предназначен для вывода данных, предоставленных Моделью. Это единственная часть MVC, которая непосредственно контактирует с пользователем. (Рис.4)

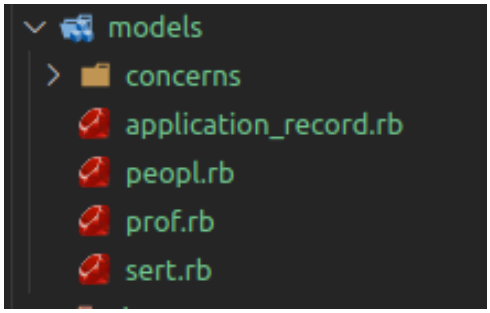


Рис.2 представление модели проекта

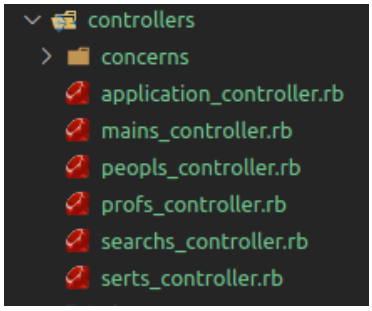


Рис.3 представление контроллеров проекта

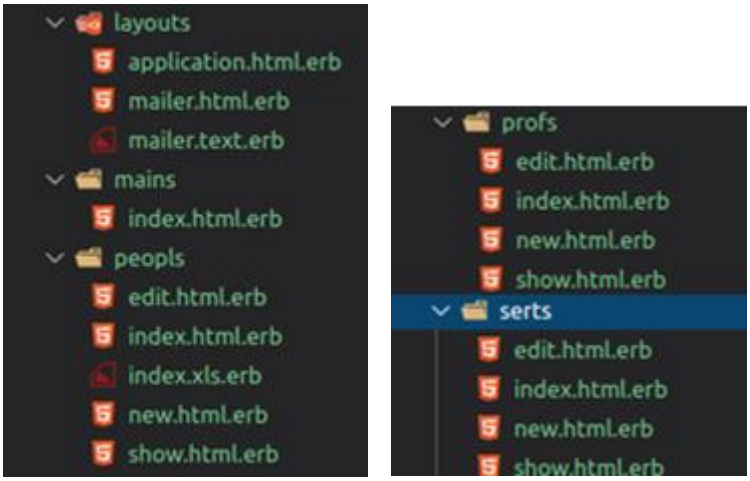


Рис.4 представление отображений проекта

Проект реализует MVC

MVC – это паттерн архитектуры приложения, четко разделяющий три его компонента:

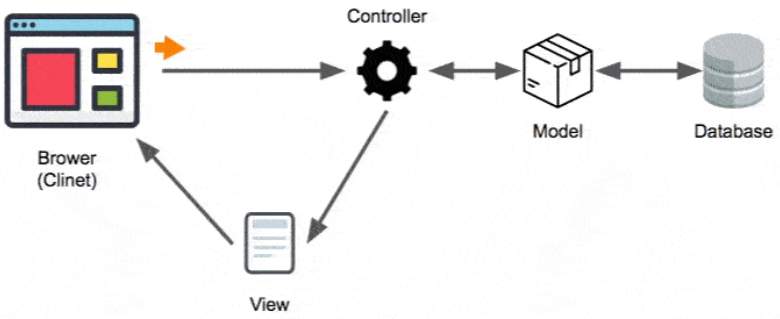


Рис.5 Принцип работы модели MVC

Active Record это M в MVC - модель - которая является слоем в системе, ответственным за представление бизнес-логики и данных. (Рис.6)

```
class CreatePeopls < ActiveRecord::Migration[6.0]
  def change
    create_table :peopls do |t|

      t.string :fn
      t.string :sn
      t.string :tn
      t.string :dolj
      t.bigint :prof_id
      t.bigint :sert_id

      t.timestamps

    end
    add_foreign_key :peopls, :profs
    add_foreign_key :peopls, :serts


  end
end
```

Рис.6 Пример содержимого модели

Шаблоны Action View пишутся с помощью тегов встроенного Ruby, смешанных с HTML. Для каждого контроллера имеется связанная директория в директории app/views, содержащая файлы шаблонов, которые формируют вьюхи, связанные с этим контроллером. Эти файлы используются для отображения вьюхи, являющейся результатом каждого экшна контроллера.(Рис.7)

```
<h1>Mine forme</h1>

<table class="table table-dark table-hover">
    <thead> <h3>Персонал</h3>
    <tr>
    <th>Фамилия</th>
    <th>Имя</th>
    <th>Отчество</th>
    <th>Должность</th>
    <th >Профессия</th>
    <th >Сертификат</th>
    <!-- <th >Удалить</th> -->
```

```
</tr>
    </thead>
    <% @peopl1.each do |peopl| %>
    <tbody >
    <td><%=peopl.fn %></td>
    <td><%=peopl.sn %></td>
    <td><%=peopl.tn %></td>
    <td><%=peopl.dolj %></td>
    <td ><%=peopl.name_prof%></td>
    <td ><%=peopl.name_sert %></td>
    </tr>
    </tbody>
    <% end %>
    </table>

    <div>
        <%= link_to "Перейти", peopls_path,
        :class=>'btn-in-home btn mine-btn' %></div>
    </div>
```

Рис.7 Пример содержимого вьюхи

Action Controller это C в аббревиатуре MVC. После того, как роутер определит, какой контроллер использовать для обработки запроса, контроллер ответственен за осмысление запроса и генерацию подходящего ответа. Контроллер можно рассматривать как посредника между моделями и вьюхами. Он делает данные модели доступными вьюхе, так что она может отображать эти данные пользователю, и он сохраняет или обновляет данные от пользователя в модель. (Рис.8)

```
class ProfsController < ApplicationController

  def index @prof = Prof.all end
  def new @prof = Prof.new end
  def show   @prof =Prof.find(params[:id]) end
  def edit @prof = Prof.find(params[:id]) end

  def update
      @prof = Prof.find(params[:id])
       if(@prof.update(prof_params))
         redirect_to profs_path
         else
             render 'edit'
         end
      end
```

```
  def destroy
      @prof = Prof.find(params[:id])
      @prof.destroy
      redirect_to profs_path
  end

  def create
      @prof = Prof.new(prof_params)
      if(@prof.save)
      redirect_to profs_path

      else
          render 'new'
      end
  end

  private def prof_params
      params.require(:prof).permit(:name_prof)
  end
```

Рис.8 Пример содержимого контроллера

Зададим связи моделей. (Рис.9)

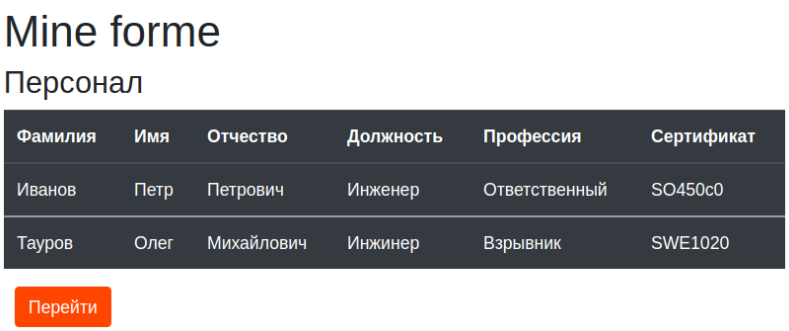
```
class Peopl < ApplicationRecord
    has_one :profs
    has_many :serts

end
```

```
class Prof < ApplicationRecord
    belongs_to :peopls
end
```

```
class Sert < ApplicationRecord
    belongs_to :peopls
end
```

Рис.9 Связывание моделей связью

Заполним данными таблицы. Результат представлен на рис. 10



Ввод

Таким образом можем сделать вывод, что проектирование Web-приложений на ROR Обеспечивает простоту постройки структуры сайта, визуальную «понятность» приложения, а так же легкость присоединения БД к сервису, в чем мы лично убедились на примере объединения нескольких таблиц в одну с соответствующем условием выборки.

Список литературы

1. Руководства предназначены для того, чтобы вы сразу работали с Rails и помогли понять, как все части сочетаются друг с другом. https://guides.rubyonrails.org/active_record_migrations.html
2. Русское пособие по ror. http://rusrails.ru/
3. Ознакомление с python. https://python-scripts.com/django-obzor Ознакомление с python.
4. Официальный сайт wordpress. https://ru.wordpress.org/
5. Официальный сайт 1c-bitrix. https://dev.1c-bitrix.ru/