# Архитектура веб приложения «Пирамида»

Веб приложение «Пирамида» организует взаимодействие между тремя компонентами (рис. 1).



Рис. 1. Архитектура веб приложения «Пирамида»

Клиенту предоставляется веб интерфейс с определенным его ролью набором функций. Веб сервер принимает запросы пользователя, взаимодействует с ВУ посредством ssh соединения и возвращает ответ клиенту.

Рассмотрим подробнее клиентскую и серверную компоненты, а также их взаимодействие.

**Ruby on Rails**

Взаимодействие клиента и сервера строилось в соответствии с архитектурой RESTful. REST API подразумевает унифицированное представление данных на сервере. Сервер предоставляет клиентским приложениям единый интерфейс доступа к данным с некоторым ограниченным набором действий. Таким образом, серверная логика значительно упрощается, делая серверное приложение более масштабируемым и надежным. Упрощение логики происходит, в том числе, из-за делегирования значительной части вычислений на клиентские приложения.

Серверная логика реализовывалась с использованием MVC фреймворка Ruby on Rails.

Ruby on Rails имеет стандартные средства автоматического создания REST API к моделям данных в форматах JSON и XML. В качестве основного формата данных REST API автором работы выбран JSON.

Автоматически создаваемый интерфейс представляет из себя ряд действий контроллера модели, доступ к которым осуществляется через URL в форматах:

GET /имя\_модели.json – выборка всех данных модели.

GET /имя\_модели/идентификатор\_экземпляра.json – выборка одного экземпляра модели по идентификатору.

POST /имя\_модели.json – создание нового экземпляра модели.

PATCH/PUT / имя\_модели/идентификатор\_экземпляра.json – редактирование экземпляра модели по идентификатору.

DELETE /имя\_модели/идентификатор\_экземпляра.json – удаление экземпляра модели по идентификатору.

Таким образом, создается полный набор действий REST API. Разработчик также освобождается от взаимодействия с базой данных.

**Angular.js**

Клиентская часть веб приложения реализована в виде нескольких SPA (single page application), каждое из которых представляет собой angular application. Архитектура SPA позволяет сократить взаимодействие с сервером следующим образом. Клиентское приложение на этапе запуска запрашивает с сервера представление и данные в количестве необходимом для самостоятельной работы некоторого функционального модуля приложения. Такой подход загромождает клиент, традиционно предназначенный лишь для предоставления пользовательского интерфейса. Разбить такое клиентское приложение на модули позволяют MVC, MVVM и MVW фреймворки. На сегодняшний день их число не велико по причине новизны вышеописанной концепции. Наиболее распространенным среди них является Angular.js от Google inc.

Одна из основных целей разрабатываемого приложения – отображение в динамике процесса вычислений. Реализация такой возможности средствами стандартного стэка JavaScript HTML CSS представляла бы из себя код, построенный на множестве функций обратного вызова. Такой код обладает низкими надежностью, производительностью и масштабируемостью. Автором разрешена эта проблема применением встроенного в Angular.js механизма двустороннего связывания данных.



Рис. 2. Архитектура Angular приложения

Механизм двусторонней привязки данных автоматически синхронизирует состояние данных представления, модели клиента (сервиса) и модели сервера. Синхронизация осуществляется путем обмена данными в формате JSON.

# Практические результаты первой части преддипломной практики

В соответствии с архитектурой веб приложения «Пирамида» работу можно разбить на 2 части (разработка испытательного стенда была завершена ранее):

* разработка серверной части веб приложения, в том числе организация взаимодействия с испытательным стендом по ssh;
* разработка клиентской части приложения, в том числе разработка пользовательского интерфейса.

**Разработка серверной части веб приложения**

Установка Ruby on Rails.

Ruby on Rails (RoR) - фреймворк для веб-разработки в рамках шаблона проектирования MVC, написанный на языке программирования Ruby. Для его работы был установлен интерпретатор Ruby, а также установлена и сконфигурирована СУБД PostgreSQL.

Взаимодействие веб сервера с испытательным сервером.

В качестве средства создания ssh подключения автором была выбрана библиотека (gem) “net-ssh” с открытым исходным кодом. Реализован контроллер, тестирующий ssh соединение с испытательным стендом. Контроллер принимает GET запрос клиента, создает ssh соединение с испытательным стендом, отправляет ответ клиенту в формате JSON о состоянии соединения.

Модели данных.

Начальная версия веб приложения «Пирамида» должна реализовывать следующий функционал:

* создание/удаление/редактирование ssh подключения по заданным пользователем параметрам;
* сохранение на сервере параметров подключения;
* сохранение пароля в локальном хранилище пользователя;
* тестирование состояния выбранного подключения;
* формирование паспорта задания ПК «Пирамида» по заданным пользователем параметрам;
* сохранение паспорта задания на сервере.

В соответствии с вышеперечисленными требованиями выделены 2 сущности: SshConnection и TaskTemplate.

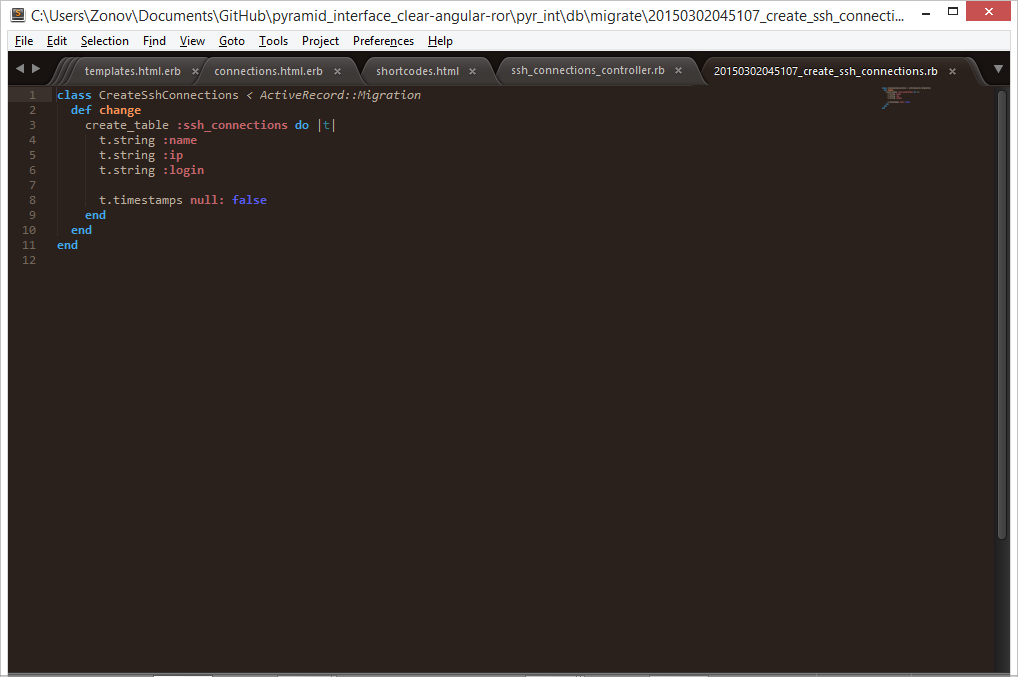


Рис. 3. Модель SshConnection

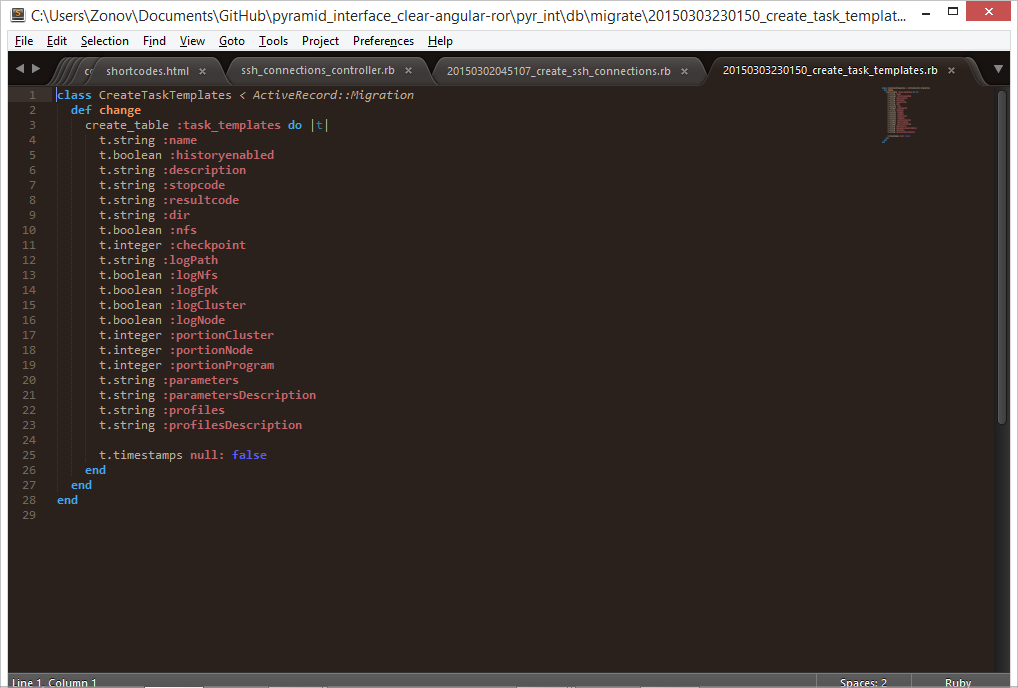


Рис. 4. Модель TaskTemplate

В соответствии с выбранной архитектурой взаимодействия клиент-серверного приложения RESTful для обеих моделей создан полноценный REST интерфейс в формате JSON.

Для работы Angular приложения на сервере создана необходимая иерархия каталогов, а также настроена маршрутизация.

**Разработка клиентской части веб приложения**

Начальная версия веб приложения «Пирамида» разработана в виде одного SPA приложения, состоящего из страничных модулей «Шаблоны» и «Соединения». Весь функционал содержится в ng-модуле “pyramidApp”. Модуль содержит сервис маршрутизации, а также соответствующие сервисы данных и контроллеры страничных модулей.