# BÁO CÁO BÀI TẬP NHÓM

# Các thành viên trong nhóm

|  |  |
| --- | --- |
| **Họ và tên** | **Mã sinh viên** |
| Nguyễn Thành Đạt | 11020069 |
| Ngô Quốc Thắng | 12020360 |
| Vũ Văn Đạt | 12020084 |
| Nguyễn Trung Đức | 12020103 |
| Nguyễn Văn Tuân |  |

# Giới thiệu

## Giới thiệu chung

Ruby on Rails hay Rails là một framework được xây dựng dựa trên ngôn ngữ Ruby theo giấy phép MIT. [David Heinemeier Hansson](https://en.wikipedia.org/wiki/David_Heinemeier_Hansson)  đã phát triển Rails từ công việc của mình trên công cụ quản lý dự án Basecamp. Rails là một framework Model-View-Controller cung cấp các cấu trúc mặc định cho cơ sở dữ liệu, web service và web page. Nó khuyến khích và tạo điều kiện sử dụng các tiêu chuẩn của web như Json hoặc XML để truyền dữ liệu và HTML, CSS và JavaScript để hiện thị . Ngoài ra, Rails còn sử dụng một số mẫu thiết kế và khuôn mẫu bao gồm  [convention over configuration](https://en.wikipedia.org/wiki/Convention_over_configuration" \o "Convention over configuration) (CoC), [don't repeat yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself) (DRY), and the [active record pattern](https://en.wikipedia.org/wiki/Active_record_pattern). Nó cho phép bạn viết ít code hơn lại làm được nhiều hơn so với ngôn ngữ và framework khác.

## Ưu và nhược điểm của Rails

### Ưu điểm

* Quá trình lập trình nhanh hơn nhiều so với framework và ngôn ngữ khác, bởi vì bản chất của lập trình hướng đối tượng của Ruby và các bộ collection lớn của mã nguồn mở có sẵn trong cộng đồng Rails.
* Các convension(qui ước) làm dễ cho các nhà phát triển để di chuyển giữa các dự án khác nhau của Rails, với mỗi dự án có xu hướng theo cùng cấu trúc và code thực tế.
* Tốt cho việc phát triển các ứng dụng nhanh, làm dễ dàng cho phù hợp với thay đổi .
* Code Ruby rất dễ đọc và tự bản thân code có thể làm tài liệu. Điều này làm tăng hiệu suất vì cần viết ít tài liệu, đồng thời dễ dàng phát triển các dự án có sẵn.
* Phát triển trọng tâm về kiểm thử và là một framework tốt về kiểm thử
* Rails và phần lớn thư viện là mã nguồn mở.

### Nhược điểm

* Không phải tất cả các host hỗ trợ Rails. Phần lớn đều hỗ trợ PHP, tuy nhiên Rails cũng thân thiện với các host có tồn tại như Heroku và EngineYard
* Java và Php được sử dụng rộng rãi hơn, có nhiều nhà phát triển trên những ngôn ngữ đó.
* ứng dụng Rails chạy không được nhanh như Java hoặc C nhưng đa số các ứng dụng chạy đủ nhanh.
* Đối với người mới học Rails thì có vẻ rất phức tạp vì có nhiều quy ước nhưng sau một thời gian thì các quy ước này lại trở nên thuật tiện cho việc phát triển.
* Khó triển khai hơn PHP.

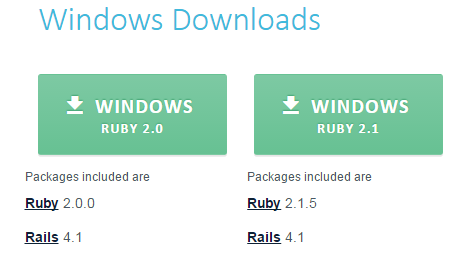
## Khi nào nên dung Rails

* Cần code cơ sở và cơ sở dữ liệu mới.
* Tốc độ phát triển nhanh.
* Phát triển ứng dụng phức tạp.
* Triển khai web
* Dự án cần tương tác với khách hang.
* Chi phí phát triển đắt đỏ.
* Giá của dự án thay đổi.
* Sớm nhận client/team

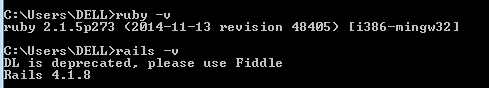
# Cài đặt

Đây là hướng dẫn cài đặt rails, slim, sửa một số lỗi và chạy thử 1 project trên windows.

* Download bộ cài đặt tại <http://railsinstaller.org/en>. Người viết chọn phiên bản “Windows Ruby 2.1”

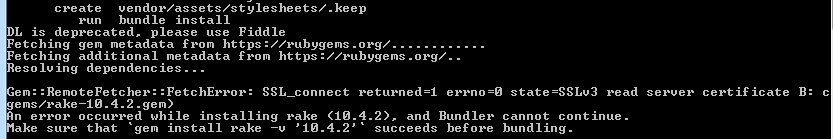


* Chạy file vừa tải về, cài đặt trong ít phút.
* Vào cmd, gõ “ruby –v” và “rails –v” để kiểm tra. Nếu thông tin phiên bản hiện ra thì ruby và rails đã được cài đặt thành công.



* Để tạo 1 project tên “demo”, gõ “rails new demo”.

1 lỗi hay gặp có thể xuất hiện khi tạo project



Để khắc phục, ta vào folder của project demo vừa tạo, edit file có tên “Gemfile”, sửa dòng đầu tiên

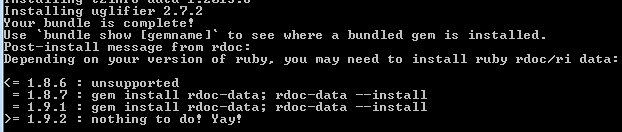
*source 'https://rubygems.org'*

thành

*source 'http://rubygems.org'*

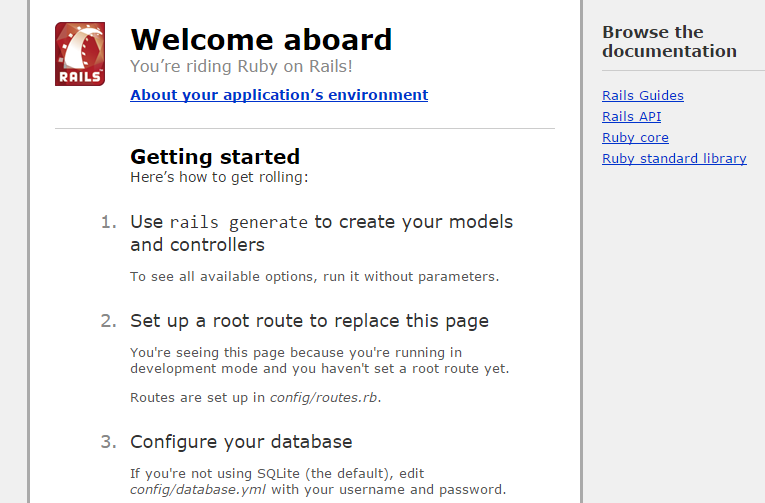
rồi save.

* Quay lại cmd, ta “cd demo” để tới folder của project. Gõ “bundle install” để tiếp tục cài các thành phần của project (mà trước đó bị lỗi không cài được). Như trong hình, project đã được khởi tạo hoàn thiện

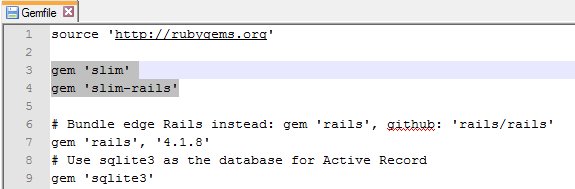


*\*Có thể có những lỗi phát sinh khác vì windows không phải là môi trường tốt nhất cho rails, bạn sẽ phải google liên tục để khắc phục*

* Tiếp theo, để chạy thử project trên trình duyệt web, tại folder demo ta gõ “rails server” hoặc “rails s”. Sau đó, đi tới địa chỉ “localhost:3000” trên trình duyệt. Giao diện welcome sẽ xuất hiện như hình dưới.



* Để cài đặt *slim* cho project, ta mở file “Gemfile” trong project demo, thêm 2 dòng sau (bôi đậm):



rồi save.

Quay lại cmd, ấn Ctrl+C và Y, Enter để tắt server. Sau đó tương tự như đã làm, ta chạy lệnh “bundle install” để cài đặt 2 gem vừa ghi trong Gemfile.

* Khi không muốn sử dụng slim nữa, ta bỏ 2 dòng vừa thêm trong Gemfile, sau đó chạy lại “bundle install”. Kiểm tra gem đang được sử dụng trong project bằng lệnh “bundle list”.

# Kiến trúc của Rails

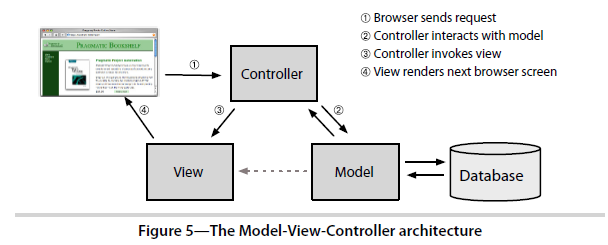
## Mô hình MVC

Trở lại năm 1979, Trygve Reenskaug giởi thiệu một kiến trúc mới trong phát triển ứng dụng tương tác. Trong thiết kế của ông ý, ứng dụng được chia ra làm 3 thành phần : models, views và controller.

Model thì chịu trách nhiệm duy trì trạng thái của ứng dụng. Đôi khi chỉ là thoáng qua, chỉ là một vài tương tác với người sử dụng. Thỉnh thoảng trạng thái cố định sẽ được lưu ra ngoài, thường cái này chính là cơ sở dữ liệu. Model không chỉ là dữ liệu, nó thực thi tất cả các quy định kinh doanh áp dụng vào dữ liệu. Việc code thực hiện trong model thì chúng tôi có thể đảm bảo rằng không có cái gì khác trong ứng dụng có thể làm cho dữ liệu của chúng ta không hợp lệ. Model hoạt động như một người gác cửa và lưu trữ dữ liệu.

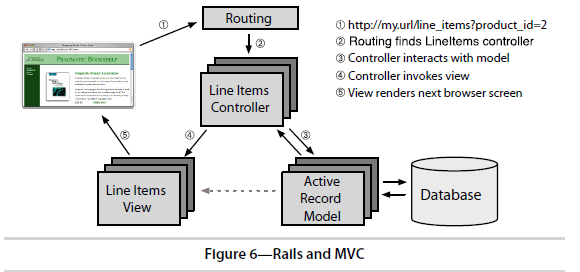
View thì chịu trách nghiệm tạo ra giao diện người dùng, giao diện được tạo ra dựa trên dữ liệu trong model. Mặc dù view hiện thị cho người dùng xem nhưng không bao giờ xử lý dữ liệu đi tới. Công việc của view hoàn thành khi đã hiện thị dữ liệu thành công. Có thể có nhiều view cùng truy cập chung một dữ liệu model, với các mục đích khác nhau.

Controller thì điều khiển ứng dụng. Các controller nhận dữ liệu từ thế giới bên ngoài(thường là người dùng) rồi tương tác với model và làm view hiện thị thích hợp với người dùng.



Kiến trúc MVC ban đầu thường được dùng đế phát triển cho ứng dụng có giao diện thông thường, đây là nơi mà các nhà phát triển ít lo lắng về việc tách biệt và nối các thành phần, làm code việc viết code và bảo trì dễ dàng hơn. Rails chú trọng đến cấu trúc – bạn phát triển các model, view, controller như các khối riêng biệt của chức năng , rồi chúng đan vào nhau để thực thi. Quy trình đan này dựa trên việc sử dụng thông minh có sẵn nên bạn không phải cấu hình thêm ở bên ngoài.

Trong ứng dụng của Rails, thì khi yêu cầu tới đầu tiên nó sẽ được giửi tới router, router sẽ hoạt động để tìm trong ứng dụng nơi yêu cầu sẽ giửi và phân tích yêu cầu. Cuối cùng, nó sẽ xác định phương thức cụ thể gọi là action trong code controller. Action này tìm dữ liệu trong yêu cầu, nó có thể tương tác với model và có thể gọi đến 1 action khác. Sau cùng, action chuẩn bị thông tin cho view, nó sẽ vẽ cho người dùng.



## Rails Model

Nói chung, chúng ta muốn các ứng dụng web của mình lưu trữ thông tin trong một cơ sở dữ liệu quan hệ. Mặc dù cơ sở dữ liệu quan hệ là tốt cho nhìn vào nhưng khi kết hợp cơ sở dữ liệu quan hệ với ngôn ngữ lập trình hướng đối tượng rất khó khan. Đối tượng bao gồm cả dữ liệu và hành động còn cơ sở dữ liệu chỉ là tập các giá trị.

Trong Rails sử dụng **Action Record** để kết hợp giữa ngôn ngữ Ruby với cơ sở dữ liệu quan hệ. Object-Relational Mapping(ORM) dùngđể ánh xạ từ bảng cơ sở dữ liệu tới các lớp, **Action Record**bám sát theo các tiêu chuẩn mô hình ORM: bảng ánh xạ tới lớp, các dòng ánh xạ tới các đối tượng, các cột ánh xạ tới thuộc tính của đối tượng. **Action Record** khác các thư viện ORM trong cách cấu hình, bằng cách dựa vào các quy ước nó làm giảm công việc cấu hình của các nhà phát triển. Cũng làm giảm sự phức tạp của giao dịch với cơ sở dữ liệu, khiến chúng ta tự do làm việc trên business logic. Nó là model nền móng trong kiến trúc của Rails

## Action Park : View và Controller

Khi bạn nghĩ về View và Controller là một phần khá quan trọng trong MVC. Controller cung cấp cho View dữ liệu và nhận dữ liệu từ các trang view, chính vì lý do này nên Rails đã gói vào làm một gọi là Action Park.

Trong Rails, View chịu trách nhiệm tạo tất cả hoặc 1 phần của phản hồi để được hiện thị trên trình duyệt, xử lý một ứng dụng hoặc giửi 1 email. Nó chỉ đơn là một đoạn code HTML mà hiện thị một vài text nhất định. Thông thường, nội dung động được tạo bởi các action method trong controller. Nội dung động này được tạo một bởi các mẫu, phần lớn lược đồ mẫu gọi là nhúng Ruby(ERB). Bạn có thể ERB để cấu trúc mảng JavaScript trên server mà có thể xử lý trên trình duyệt. Rails cũng cung cấp XML Builder để cấu trúc tài liệu XML sử dụng code Ruby – cấu trúc tạo XML sẽ tự động theo cấu trúc của code.

Controller trong Rails là trung tâm logic của ứng dụng. nó tương tác giữa người dung, view và model. Tuy nhiên, Rails hầu hết xử lý các tương tác ở đằng sau, code của bạn cô đọng về mức chức năng của ứng dụng, điều này làm cho phát triển và bảo tri trở nên dễ ràng hơn. Controller cũng là nơi mà một số dịch vụ quan trọng xử lý: định tuyến các yêu cầu bên ngoài để hoạt động nội bộ, quản lý caching, quản lý mô đun helper, mở rộng các khả năng view mẫu, quản lý session.

# Ứng dụng đầu tiên

## Triển khai chạy trên Internet sử dụng Heroku

### Giới thiệu

Heroku là một nền tảng đám mây (cloud platform) giúp xây dựng và triển khai các ứng dụng web. Nó được mô tả là “cách nhanh nhất để đi từ ý tưởng đến URL, bỏ qua mọi rắc rối về vấn đề cơ sở hạ tầng. Bạn chỉ việc tập trung vào ứng dụng của bạn”.

### Cài đặt và sử dụng

Heroku đã hướng dẫn khá kĩ tại đây.

<https://devcenter.heroku.com/articles/getting-started-with-ruby#introduction>

Thực hiện theo hướng dẫn, và sửa lỗi phát sinh thông qua google

## Giao diện

### Giới thiệu Bootstrap

Đây là một front end framework phổ biến nhất hiện nay. Nó bóng bẩy, trực quan, mạnh mẽ và là front end framework đầu tiên hướng tới điện thoại di động. Nó sử dụng HTML, CSS và JavaScript để làm trang web của bạn trong đẹp hơn.

Lợi ích khi sử dụng Bootstrap:

* Tiếp cận theo hướng di động đầu tiên.
* Hỗ trợ tất cả các trình duyệt web phổ biến như Chrome, Fire Fox, Opera, Internet Explorer, Safari …
* Bootstrap dễ học : Chỉ với kiến thức HTML và CSS là bạn có thể bắt đầu với Bootstrap.
* Responsive design : Cách thiết kế này giúp trang web của bạn điều chỉnh giao diện cho máy tính, máy tính bảng và thiết bị di động.
* Đây được coi như là tiêu chuẩn ngầm giữa các nhà phát triển.
* Có thể tùy chỉnh.

### Cài đặt Bootstrap với Rails

Các bước cài đặt:

* Mở file Gem trong ứng dụng
* Thêm 2 gem bootstrap-sass và sass-rails vào file Gem.
* gem ‘bootstrap-sass’, ‘~>3.3.5’
* gem ‘sass-rails’ , ‘>= 3.2’
* chạy lệnh : bundle install
* Import Bootstrap CSS assets : thêm 2 dòng dưới vào đầu file app/assets/stylesheets/application.scss
  + @import "bootstrap-sprockets";
  + @import "bootstrap";
* Import Bootstrap javascript assets : thêm dòng dưới vào file app/assets/javascripts/application.js
  + //= require bootstrap-sprockets

Vào sau dòng

* //= require jquery

## Sign up

* 1. **Xây dựng mô hình dữ liệu User cho chức năng đăng ký**
* Mỗi người dùng có 1 định danh duy nhất được xác định bởi **id**
* Một tên gọi **Name** (thuộc kiểu String)
* Một địa chỉ Email (thuộc kiểu String)

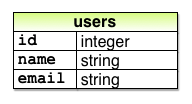
\*Mô hình dữ liệu **User** cho chức năng đăng ký

Figure .The data model for users.

* Lệnh cài đặt dữ liệu
  + $ rails generate scaffold User name:string email:string
  + $ rake db:migrate
* Khởi động server:
  + $ rails server
  1. **Xây dựng ứng dụng theo mô hình MVC**
* Mô hình MVC cho 1 ứng dụng Ruby on Rails

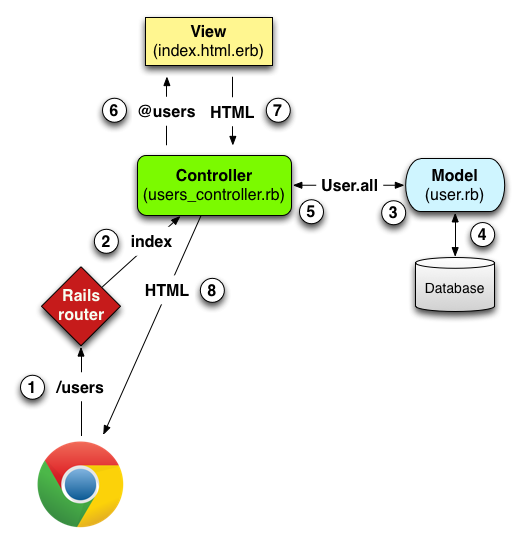
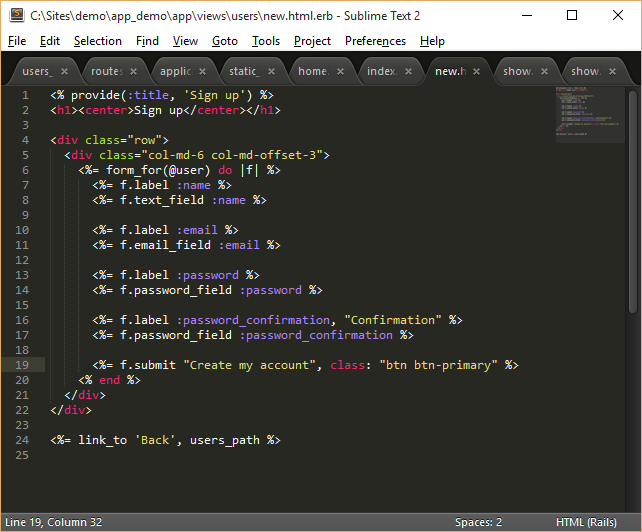
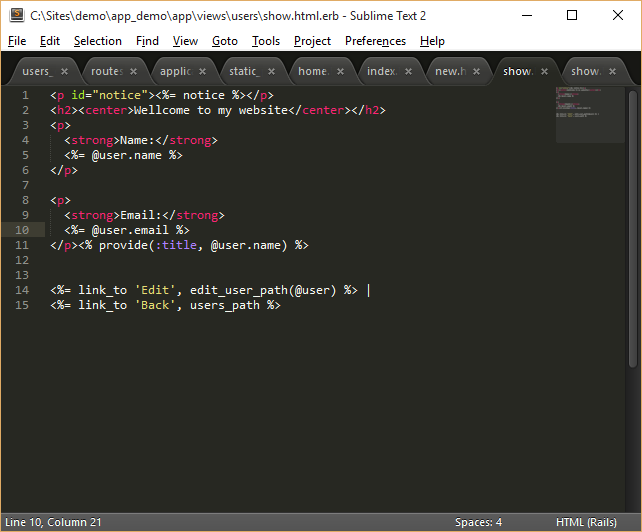
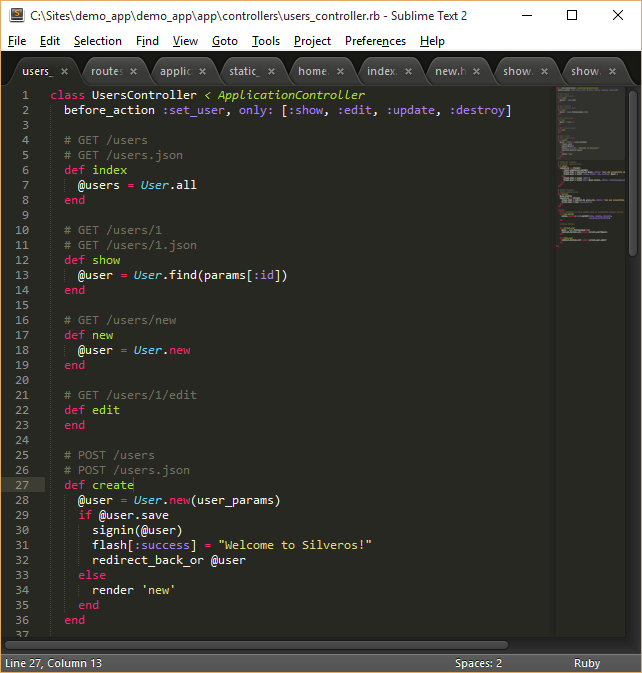
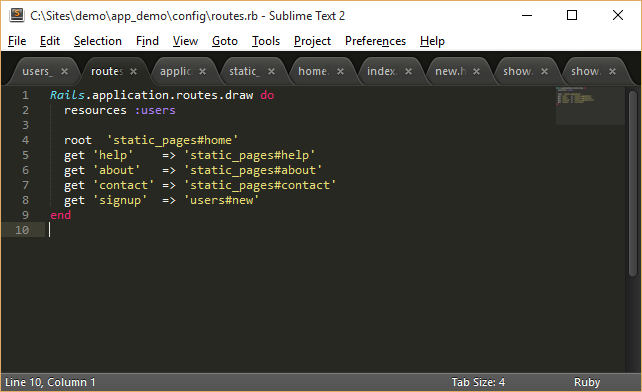


Figure 2.Model MVC Ruby on Rails

* **Tại View:**
  + Xây dựng form biểu cho phép người dùng nhập thông tin đăng ký:
  + Hiển thị thông tin User khi đăng ký thành công
* **Tại Controller:**
  + Xây dựng Controller cho phép thêm người dùng vào hệ thống:



* Điều hướng **config -> routes**
  1. **Giao diện ứng dụng chức năng đăng ký**

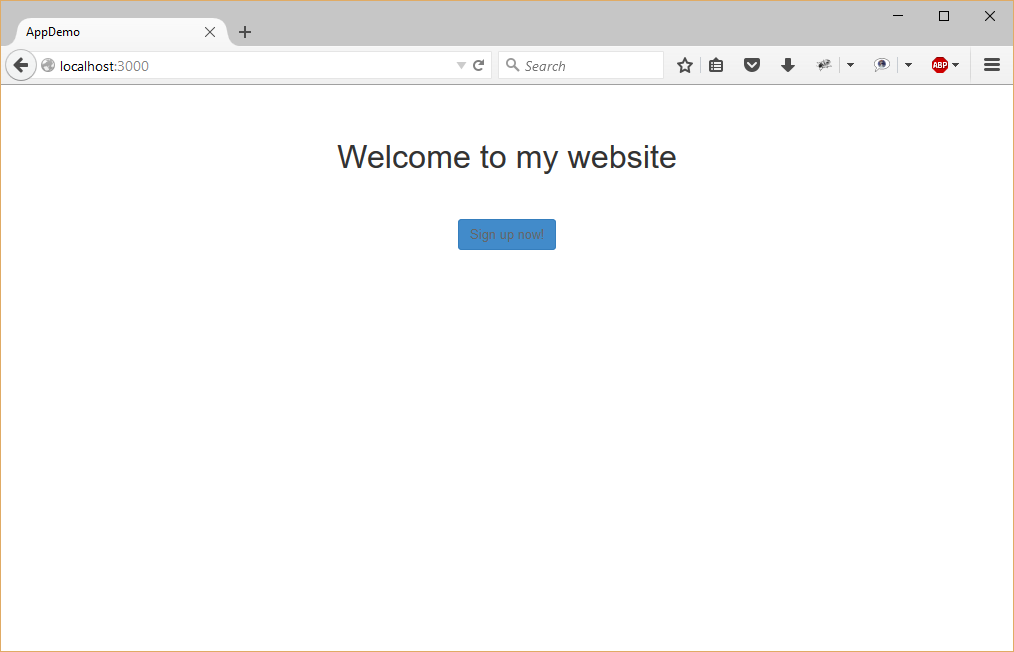


Figure .Home page

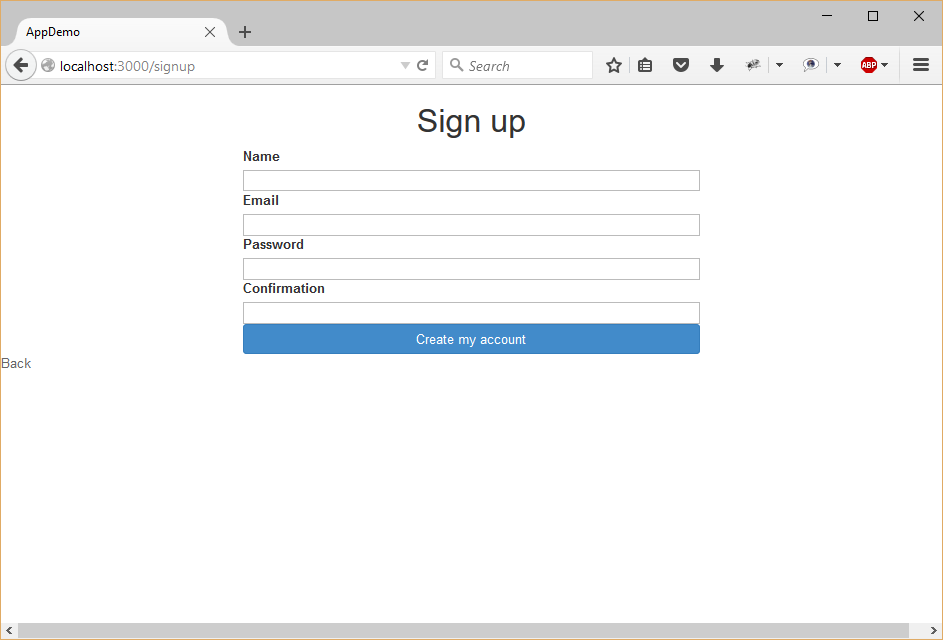
* Click Sign up:

Figure 4.Sign up interface

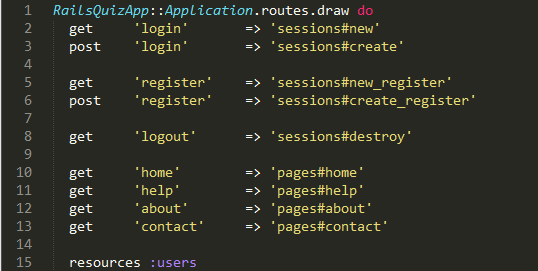
## Login và Logout

### Code

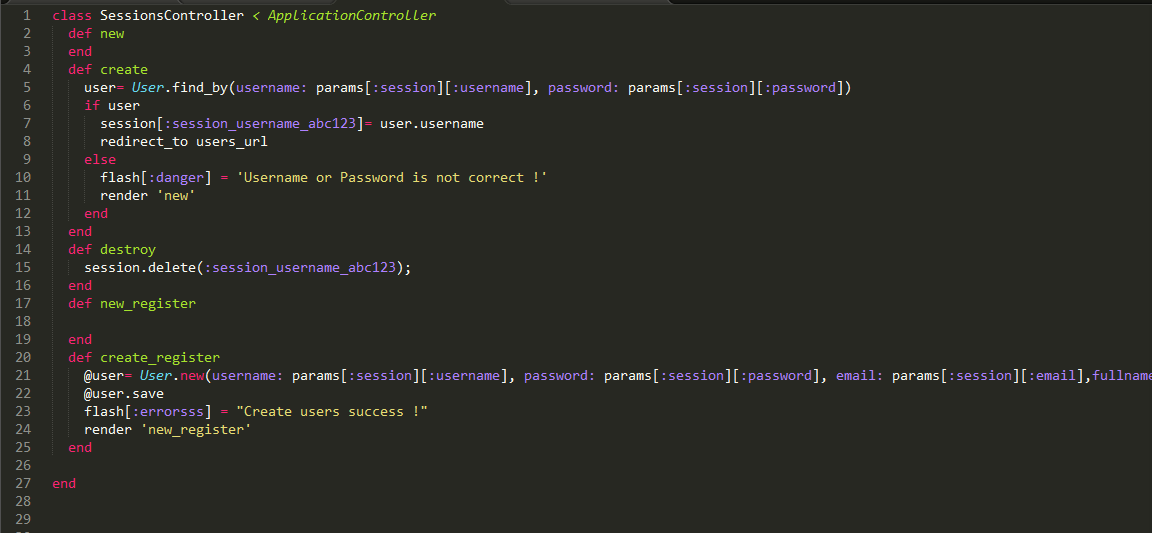
Chúng ta tạo chức năng login và logout: gõ lệnh **rails g controller Sessions new create destroy .** Sauk hi gõ lệnh trên thì tự động rails sẽ sinh code cho chúng ta bao gồm router, controller, view, model và chúng ta sẽ viết code cho các phần này

**Chức năng login:**

* Route: gồm get ‘login’ và post ‘login’

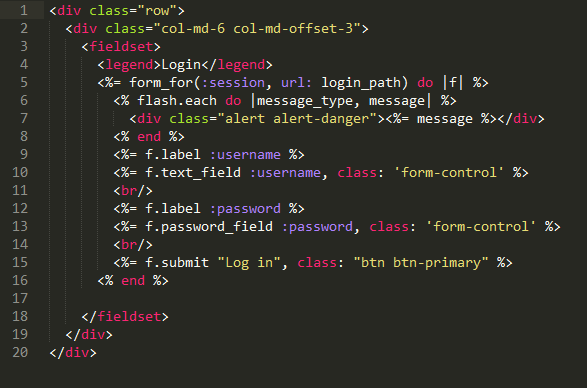


* Controller: (sessions\_controller.rb)



Ở đây phần code xử lý cho chức năng đăng nhập là **create,** chúng ta sẽ lấy dữ liệu từ form gửi lên bao gồm username, password và so sánh dữ liệu này với dữ liệu trong CSDL nếu trùng nhau sẽ đẩy người dùng vào trang quản lý users nếu không chính xác sẽ trả về dòng thông báo lỗi **'Username or Password is not correct !'**

* View: (new.html.erb)



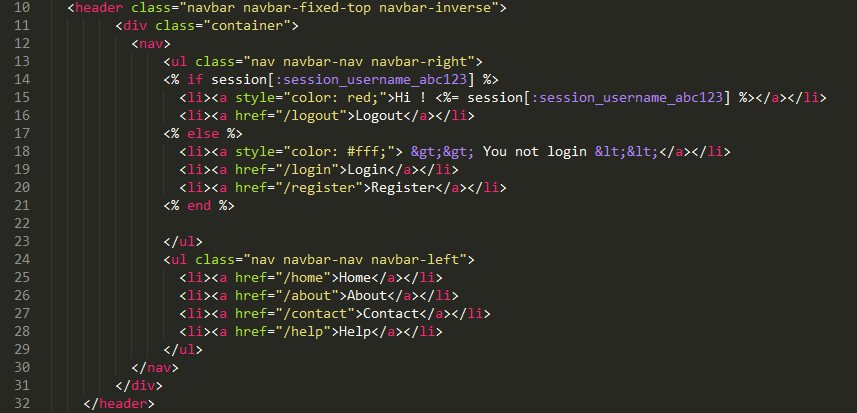
**Chức năng Logout**

Khi người dùng muốn logout khỏi hệ thống, hõ sẽ click chọn nút logout để đăng xuất. Lúc trước khi người dùng đăng nhập chúng ta đã tạo một biến sessions để lưu username của người dùng và hiển thị nó trên phần header bây giờ khi đăng xuất chúng ta sẽ hủy session này đi và sẽ hiển thị lại nút login để đăng nhập

* Xử lý đăng xuất:

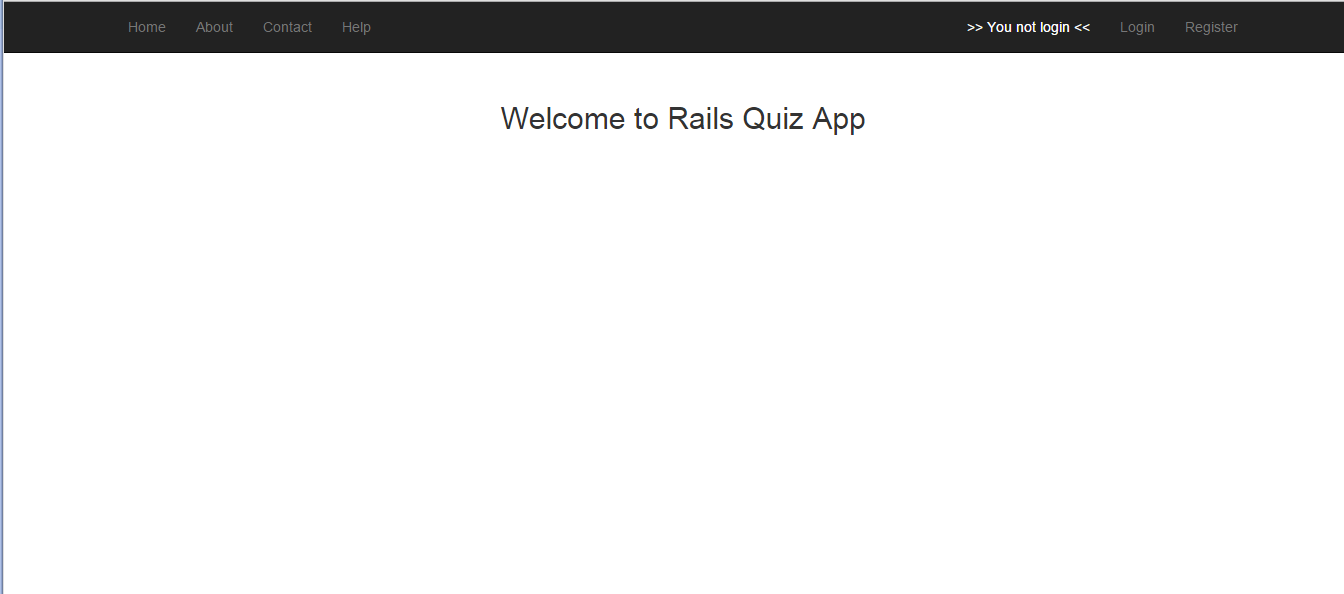
C:\Users\Admin\Desktop\ádas.PNG

* Code header(layouts/application.html.erb):

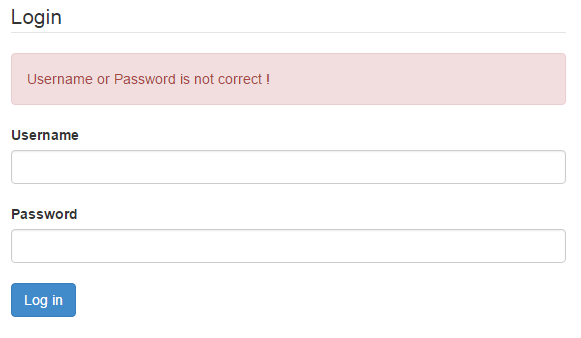


### Giao diện

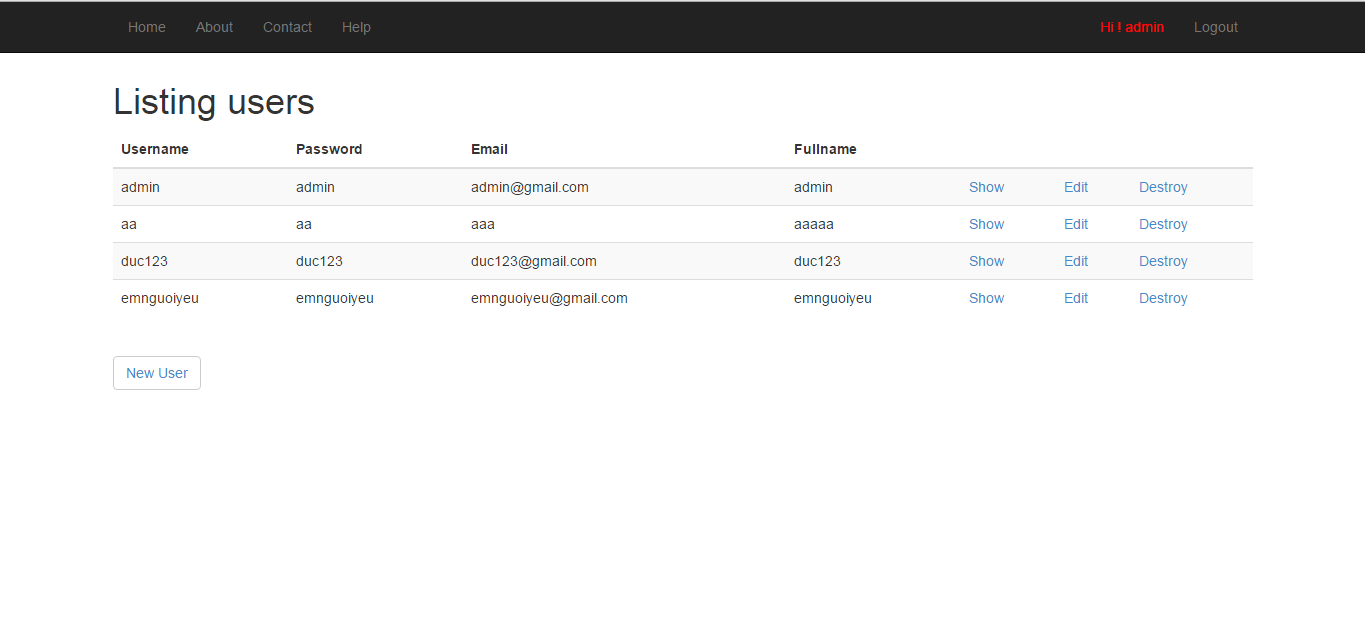
* Trang chủ:



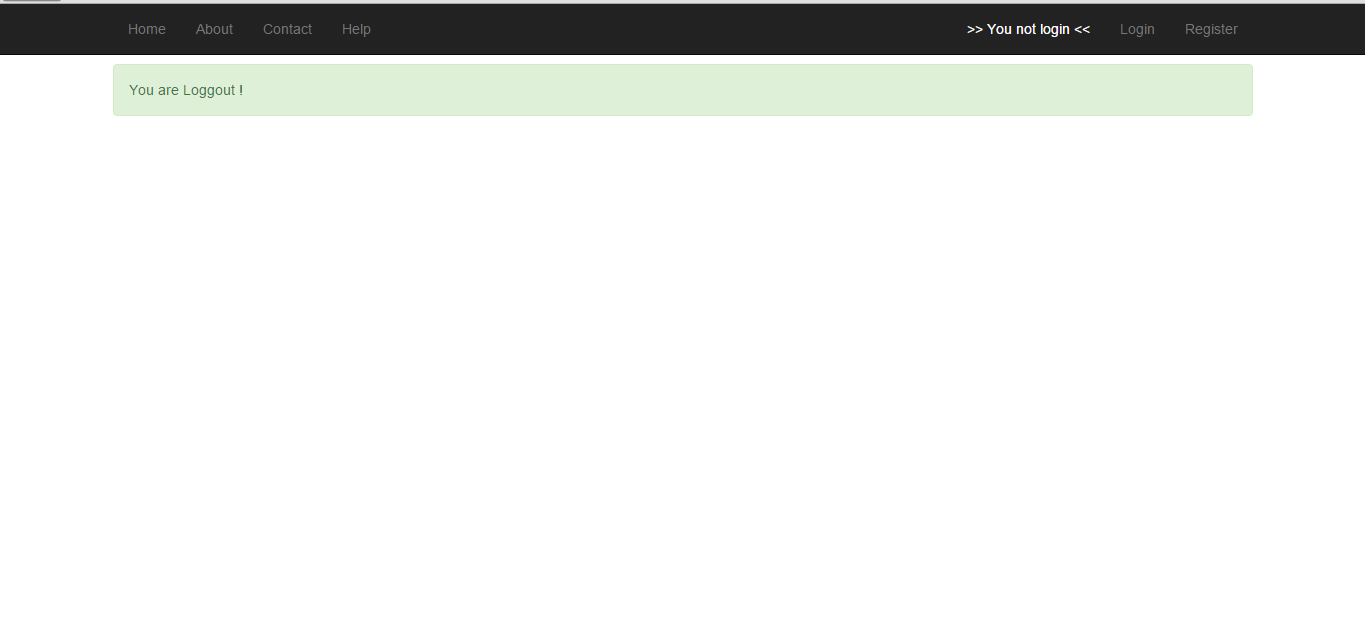
* Trang login:



* Trang users:



* Logout:

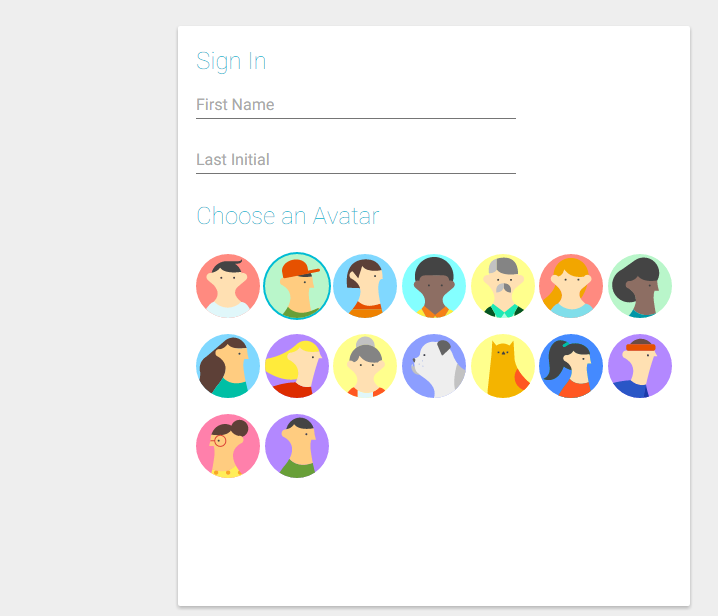


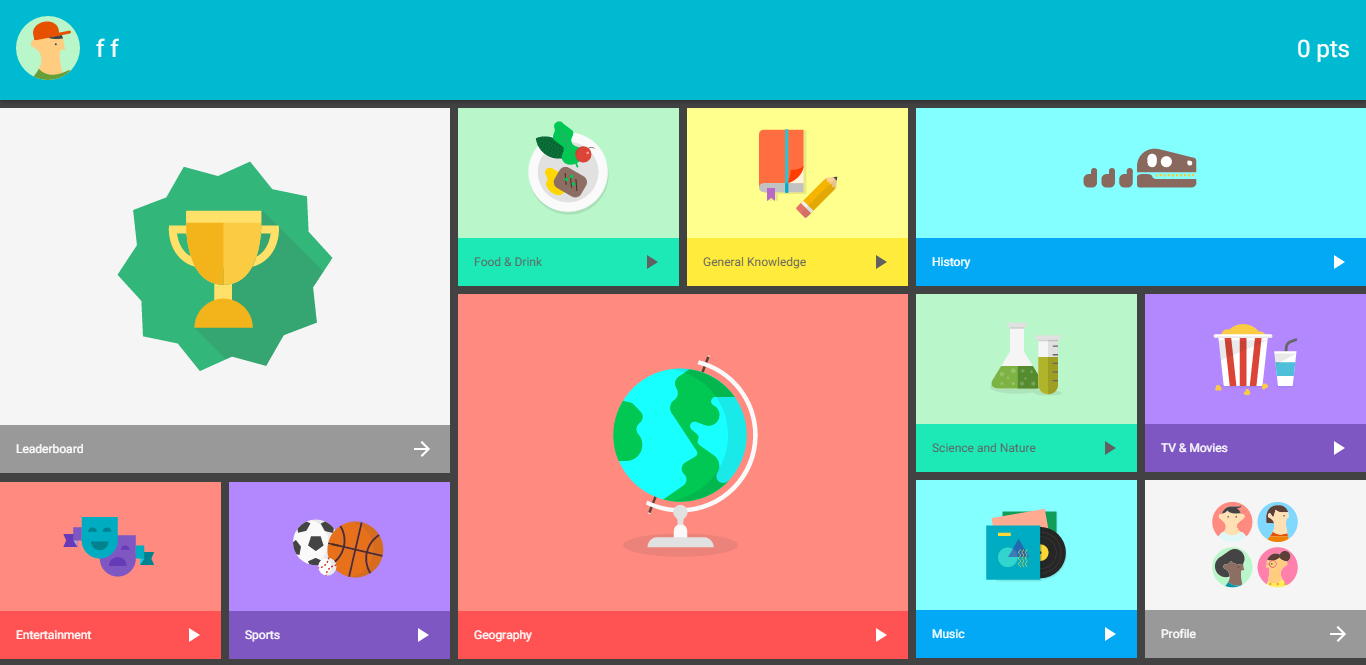
1. **Xây dựng ứng dụng trả lời câu hỏi sử dụng Rails**
2. **Phát biểu bài toán**

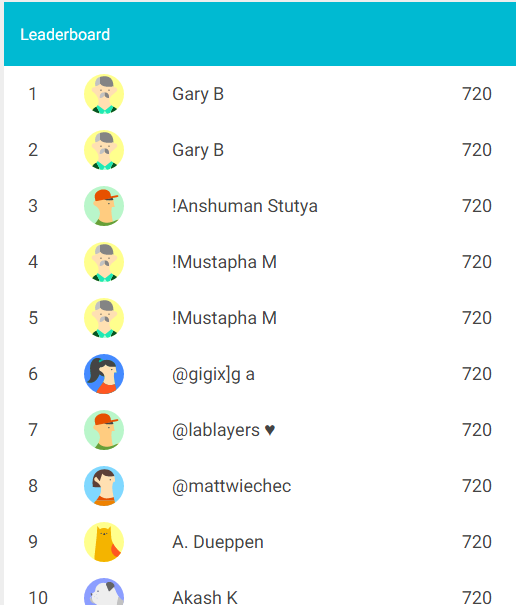
Xây dựng một ứng dụng web sử dụng Rails. Ứng dụng cho phép người chơi trả lời câu hỏi (Quiz) thuộc nhiều lĩnh vực như: thức ăn & đồ uống, lịch sử, nhạc, TV & phim, kiến thức chung, khoa học và tự nhiên, thể thao, giải trí.

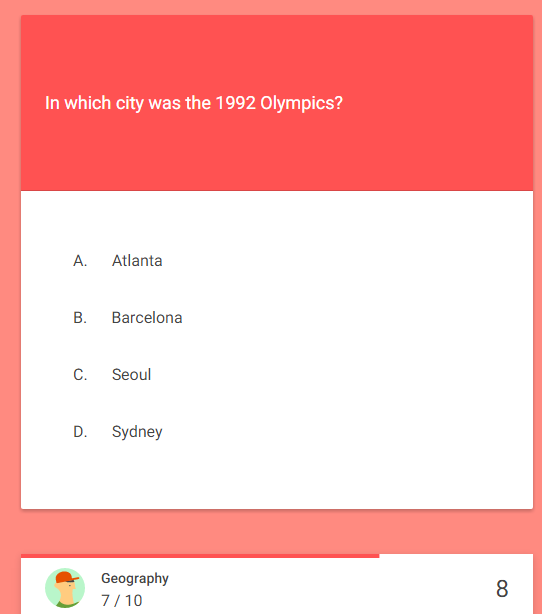
Trong mỗi 1 lần chọn lĩnh vực người chơi sẽ có 10 câu hỏi, sau khi trả lời xong số điểm của người chơi sẽ được tính, mỗi câu trả lời đúng sẽ được tính 1 điểm.

Dựa vào Quiz App (*polymer-topeka.appspot.com*) để xây dựng. Dưới đây là một số hình ảnh về Quiz App này:









1. **Xác định yêu cầu**

* Ứng dụng sẽ có chức năng: đăng ký, đăng nhập, đăng xuất, sửa thông tin dành cho người dùng.
* Admin có thể thêm, sửa, xóa câu hỏi, câu trả lời.
* Ứng dụng tự động sinh ra 10 câu hỏi trong lĩnh vực người dùng chọn.
* Tự động tính điểm và thống kê 10 người có điểm cao nhất.
* Sử dụng framework Rails.

1. **Giải pháp**
2. **Chiến lược phát triển:**

Chiến lược phát triển ứng dụng được lựa chọn đó là *Phát triển tăng dần*. Cụ thể là:

* Phát triển dựa trên ý tưởng.
* Làm đủ yêu cầu kỹ thuật rồi tạo các chức năng.
* Thử những ý tưởng rồi thu thập thông tin phản hồi và tiếp tục phát triển.
* Đòi hỏi có sự hợp tác chặt chẽ của người dùng.
* Trong quá trình phát triển, việc thường xuyên show cho khách hàng quan sát có thể giúp khách hàng xác định cái thực sự mình mong muốn. Điều này làm giảm công sức và chi phí phát sinh.

1. **Giải pháp cụ thể**

Quy trình phát triển ứng dụng gồm 3 bước như sau:

* **Xác định các use case**

Use case là danh sách các hành động hoặc các bước sự kiện thể hiện sự tương tác giữa tác nhân và hệ thống.

Dựa vào phần xác định yêu cầu của ứng dụng Quiz App đã nêu ở trên, ta có thể xây dựng biểu đồ use case cho Quiz App (như ở hình dưới). Thực tế ứng dụng có thể không tuân theo biểu đồ này 100% nếu có sự thay đổi phù hợp hơn.



Biểu đồ use case cho Quiz App

* **Xác định luồng làm việc giữa các trang**
  + Khi làm một trang web chúng ta phải luôn hiểu người dùng sẽ di chuyển giữa các trang như thế nào, từ đó thiết kế chúng một cách hợp lí.
  + Giai đoạn đầu của phát triển tất nhiên chưa thể đầy đủ, hợp lí được. Chúng ta nên chú trọng vào những thứ quan trọng cần làm ngay và phác họa trình tự các hành động mà hiện tại ta thấy hợp lí.
  + Để làm việc này chúng ta có thể vẽ tay hoặc vẽ trên máy bằng những phần mềm chuyên dụng.

Dưới đây là sơ đồ luồng làm việc giữa các trang ở mức đơn giản. Một số hướng đi giữa các trang được bỏ qua do đã được nhắc đến, nhằm làm sơ đồ dễ nhìn hơn.



* **Xác định thiết kế cơ sở dữ liệu** 
  + Có thể chưa cần sử dụng sơ đồ đầy đủ, chi tiết ở đây. Cũng chưa cần nói về tables, keys, …Đây có thể chỉ là thiết kế ban đầu của cơ sở dữ liệu, vì trong quá trình phát triển ứng dụng nó sẽ phải được điều chỉnh phù hợp hơn.
  + Được xây dựng dựa trên use case và các luồng làm việc

Từ những điều trên và quá trình thực hiện, ta xác định rằng cơ sở dữ liệu cho Quiz App sẽ gồm các bảng chính sau:

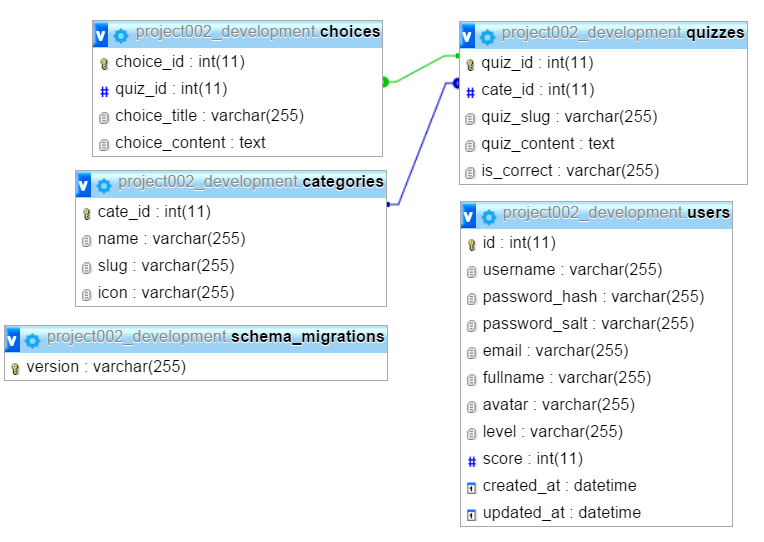
**Users:** lưu thông tin người dùng

**Quizzes:** lưu thông tin các câu hỏi

**Categories:** lưu thông tin về các chủ đề của câu hỏi

**Choices:** lưu thông tin các câu trả lời của câu hỏi

Từ đó, thiết kế chi tiết cho cơ sở dữ liệu được xây dựng, kết quả là sơ đồ như hình dưới:



Thiết kế cơ sở dữ liệu

* Ngoài ra, ta cần chú ý Rails là một back-end framework chỉ chuyên xử lý phần server chứ không chú trọng giao diện. Khi xây dựng ứng dụng web Rails thường dùng kèm theo front-end framework để tăng sự tương tác với người dùng. Ở đây ta sử dụng Bootstrap (đã đề cập ở phần V.2)

# Tài liệu tham khảo

1. Ebook : Agile web development with ruby on rails 4 by Sam Ruby.
2. <http://bitzesty.com/2014/01/10/ruby-on-rails-what-it-is-and-why-we-use-it-for-web-applications/>
3. <http://www.slideshare.net/dosire/when-to-use-ruby-on-rails-1308900>
4. <http://guides.rubyonrails.org/getting_started.html>
5. <https://en.wikipedia.org/wiki/Ruby_on_Rails>
6. <https://www.railstutorial.org/book>