Ruby on Rails

Frank Benavides Murillo
Tecnológico de Costa Rica
Cartago, Costa Rica
Email: fjos807@gmail.com

Arturo Chaves Vargas
Tecnológico de Costa Rica
Cartago, Costa Rica
Email: artch92@gmail.com

O. Gabriel Quirós Obando
Tecnológico de Costa Rica
Cartago, Costa Rica
Email: gabrielquiros10@gmail.com

***Abstract*— The following investigation aims to have an overview of the framework Ruby on Rails (RoR), which is one of the most popular frameworks to develop web applications. RoR is based on programming language Ruby and is used to develop the back-end in a web application. The investigations is focused on the main features that make this framework popular, but also on the history and some basic concepts that are necessary in order to create and understand a RoR project.**

I. Introducción

Desde el inicio del desarrollo web para la creación de páginas, sitios y aplicaciones web, han ido surgiendo diversas tecnologías que han evolucionado el modo de desarrollo web. Los frameworks de aplicaciones web se han convertido en herramientas indispensables y populares para un desarrollo web rápido. Una de estos frameworks es Ruby on Rails (RoR) conocido también como Rails, framework de código abierto escrito en el lenguaje de programación orientado a objetos Ruby.

Rails fue lanzado en el año 2004 y ha adquirido popularidad en el mercado y en los desarrolladores a lo largo de los años, permitiéndole madurar como framework y contar con el respaldo de una comunidad estable.

En esta investigación se hará un repaso por la historia de Rails, mencionando aquellas características que lo han hecho destacar por sobre otros frameworks, así como los tipos de proyectos donde Rails es perfecto y en cuáles no, y ejemplos de aplicaciones desarrolladas con este framework para así dar a conocer el impacto que ha tenido este framework en el desarrollo de aplicaciones web. ciones web.

II. Revisión de Ruby on Rails

Es importante aclarar que Ruby on Rails y Ruby no es lo mismo. Por un lado, Ruby es un lenguaje de programación dinámico y de código abierto enfocado en la simplicidad y productividad [1], y en hacer de la programación algo divertido. Fue creado en 1993 por Yukijiro Matsumoto como alternativa para lidiar con la complejidad de otros lenguajes como C++ y Java. A pesar de haber sido lanzado en 1993, se hizo popular hasta el año 2008 [2], después de haber sido utilizado para crear el framework Ruby on Rails.

Ruby on Rails, también conocido como Rails, es un framework web de código abierto desarrollado por David Heinemeier Hansson (conocido como DHH en las comunidades de Ruby) en el 2003 [3]. DHH es cofundador y Chief Technology Officer (CTO) en Basecamp, donde Rails fue desarrollado para uso interno, pero posteriormente, en julio de 2004 fue liberado de forma pública.

Rails fue liberado en la Web 2.0 como un framework para el desarrollo ágil de aplicaciones web [4], y con su lanzamiento, posicionó al lenguaje de programación Ruby como una alternativa de PHP y Java [5] porque intenta combinar la simpleza de PHP con la arquitectura de Java.

A. Principios de Rails

Rails tiene seis objetivos básicos que son: simplicidad, reutilización, capacidad de expansión, capacidad de pruebas, productividad y mantenimiento [4]. Para cumplir estos objetivos, existen seis principios que los soportan, mencionados a continuación.

1) Arquitectura MVC

Rails implementa la arquitectura Modelo-Vista-Controlador (MVC), donde cualquier solicitud web del cliente resulta en una llamada a algún método en el controlador, que a su vez utilizó el modelo para realizar accesos a la base de datos y que eventualmente devuelve la respuesta a la vista [6]. Este patrón organiza el código según su funcionalidad con el fin de evitar que los cambios realizados en un parte del código perjudiquen en otra parte. Además, con ese patrón se asegura que partes del código tengan una sola responsabilidad.

2) Convención sobre configuración

Este principio evita que se los desarrolladores tengan que realizar configuraciones, hablando en el sentido más amplio. El único archivo de configuración es database.yml, el cual contiene la configuración para realizar conexiones con la base de datos [4]. El framework integra los componentes necesarios para su funcionamiento, pero los desarrolladores pueden editar convenciones para realizar funciones especiales y ajustes personalizados. Este principio busca que los desarrolladores usen menos código en sus proyectos al seguir las convenciones.

3) No te repitas

Principio conocido como DRY (por sus siglas en inglés, Don't Repeat Yourselve), busca que los desarrolladores eviten duplicar el código y en su lugar, reutilicen el código.

4) Scaffolding

Rails permite crear plantillas básicas de controladores y vistas con la funcionalidad Crear-Leer-Actualizar-Eliminar (CRUD, por sus siglas en inglés) [4]. Esto permite un ahorro en tiempo y permite al desarrollador trabajar de inmediato en la funcionalidad de la aplicación.

5) Realimentación inmediata

Rails no requiere que los proyectos se compilen primero para ver el resultado, permitiendo a los desarrolladores conocer el estado actual de la aplicación con sólo recargar la página del navegador.

6) Ruby

Como ya se mencionó, Rails está basado en el lenguaje Ruby. Este lenguaje permite a los desarrolladores realizar grandes funcionalidades con pocas líneas de código. Además, Ruby es fácil de leer y está basado en lenguajes como Perl, Python, Smalltalk y Lisp.

B. Componentes de Rails

A continuación, se muestran los componentes del framework Rails.

1) Active Record

Este componente establece la conexión entre los objetos y la base de datos. El Active Record transforma las funciones del CRUD en comandos SQL, envía las solicitudes a la base de datos y devuelve los resultados al controlador. Además, es el responsable de validar los permisos del usuario en cuanto las acciones que puede realizar en la base de datos [7].

El componente sigue el patrón de mapeo relacional de objetos (ORM). Con este mapeo, las clases se relacionan con las tablas de la base de datos, los registros representan los objetos, las columnas los atributos y la herencia como un llave foránea a la tabla correspondiente.

Por lo general, para hacer este mapeo relacional de objetos es necesario realizar una configuración por medio de un XML, pero gracias al principio Configuración sobre Convención, esta configuración se vuelve sencilla.

2) Action Controller

El componente Action Controller recibe las solicitudes del cliente y devuelve una vista. En este componente se define como reaccionará el controlador antes las solicitudes, además se establece la conexión con Active Record y brinda datos de la base de datos a Action View, Web Services y Mailer.

El controlador cuenta con filtros que pueden ser ejecutados antes (before), después (after) o ambos (around) de una acción del controlador [8]. Los filtros before pueden detener el ciclo de ejecución, porque se requiere de una acción para continuar con la ejecución. Los filtros after no pueden detener el ciclo de ejecución como si lo pueden hacer los filtros before, y sólo se ejecutan cuando una acción ya se ha ejecutado de forma exitosa y se puede enviar una respuesta al cliente. Por último, los filtros around contienen código que es ejecutado antes o después de una acción, cediendo la ejecución en el lugar que sea necesario.

3) Action View

Este componente es el responsable de mostrar los datos. Para separar la capa de presentación con el resto de la aplicación, Rails define plantillas en RHTML y RXML. El primero es una mezcla de código Ruby con HTML, donde el código Ruby llena la plantilla con los datos durante la ejecución. El segundo son archivos XML a partir de comandos de Ruby.

Además de las plantillas, existen plantillas parciales o partials y layouts [9]. Los partials extraen partes de código de una plantilla para reutilizarlo en otras plantillas mientras que los layouts renderizan las plantillas según los resultados del controlador, es decir, la forma de presentar la información varía de acuerdo con los resultados.

4) Otros componentes

Seguidamente se mencionan otros componentes de Rails:

- Active Support Core Extensions: responsable de brindar extensiones del lenguaje Ruby.
- Action Mailer Basics: componente para enviar y recibir correos electrónicos en la aplicación.
- Active Job Basics: para crear, poner en cola y ejecutar trabajos en segundo plano.
- Active Storage Overview: para adjuntar archivos a los modelos de Active Record y almacenarlos en la base de datos.
- Action Cable Overview: para incorporar características de tiempo real a través de Web Sockets a la aplicación.

C. Versiones de Rails

En la lista que se muestra a continuación, se observan las versiones más relevantes de este framework [10], así como los aspectos más significativos.

- Rails 1.0 (2005) - mejoras del primer lanzamiento e inclusión de Scriptaculous 1.5 y Prototype 1.4.
- Rails 1.2 (2007) - REST y soporte HTTP.
- Rails 2.0 (2007) - mejoras en recursos de enrutamiento, multiview, autenticación HTTP, sesiones para guardar cookies.
- Rails 2.2 (2008) - compatibilidad con Ruby 1.9 y JRuby.
- Rails 2.3 (2009) - templates, Rack.
- Rails 3.0 (2010) - nuevo motor de consulta, nuevo router para el controlador, protección Cross-site request forgery (CRSF).
- Rails 3.1 (2011) - jQuery, SASS, CoffeeScript, Sprockets con Assets Pipeline.
- Rails 3.2 (2012) - Journey, mejoras en el modo de desarrollo.
- Rails 4.1 (2014) - Spring aplication preloader.
- Rails 4.2 (2014) - consola web.
- Rails 5.0 (2016) - Acton Cable.
- Rails 5.1 (2017) - Soporte a Yarn.
- Rails 5.2 (2018) - Active Storage, almacenamiento en caché de Redis.
- Rails 6.0 (2019) - Action Mailbox, Action Text, pruebas paralelas, pruebas del Action Cable.

D. Gemas

Ruby on Rails cuenta con librerías ya programadas de código abierto conocidas como Gemas que ayudan a los desarrolladores en la programación de la aplicación web [2]. A la fecha de esta investigación, existen 10510 Gemas en total [11] que facilitan y aceleran el desarrollo, además de extender las funcionalidades. Algunas Gemas son:

- Bundler: permite instalar y gestiona todas las gemas que se van a utilizar en la aplicación.
- Callisto: indispensable cuando se trabaje con imágenes porque permite crear thumbnails o miniaturas.
- Devise: solución que permite colocar un sistema autenticación fácil y completo en la aplicación.
- Omniauth: sistema de autenticación para conectarse a redes sociales.
- Faker: permite generar datos falsos que se utilizan durante el desarrollo para realizar las pruebas.
- Kaminari: para aplicar distintas plantillas según los gustos o necesidades del cliente.
- Rspec: para realizar test o pruebas al código de la aplicación.

E. Ventajas de Rails

Algunas de las ventajas de este framework al momento de desarrollar aplicaciones web son:

- Rails es conocido como el framework ágil debido a que está basado en metodologías ágiles, aumentando la productividad de los desarrolladores y disminuyendo el tiempo de desarrollo.
- Permite el desarrollo de aplicaciones complejas y escalables con equipo pequeños gracias a las gemas.
- Como implementa la arquitectura MVC, simplifica la escalabilidad de los proyectos al dividir las partes de la aplicación.
- El lenguaje Ruby del que extiende favorece un desarrollo más dinámico así como permitir la metaprogramación.
- Cuenta con documentación suficiente debido su comunidad consolidada y activa.
- La sintaxis de simple y fácil de leer y escribir.

F. Desventajas de Rails

Algunas de las desventajas que posee este framework se mencionan a continuación.

- Su desempeño es costoso, consumiendo altos recursos del servidor.
- Rails no es escalable a simple vista, porque para escalar una aplicación se debe actualizar el servidor donde se está ejecutando la aplicación o dividir el proyecto en múltiples servidores y herramientas.
- Por su principio de convención sobre configuración, existen características configuradas que no dejan espacio a la creatividad.
- No se cuenta con las librerías necesarias para realizar aplicaciones que utilicen inteligencia artificial o machine learning.

G. Plataformas donde se puede usar

Dependiendo la plataforma en la que se decida instalar Ruby on Rails pueden encontrarse ciertas ventajas y desventajas, la comunidad lo que recomienda es el uso de un SO basado en Linux, pero puede ser instalado sin problemas en otras plataformas, entre las plataformas más comunes:

- Linux
- Mac OS
- Windows

H. Instalación para esas plataformas

1) Mac OS

En Rails el utilizar Mac OS como una plataforma ha sido una opción muy popular entre desarrolladores, para configurar este entorno debemos primero de realizar los siguientes pasos:

a) Command Line Tools for XCode

- Crea una cuenta de Apple ID y accede con la misma.
- Regístrate con dicho ID en otros servicios de Apple como iTunes, iCloud, etc.
- Accede al Apple Developer Portal
- Busca “Command Line Tools (OS X Lion) for Xcode”
- Descarga e instala dicho paquete

b) Instalación con RVM

Con RVM podes instalar distintas versiones de Ruby y mantenerlas a la vez. Para instalar solo debes de ejecutar:

```
$ bash -s stable < <
(curl -L https://get.rvm.io )
$ rvm install 1.9.3 --with-gcc=clang
$ gem install rails
```

2) Windows

Realizar la instalación en Windows es sencillo gracias a RailsInstaller, un instalador desarrollado por Engine Yard que permite tener en un solo medio todas las herramientas necesarias incluido Rails.

a) Descargar e Instalar RailsInstaller

Ingresa al enlace http://railsinstaller.org/ y descarga la versión más actualizada del instalador. Posteriormente instala verificando que estén seleccionadas todas las casillas de variables de entorno como Ruby, DevKit o Git.

b) Comprobar versión de la Gema

Una vez instalado debemos de verificar que la versión de Ruby es la actualizada para ello ejecutamos el siguiente comando en cmd:

```
C:\>gem --version
```

En caso de no encontrarse instalado o estar desactualizado debemos de descargar la última versión en la página https://rubygems.org/pages/download, posteriormente la instalamos.

```
C:\>gem install --local
C:\rubygems-update-2.4.6.gem
C:\>update_rubygems --no-ri --no-rdoc
```

c) Ejecutar Bundle sobre el proyecto

Una vez instalada la versión actualizada solo debemos de situarnos en el proyecto que deseemos trabajar en cmd y ejecutar el siguiente comando:

```
C:\>cd myapp
C:\myapp>bundle install
```

3) Linux

Para instalar sobre Linux existen muchos métodos, uno de los más sencillos es mediante el uso de RVM que instala todo lo necesario para ello inicialmente de abrir la terminal y ejecutar el siguiente comando:

```
$ sudo apt-get install -y git-core
subversion
```

a) Instalación de la firma

Para instalar sin problemas debemos como requisito instalar una key, ya que RVM a partir de su versión 1.26 realiza una comprobación de la misma.

```
$ gpg --keyserver hkp://keys.gnupg.net
--recv-keys
409B6B1796C275462A1703113804BB82D39DC0E3
```

b) Instalación del RVM

Una vez instalada la key solo debemos de ejecutar el siguiente comando en la terminal:

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

Este comando instala las últimas versiones de RVM, Ruby y Rails. Ahora solo debemos de verificar que dichos paquetes se instalaron y en sus versiones más actualizadas.

```
$ type rvm | head -n 1
```

Con este comando podemos verificar que el RVM se encuentra configurado correctamente y que podemos hacer uso de este. Para verificar las distintas versiones instaladas por el RVM utilizamos el siguiente comando:

```
$ rvm list
```

I. Aplicaciones desarrolladas en Ruby on Rails

La tendencia de su uso en el desarrollo de aplicaciones actualmente ha ido disminuyendo, aunque muchas aplicaciones en su momento fueron desarrolladas en Ruby on Rails han ido con el tiempo migrando a otros frameworks debido a distintos factores, entre algunas aplicaciones que originalmente fueron desarrolladas y siguen utilizando este framework se encuentran:

- GitHub
- Shopify
- Heroku
- GitLab
- Airbnb
- Basecamp
- Kickstarter
- Clarity
- ASKfm
- Fiverr

J. Estructura de un proyecto

Al crear un proyecto en Rails contamos con los siguientes directorios y su respectivo contenido:

- app: ficheros de las vistas, modelos y controladores.
- config: archivos de configuración para la conexión con la base de datos. así como el routing de las peticiones de la aplicación.
- db: esquemas y migraciones de la base de datos.
- doc: documentación interna de la aplicación.
- lib: todas la librerías externas necesarias para el funcionamiento de la aplicación.
- log: registros log de la aplicación.
- public: recursos que pueden ser accedidos de forma pública tales como css, imágenes, javascripts, etc.
- tmp: archivos temporales.
- vendor: ficheros de Ruby y Rails así como las gemas.

K. Sintaxis

Ruby es un lenguaje ROO (Radical Object Oriented Language) a diferencia de otros lenguajes con un perfil de objetos como C# y Java que son menos Orientados a Objetos en su sintaxis y semántica.

Entre algunos ejemplos que se pueden mencionar de Ruby y Rails:

1) Comentarios

En Ruby hay dos formas de definir comentarios dentro del código.

La primera es cuando el comentario sólo abarca una línea o parte de una línea. Para este caso se utiliza el carácter de gato (#) y después se agrega el comentario.

```
#--- Esto es un comentario de una línea
```

La segunda es cuando el comentario abarca un bloque de líneas. Para este caso se utiliza el bloque =begin =end

```
=begin
Esto es un comentario de todo un bloque
de líneas
=end
```

2) Clases

La sintaxis para definir en Ruby una clase es similar a lo que pasa en C++, Java y C#:

```
class NombreDeLaClase
end
```

3) Métodos

La sintaxis para definir en Ruby el método de una clase es:

```
def nombreDelMetodo( param1, param2, ...
paramN )
return( valor ) #--- Ojo! usar return es
opcional
end
```

4) Constructores

De igual manera que en C++, Java y C# Ruby tiene un método especial que permite la construcción de un objeto en tiempo de ejecución (constructor). Sin embargo a diferencia de otros lenguajes, en Ruby la sintaxis de constructor NO sigue

las convenciones de nombrado del constructor. Esto es, no se llama igual que el nombre de la clase. La sintaxis es:

```
def initialize( param1, param1, param3 )
end
```

5) Variables de Instancia y Atributos

Una variable de instancia se declara de la siguiente manera:

```
@variableDeInstancia = valor
```

III. Conclusiones

Ruby on Rails sigue siendo hoy en día un framework popular y preferido por muchos para el desarrollo de aplicaciones web. Los principios más fuertes con los que cuenta RoR que son Convención sobre configuración y No te repitas, hacen que sea un framework sencillo de utilizar por los desarrolladores, con el cual se pueden construir aplicaciones en poco tiempo.

En cada año lanzan una nueva versión para mejorar el framework y agregarle nuevas características para hacerlo más completo y con soporte a nuevas tecnologías que van surgiendo. Las Gemas es una de sus características sobresalientes, pues facilitan y agilizan el desarrollo con el código ya programado.

A pesar de su popularidad y sus características, se debe valorar el framework al momento de desarrollar una aplicación porque requiere de inversión en infraestructura para tener buen desempeño y rivalizar con los nuevos frameworks.

References

[1] ”Lenguaje de Programación Ruby”, *Ruby*. [En línea]. Disponible en: https://www.ruby-lang.org/es/.

[2] M. Maldonaro, ”Uso de Ruby on Rails en el desarrollo de software”, *DIGITAL55*, 2018. [En línea]. Disponible en: https://www.digital55.com/desarrollo-tecnologia/ruby-on-rails-desarrollo-software/

[3] D. Heinemeier Hansson, ”David Heinemeier Hansson (DHH)”, *Dhh*. [En línea]. Disponible en: https://dhh.dk/..

[4] M. Bächle and P. Kirchberg, ”Ruby on Rails,” in *IEEE Software*, vol. 24, no. 6, pp. 105-108, Nov.-Dec. 2007.

[5] L. Grimmer, ”Interview with David Heinemeier Hansson from Ruby on Rails”, *Web.archive.org*, 2006. [En línea]. Disponible en: https://web.archive.org/web/20130225091835/http://dev.mysql.com/tech-resources/interviews/david-heinemeier-hansson-rails.html.

[6] J. An, A. Chaudhuri and J. S. Foster, ”Static Typing for Ruby on Rails,” *2009 IEEE/ACM International Conference on Automated Software Engineering*, Auckland, 2009, pp. 590-594.

[7] ”Active Record Basics”, *Ruby on Rails Guides*. [En línea]. Disponible en: https://guides.rubyonrails.org/active-record-basics.html.

[8] ”Action Controller Overview”, *Ruby on Rails Guides*. [En línea]. Disponible en: https://guides.rubyonrails.org/action-controller-overview.html.

[9] ”Action View Overview”, *Ruby on Rails Guides*. [En línea]. Disponible en: https://guides.rubyonrails.org/action-view-overview.html.

[10] ”Releases”,*Riding Rails*, 2019. [En línea]. Disponible en: https://weblog.rubyonrails.org/releases/.

[11] ”Gems”, *Rubyg Gems*, 2020. [En línea].Disponible en: https://rubygems.org/gems.