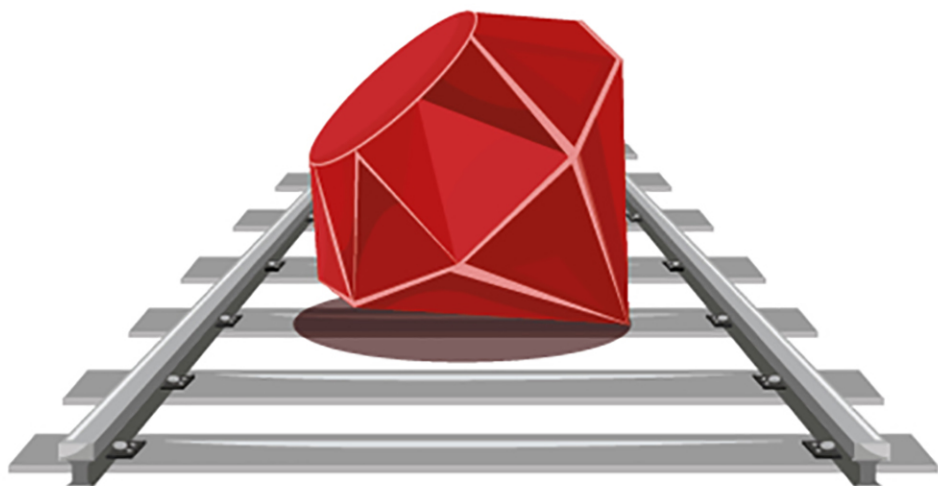


# RUBY ON RAILS

Aprende a crear aplicaciones web  
desde cero



Javier Vázquez y Daniel Lorenzo



# **RUBY ON RAILS**

**Aprende a crear aplicaciones  
web desde cero**

# **RUBY ON RAILS**

**Aprende a crear aplicaciones  
web desde cero**

**Javier A. Vázquez Olivares  
Daniel Lorenzo Martínez**



ALPHA EDITORIAL

ALFAOMEGA COLOMBIANA S.A.

Calle 62 No.20-46 esquina, Bogotá  
Teléfono (57-1) 746 0102 Fax: (57-1) 210 0122  
cliente@alfaomegacolombiana.com  
www.alfaomega.com.co  
www.alpha-editorial.com

RC LIBROS

Calle Mar Mediterráneo, 2. N-6  
28830 San Fernando de Henares, Madrid  
Teléfono +34 91 677 57 22  
info@rclibros.es  
www.rclibros.es

Primera edición: Madrid, 2020  
Bogotá, 2020

© Javier A. Vázquez Olivares y Daniel Lorenzo Martínez  
© Alpha Editorial  
© Alfaomega Colombiana S.A.  
© RC Libros, 2020

Todos los derechos son reservados. Esta publicación no puede ser reproducida total ni parcialmente. No puede ser registrada por un sistema de recuperación de información, en ninguna forma ni por ningún medio, sea mecánico, fotoquímico, electrónico, magnético, electroóptico, fotocopia o cualquier otro, sin el permiso previo y por escrito de la editorial.

Diseño de colección y pre-impresión: Grupo RC  
Diseño de cubierta: Cuadratín

ISBN: 978-958-778-636-1 (Colombia)  
ISBN: 978-84-948972-9-0 (España)

Hecho en Colombia  
*Printed and made in Colombia*

# ÍNDICE

---

<b>INTRODUCCIÓN .....</b>	<b>XV</b>
<b>PARTE I RUBY</b>	
<b>MI PRIMER ENCUENTRO .....</b>	<b>1</b>
<b>CAPÍTULO 1. ACERCA DE RUBY .....</b>	<b>3</b>
Sobre el lenguaje y su creador.....	3
Características y comparación.....	4
<b>CAPÍTULO 2. PREPARACIÓN DEL ENTORNO DE TRABAJO.....</b>	<b>7</b>
Instalación de Ruby y Ruby on Rails .....	7
Instalación en Windows .....	8
Instalación en GNU/Linux .....	11
Instalación en OS X.....	17
Entornos de desarrollo .....	20
Editores de código.....	20
IDEs.....	21
<b>CAPÍTULO 3. MIS PRIMEROS PROGRAMAS EN RUBY .....</b>	<b>23</b>
EL IRB de Ruby .....	23
Ejecución de un script desde la terminal.....	25
Antes de continuar considera lo siguiente .....	27
<b>CAPÍTULO 4. INTRODUCCIÓN A LA POO .....</b>	<b>31</b>
La POO en palabras simples.....	31
<b>CAPÍTULO 5. UTILIZANDO OBJETOS EN RUBY .....</b>	<b>35</b>
Los objetos de Ruby.....	35
Creación de un objeto en Ruby.....	36
Trabajando con los métodos de Ruby.....	37
Valores de retorno y paso de argumentos en los métodos .....	38

Paso de argumentos a los métodos .....	39
La biblioteca de clases de Ruby .....	42
<b>VARIABLES Y OBJETOS .....</b>	<b>47</b>
<b>CAPÍTULO 6. LAS VARIABLES .....</b>	<b>49</b>
Asignación de objetos a las variables .....	49
Un nombre correcto para mi variable .....	50
¿Qué es el tipado dinámico? .....	51
Asignaciones abreviadas .....	52
Asignación del valor de una variable a otra .....	53
<b>CAPÍTULO 7. LOS NÚMEROS .....</b>	<b>57</b>
¿Qué son los números y de dónde provienen? .....	57
Números enteros .....	58
Sistemas numéricos .....	59
Algunos métodos de la clase Integer .....	60
Números flotantes .....	62
Truncamiento y redondeo .....	62
Algunos métodos heredados de Numeric .....	64
Operaciones con números .....	64
Operadores aritméticos y algunos ejemplos sencillos .....	64
Evaluación de expresiones con varios operadores involucrados .....	67
comparaciones de números .....	69
Operadores relacionales o de comparación .....	69
Aplicación de los operadores relacionales .....	70
Números aleatorios y rangos .....	70
<b>CAPÍTULO 8. CADENAS DE CARACTERES .....</b>	<b>75</b>
Generalidades de las cadenas .....	75
Algunos métodos para empezar .....	76
Transformación de los caracteres alfabéticos .....	78
Comparación de cadenas .....	79
Concatenación .....	80
Búsqueda de caracteres .....	82
Obtención de caracteres y subcadenas .....	82
Inserción en una cadena .....	83
Reemplazo de caracteres .....	84
Eliminación de fragmentos de una cadena .....	85
Las particularidades de cada comilla .....	87
Comillas dobles .....	87
Las secuencias de escape .....	87
La interpolación .....	89
Comillas simples .....	90
Los únicos caracteres especiales .....	91
Percent strings (%Q y %q) .....	91
here documents o heredocs .....	92

<b>CAPÍTULO 9. FECHAS Y HORAS .....</b>	<b>95</b>
La clase Time.....	95
La clase Date .....	98
DateTime, una subclase de Date .....	99
Formateo de fechas y horas .....	100
Comparación de fechas y horas.....	102
<b>CAPÍTULO 10. COLECCIONES DE OBJETOS.....</b>	<b>105</b>
Los arreglos.....	105
Creación de un arreglo .....	105
Una miscelánea de métodos básicos .....	107
¿Cómo extraer elementos de un arreglo? .....	110
Diversas maneras de agregar más elementos a un arreglo .....	112
Reducción de un arreglo .....	114
Operaciones de arreglos .....	116
Operación resta.....	118
Operación unión.....	119
Operación intersección .....	120
Concatenación de arreglos.....	120
Los Hashes .....	121
Creación de un hash .....	121
¿Qué son los símbolos? .....	122
Algunos métodos similares a los de las cadenas.....	123
Pregúntale al hash acerca de.....	124
Obtención de claves y valores .....	126
Obtención de una sola clave o valor .....	127
Obtención de varias claves o valores .....	127
Modificación de los hashes .....	128
Inserciones de clave-valor y sustitución de valores .....	128
Combinación de hashes .....	129
Eliminación a través de las claves .....	130
<b>CAPÍTULO 11. CONVERSIONES DE DATOS.....</b>	<b>131</b>
Conversiones frecuentes .....	131
Conversiones de números sin dejar de ser números .....	131
De texto a números.....	132
Entre colecciones de objetos .....	133
Cualquier otro objeto a texto.....	134
<b>ESTRUCTURAS DE CONTROL .....</b>	<b>135</b>
<b>CAPÍTULO 12. CONDICIONES .....</b>	<b>137</b>
Expresiones.....	137
Expresión if.....	137
Expresión if - else .....	138
expresión elsif .....	141
Expresión case .....	143
case como un if-elsif.....	144

Case con un argumento .....	145
Expresión unless .....	148
if y unless como modificadores .....	149
Condicionales y operadores lógicos .....	150
Operadores lógicos .....	150
Operador ! .....	151
Operador && .....	152
Operador    .....	153
Más de un operador lógico en una expresión .....	154
Ahora todo junto .....	155
Un último ejemplo y tarea para la casa .....	156
<b>CAPÍTULO 13. CICLOS E ITERADORES .....</b>	<b>159</b>
Ciclos .....	159
Ciclo for .....	159
Ciclo while .....	162
Ciclo until .....	163
Usando while y until como modificadores.....	164
Hacer al menos una repetición con while o until se cumpla o no la condición	164
Declaraciones break, next y redo.....	168
Iteradores.....	170
Sobre los iteradores en general .....	170
Iteradores times, upto y downto .....	171
Las cadenas también se iteran con estilo .....	173
Iteración de un arreglo .....	173
Iteración de un hash .....	178
<b>LA POO A DETALLE.....</b>	<b>181</b>
<b>CAPÍTULO 14. LOS MÉTODOS .....</b>	<b>183</b>
Sobre los métodos en general.....	183
Métodos con argumentos .....	185
Argumentos posicionales con valores por defecto.....	187
Argumentos de palabras clave (no posicionales).....	188
Argumentos Array/Hash .....	190
Los métodos y los bloques de código.....	193
Algunos aspectos de los bloques .....	193
De bloques a objetos Proc y viceversa.....	196
Recursividad .....	199
<b>CAPÍTULO 15. CLASES Y OBJETOS .....</b>	<b>203</b>
Diseño de clases .....	203
Acciones (métodos) que realizarán los objetos .....	204
Las características (atributos) que tendrán los objetos .....	205
Inicialización de atributos .....	207
Todo junto.....	210
Herencia de clases.....	212
Sobrescritura de métodos .....	213

De lo general a lo específico.....	214
La herencia con las clases de Ruby.....	217
La visibilidad de los métodos.....	217
La clase Singleton, los métodos de clase y las variables de clase.....	222
<b>CAPÍTULO 16. LOS MÓDULOS.....</b>	<b>227</b>
¿Qué es un módulo, para qué sirve y cómo se define?.....	227
Los módulos como namespaces.....	228
Los mixins.....	231
Módulos en diferentes archivos.....	233
Explicación del scope de las constantes en módulos / clases.....	234
<b>CUENTAME MÁS SOBRE RUBY.....</b>	<b>237</b>
<b>CAPÍTULO 17. LAS EXPRESIONES REGULARES.....</b>	<b>239</b>
Perdiéndole el miedo a las expresiones regulares.....	239
Construcción de Regex para números telefónicos con 10 dígitos (paso a paso).....	242
¿Dónde más se usan los regex?.....	245
Tablas de ayuda.....	245
<b>CAPÍTULO 18. ARCHIVOS Y CARPETAS.....</b>	<b>247</b>
Archivos.....	247
Permisos en archivos de texto plano.....	247
Permiso de solo lectura.....	248
Permiso de solo escritura.....	250
Permiso de lectura y escritura.....	251
Directorios.....	252
Creación de directorios.....	252
Obtención de la ruta, nombre y extensión de un archivo.....	255
Verificación y búsqueda.....	256
<b>CAPÍTULO 19. EXCEPCIONES.....</b>	<b>259</b>
¿Qué es una excepción en Ruby?.....	259
Otros tipos de excepciones que ocurren frecuentemente.....	261
Manejo de excepciones.....	263
¿Cómo evitar que un programa termine antes?.....	263
El método raise: Lanzar excepciones intencionadamente.....	264
Dos elementos más en el bloque: else y ensure.....	266
<b>PARTE II RUBY ON RAILS</b>	
<b>FUNDAMENTOS.....</b>	<b>267</b>
<b>CAPÍTULO 20. INTRODUCCIÓN A RUBY ON RAILS.....</b>	<b>269</b>
¿Qué es Ruby on Rails?.....	269
La filosofía Rails.....	270
<b>CAPÍTULO 21. FUNDAMENTOS DEL DESARROLLO WEB.....</b>	<b>269</b>
¿Cómo funciona una aplicación web?.....	273
¿Qué es un servidor web?.....	275
El protocolo de comunicación HTTP.....	276
Peticiones y respuestas HTTP.....	277

Métodos de peticiones HTTP .....	279
Código del lado del cliente vs código del lado del servidor .....	281
<b>CAPÍTULO 22. PRIMER PROYECTO EN RUBY ON RAILS .....</b>	<b>285</b>
Creación de un proyecto en Ruby on Rails.....	285
Ejecución del servidor local .....	291
Arquitectura MVC en Ruby on Rails .....	292
Solución al bug con usuarios de Windows .....	294
<b>FUNDAMENTOS DEL TRABAJO EN ROR .....</b>	<b>297</b>
<b>CAPÍTULO 23. PRIMEROS PASOS EN RUBY ON RAILS .....</b>	<b>299</b>
Creación de un controlador .....	299
¿Qué acabo de crear? .....	301
Proceso de una petición en Ruby on Rails .....	303
Comunicación entre la vista y el controlador .....	310
Expresiones y scriptlets en las vistas .....	311
Incluamos un modelo.....	314
SQLite .....	315
Configuración de la base de datos .....	318
Creación de un modelo en Ruby on Rails .....	320
Ejecutando la migración .....	323
Uso de la consola de Rails.....	327
Juntemos todo .....	335
Estructura de un proyecto Ruby on Rails .....	338
El archivo Gemfile .....	341
<b>MODELOS EN PROFUNDIDAD .....</b>	<b>345</b>
<b>CAPÍTULO 24. CREACIÓN DE MODELOS EN RUBY ON RAILS .....</b>	<b>347</b>
Introducción .....	347
Generación de modelos .....	348
Usos avanzados del generador de modelos .....	350
Preparación de un nuevo proyecto en Rails.....	352
<b>CAPÍTULO 25. ACCIONES CON MODELOS.....</b>	<b>355</b>
Acciones CRUD: Crear .....	355
El archivo seeds.rb .....	357
Acciones CRUD: Actualizar .....	358
Acciones CRUD: Eliminar .....	359
Acciones CRUD: Leer .....	360
Métodos para obtener una instancia de modelo .....	360
Métodos de búsqueda dinámicos.....	363
Métodos para obtener colecciones de instancias de modelo y otros métodos más .....	364
Utilizando not y or .....	368
Ordenamiento de datos.....	368
Selección específica de campos .....	369
Buscar o construir un nuevo objeto.....	371
Realizar búsquedas mediante SQL nativo.....	372

Obtener datos específicos de una consulta .....	373
Los cálculos en Active Record .....	374
Contar registros.....	375
Agregando columna edad a los usuarios .....	377
Estableciendo valores aleatorios al campo edad .....	380
Obtener el mínimo, máximo, la suma y el promedio.....	381
<b>CAPÍTULO 26. ASOCIACIÓN DE MODELOS .....</b>	<b>383</b>
Asociaciones en Active Record .....	383
Creando los modelos Book y Borrowing .....	383
Tipos de relaciones en los modelos.....	388
Asociación belongs_to .....	391
Asociación has_one.....	392
Asociación has_many.....	393
Asociación has_many :through .....	393
Asociación has_one :through.....	395
Asociación has_and_belongs_to_many.....	395
Métodos añadidos a los modelos para manipulación de asociaciones .....	398
Métodos automáticamente añadidos en asociaciones has_many, has_many :through y has_and_belongs_to_many.....	399
Métodos añadidos en asociaciones has_one y belongs_to .....	401
Operaciones con relaciones .....	403
Crear asociaciones entre instancias de modelos .....	403
Operaciones de selección en asociaciones .....	414
Crear métodos para manipular asociaciones has_many .....	418
<b>CAPÍTULO 27. JOINS.....</b>	<b>427</b>
Uniones de tablas .....	427
Uniones mediante el método joins .....	429
Uniones mediante el método left_outer_join .....	433
El uso de includes .....	435
<b>VISTAS Y CONTROLADORES.....</b>	<b>439</b>
<b>CAPÍTULO 28. LOS CONTROLADORES DE RAILS.....</b>	<b>441</b>
Generalidades sobre los controladores .....	441
Creación de controladores .....	442
Convenciones en los controladores.....	448
Conceptos importantes sobres los controladores.....	449
Las vistas y ActionView .....	450
Comunicación entre controladores y vistas .....	450
Manejo de parámetros en los controladores .....	454
Recepción de parámetros en el controlador.....	454
Parámetros como array.....	459
Rutas, controladores y parámetros.....	460
La clase ApplicationController .....	463
<b>CAPÍTULO 29. VISTAS Y RENDERIZACIÓN EN RAILS.....</b>	<b>465</b>
Introducción .....	465

Continuación del proyecto Biblioteca .....	465
Visualización de los datos de los modelos en las vistas .....	466
Modificación de los datos de los modelos desde las vistas .....	477
Uso de los formularios HTML.....	477
Creación de formularios para creación y edición de datos mediante helpers .	478
Eliminación de registros desde una vista.....	492
Los helpers y las vistas .....	495
Helpers de formularios .....	495
Helpers para selects.....	502
Renderización: Métodos y formatos .....	506
Renderizado en formato JSON y XML .....	508
El método render vs redirect_to.....	508
Layouts .....	509
Partials.....	516
Discriminación de segmentos de vista .....	519
<b>CAPÍTULO 30. ENRUTAMIENTO AVANZADO EN RAILS .....</b>	<b>521</b>
Introducción .....	521
El archivo routes.rb .....	522
Enrutamiento automático.....	522
El método resources .....	523
URL Helpers.....	524
Uso de los URL y helpers generados por resources .....	525
Cambios en los formularios .....	529
Namespaces y rutas .....	532
Uso del método scope .....	534
Recursos anidados .....	536
Añadir más de una url RESTful .....	544
<b>CAPÍTULO 31. UTILIZACIÓN DE LOS ASSETS .....</b>	<b>547</b>
Introducción .....	547
Asset pipeline .....	547
¿Cómo utilizar los assets? .....	549
Utilizar archivos javascript .....	549
Utilizar hojas de estilo.....	551
<b>RAILS AVANZADO .....</b>	<b>553</b>
<b>CAPÍTULO 32. SCAFFOLDING EN RUBY ON RAILS .....</b>	<b>555</b>
Introducción .....	555
Crear un scaffold en una aplicación .....	555
El comando scaffold .....	556
Exploración del código generado.....	559
<b>CAPÍTULO 33. TÓPICOS AVANZADOS EN MODELOS .....</b>	<b>567</b>
Introducción .....	567
Callbacks.....	568
Explicación del funcionamiento de un callback .....	568
Múltiples ejecuciones para un callback .....	569

Callbacks disponibles.....	570
Disparadores de callbacks .....	571
Callbacks relacionales.....	571
Conclusión sobre los callbacks .....	572
Validaciones.....	573
Ejemplo práctico de validaciones .....	573
Validación del modelo author .....	573
Validación del modelo Book.....	575
Validación del modelo User .....	578
Ejercicios de validación .....	579
Mostrar los errores de validación en las vistas .....	580
Migraciones .....	584
Creación de migraciones .....	585
Añadir campos a una tabla existente .....	586
Remover campos de una tabla existente .....	587
Creación de tablas.....	587
Creación de llaves foráneas .....	588
Métodos para usar en el método change de una migración .....	588
Rollback de migraciones.....	589
Configuración de la base de datos.....	590
Configuración con postgresql.....	591
Configuración con mysql .....	594
Conclusión .....	595
<b>CAPÍTULO 34. TÓPICOS AVANZADOS EN CONTROLADORES .....</b>	<b>597</b>
Introducción .....	597
Strong parameters.....	597
Los filtros .....	600
uso de sesiones en Rails .....	602
Guardar datos en la sesión.....	603
Ejemplo de uso de la sesión .....	605
El hash flash.....	607
Las cookies .....	610
<b>CAPÍTULO 35. API REST .....</b>	<b>613</b>
Introducción .....	613
Instalación de un cliente http.....	614
Los datos JSON.....	617
Creación de una aplicación Rest API en Rails .....	618
Creación de modelos.....	619
Creación de controladores .....	620
Creación del método json_response.....	621
Creación de datos de prueba .....	622
Creando acciones en los controladores .....	624
Probar la acción index de la rest api de checklist.....	625
Implementación del resto de las acciones CRUD de una API Rest.....	626

Probar el resto de las acciones CRUD en Checklist e Item.....	630
Conclusión.....	635
<b>CAPÍTULO 36. CORREO ELECTRÓNICO .....</b>	<b>637</b>
Introducción .....	637
Creación y configuración de la aplicación .....	637
Generación de Mailer.....	639
Creación de una acción para envío de email .....	639
Creación de una vista para el email .....	640
Invocación de Mailer desde un controlador .....	641
Prueba del Mailer .....	642
Un posible fallo en Gmail.....	643
<b>CAPÍTULO 37. TESTING .....</b>	<b>649</b>
Introducción .....	649
Desarrollo orientado a Testing.....	650
Instalación de RSpec y Capybara.....	650
Tipos de test.....	652
Nuestro primer Test .....	652
Probando otro escenario .....	663
Testing en modelos .....	666
Conclusión .....	670
Gemas de utilidad para el testing en Rails.....	670
shoulda-matchers .....	671
faker.....	672
Sigüientes pasos en testing.....	672
<b>CAPÍTULO 38. PROYECTO FINAL EN RUBY ON RAILS .....</b>	<b>673</b>
Introducción .....	673
<b>ÍNDICE ANALÍTICO .....</b>	<b>675</b>

# INTRODUCCIÓN

---

Sin duda alguna, Ruby y su framework Rails han despertado el interés de muchísimas personas alrededor del mundo. La razón que con más probabilidad ha causado esto es la necesidad de herramientas de desarrollo ágil que permitan al programador construir y depurar productos de software rápida y ordenadamente. En este aspecto, pocos frameworks de desarrollo web se comparan con Ruby on Rails, sin embargo, Rails no sería lo que es si no fuese por el lenguaje de programación que lo soporta. Cuando combinas la versatilidad y rapidez de Rails con la elegancia y legibilidad del lenguaje Ruby tienes como resultado programadores felices y productivos.

Cuando llegues a dominar Ruby on Rails, es posible que puedas realizar en la mitad del tiempo, lo que otro programador haría en otra tecnología. Ruby on Rails es simplemente genial.

En este libro aprenderás paso a paso cómo crear software con Ruby on Rails desde cero. Comenzaremos con las bases del lenguaje Ruby y paulatinamente irás aprendiendo temas más avanzados. Te darás cuenta de que Ruby es intuitivo e idiomático, esto quiere decir que programar en Ruby se parece mucho al lenguaje natural (en inglés) y la estructura de código es tan sencilla que por momentos parece que escribimos en pseudocódigo.

Con explicaciones sencillas y varios ejemplos que podrás practicar por ti mismo, aprenderás sobre variables, objetos, arreglos, hashes y otros temas de importancia en Ruby.

El libro se compone de dos secciones. En la primera sección verás los temas relativos al lenguaje Ruby. En la segunda sección, aprenderás sobre su framework Rails. En esta segunda sección, aprenderás las bases del desarrollo web, servidores web, protocolo HTTP, la filosofía y formas de trabajo en Ruby on Rails, sus ventajas respecto a otras plataformas y de inmediato comenzaremos a desarrollar nuestras primeras aplicaciones paso a paso e incrementando el nivel paulatinamente conforme avanzamos. Y como comprador tienes acceso a los anexos complementarios de este libro que pueden ser descargados gratuitamente, visitando: <http://rclibros.es>, y después su página individual.

Al terminar este libro/curso, serás capaz de realizar potentes aplicaciones web, construir servicios API Rest y montar tus servicios o aplicaciones en un servidor en la nube.

## LOS AUTORES

---

Javier Arturo Vázquez está comprometido con el mundo del desarrollo de sistemas desde hace 8 años, ha participado como ingeniero de software para las fuerzas armadas de México y algunas empresas de ámbito privado, desarrollando simuladores, sistemas cliente-servidor y sistemas web, utilizando diversas tecnologías y lenguajes de programación. Es un apasionado de la enseñanza y cuenta con varios cursos en la plataforma udemy.com, además de un canal en youtube (profe Javier) donde enseña programación en Java y Ruby principalmente. Actualmente colabora en una empresa emergente como director de desarrollo utilizando Ruby on Rails y varias otras tecnologías para la construcción de una plataforma web y móvil de impacto mundial.

Daniel Lorenzo Martínez es ingeniero y licenciado en sistemas por el Instituto Tecnológico de Misantla de Veracruz, México; a lo largo de su carrera, ha trabajado en el desarrollo de productos de software con varios lenguajes y tecnologías, especialmente Ruby y Rails; su actividad está relacionada con el desarrollo de sistemas de facturación electrónica y web con fines educativos, actualmente trabaja como desarrollador de proyectos para una importante empresa americana utilizando técnicas de repetición espaciada para la memorización.

# Parte I

# RUBY

## MI PRIMER ENCUENTRO

Esta parte se compone de cinco capítulos. En los primeros tres conocerás las características de este maravilloso lenguaje, lo instalarás y harás tus primeros programas. Luego habrá un capítulo en donde explicaremos de manera conceptual y con ejemplos del mundo real en qué consiste la programación orientada a objetos (POO), ya que Ruby se caracteriza por ser 100% orientado a objetos. Y el último capítulo será prácticamente un paseo por la API de Ruby.

# 1

## ACERCA DE RUBY

Permíteme hablarte un poco sobre el lenguaje y desmitificar también algunos aspectos en este capítulo. Trataré de ser objetivo para darte una descripción lo más certera posible para que con ello estés plenamente consciente de lo que estás próximo a aprender.

### **SOBRE EL LENGUAJE Y SU CREADOR**

---

A este precioso y valioso lenguaje (como la piedra que lo representa) lo que lo hace precioso y de gran estima es su naturalidad. Yukihiro Matsumoto ha logrado crear un lenguaje con una envidiable facilidad de lectura, tanto que desde que hagas tu primer “hola mundo” verás su gran naturalidad.

Para entender la razón que llevó a Matz a hacer su propio lenguaje, debes de tener en cuenta que él era experimentado en los lenguajes: Perl, Smalltalk, Eiffel, Ada y Lisp. Cuando Matz usaba uno en particular no se sentía satisfecho, como seguramente nosotros tampoco nos hemos sentido complacidos en su totalidad al usar alguno. Él admiraba la belleza y el potencial de cada uno, pero siempre había algo que lo desilusionaba y, cuando usaba otro, ocurría lo mismo. La gran mayoría de las veces, su insatisfacción fue que los lenguajes no eran muy expresivos, su lectura era posible únicamente por gente experimentada en el lenguaje. Este fue el mayor descontento y el motivo suficiente que lo llevó a crear el lenguaje Ruby. Matz aprovechó lo mejor de cada uno. Mientras creaba Ruby, tú y yo estuvimos en su mente, en el corazón de él estaba darnos a conocer un lenguaje, ¡sí!, pero más que un lenguaje, en realidad quería presentarnos a un amigo. Matz quiere que Ruby sea tu amigo y yo me encargaré de hablarte sobre él.




Ruby ya tiene sus ayerres pues nació en la década de los noventa, para ser exactos, en 1995 (fecha en la que se publicó) pero su creación llevó dos años. Una vez que lo concretó y lo publicó, enseguida se hizo popular.

Desde los primeros años, Ruby fue todo un éxito, fue muy bien recibido y además lanzado en el momento oportuno (en los inicios de la Web), justo por eso es que existen numerosos *frameworks* y *microframeworks* para este lenguaje. Por supuesto, el más popular de ellos es Ruby on Rails (el *framework* de aplicaciones web preferido por los *rubyist*). Ruby On Rails salió diez años después que Ruby, pero marcó la historia al llevar el florecimiento de Ruby a gran escala. De esta forma, Ruby fue abriéndose brecha entre los lenguajes más posicionados y atrayendo a más adeptos con sus *frameworks* como si fuesen invitaciones coloridas.

## CARACTERÍSTICAS Y COMPARACIÓN

Aunque hay muchas tecnologías y lenguajes que nos permiten el desarrollo web, comparemos Ruby con dos de los lenguajes más populares para el desarrollo web: Java y PHP. De antemano aclaro que no tengo la intención de hablar mal de un lenguaje solo para engrandecer a otro, por favor ten esto muy presente.

### LENGUAJES DE PROGRAMACIÓN

CARACTERÍSTICA	 RUBY	 JAVA	 PHP
Interpretado	😊	😞	😊
Multiparadigma	😊	😊	😊
Libre y de código abierto	😊	😞	😊
Rápido (velocidad de procesamiento)	😞	😊	😞
Legible	😊	😞	😞
Popular	😊	😊	😊
Multiplataforma	😊	😊	😊
Fácil de aprender	😊	😞	😊
Con buena documentación	😊	😊	😊
Con una comunidad activa y creciente	😊	😊	😊
Propósito general	😊	😊	😊

Hay algunas características que ya las mencioné anteriormente de forma indirecta, pero hay otras que necesitan ser definidas. Por ejemplo:

**Interpretado:** Cuando escribimos código lo hacemos en un lenguaje muy similar al nuestro –en este caso usando Ruby– pero la computadora no entiende ni una *jota* de lo que queremos que haga. Para darle a conocer a nuestra computadora las instrucciones en su lenguaje, necesitamos de un procesador de lenguaje que actúe como un intermediario entre nosotros y nuestra computadora. Los procesadores de lenguaje son programas que convierten el código fuente en lenguaje máquina (ceros y unos). Los procesadores de lenguajes pueden ser compiladores o intérpretes.

En un lenguaje interpretado el código fuente será convertido al lenguaje máquina en el momento de ejecución por el intérprete, mientras que en un lenguaje compilado requerirá convertir el código fuente en su totalidad a lenguaje máquina, antes de ser ejecutado.

Ruby es un lenguaje interpretado y “el intermediario entre nosotros y Ruby” se llama *Interactive Ruby Shell (irb)*.

**Rápido (velocidad de procesamiento):** El que Ruby use un intérprete no son buenas noticias en cuanto al tiempo de ejecución, ni para él ni para algún otro lenguaje que funcione de esta manera.

En conclusión, con un lenguaje interpretado estamos sacrificando velocidad por comodidad, pero descuida: Ruby tampoco es lento como un caracol, si así fuera nadie lo querría usar.

**Multiparadigma:** Ruby es multiparadigma. Los *paradigmas de programación* son propuestas aceptadas como modelos que pretenden resolver los problemas en el desarrollo de software bajo la influencia de ciertas filosofías. Ruby acepta la programación imperativa y la funcional, pero, aunque es posible programar de forma funcional, lo suyo son los objetos. La *programación orientada a objetos* (POO) es sin duda una de las características principales de Ruby y consiste en imitar a los objetos del mundo real y hacer de ellos objetos de programación, objetos que al igual que los del mundo real tienen características y comportamientos.

No te preocupes ahora por entender este paradigma. Un poco más adelante te explicaré en qué consiste, utilizando analogías del mundo real.

**Libre y de código abierto:** Ruby puede ser usado por cualquiera sin ningún costo. Y del código fuente del lenguaje puedes hacer lo que te plazca, si quieres puedes adaptarlo a tus necesidades, resolver bugs, realizar alguna mejora, distribuirlo, etc. Tienes toda la libertad concedida.

**Multiplataforma:** Como consecuencia de ser un lenguaje interpretado, tus programas en Ruby podrán ser ejecutados en cualquier sistema operativo (en

Windows, en alguna distribución GNU/Linux o Mac Os) siempre y cuando tengas instalado el intérprete de Ruby.

**Fácil de aprender:** No es que está característica necesite de una definición clara y detallada, sino que veo una oportunidad para hacer una aclaración sobre los comentarios precipitados y endeble que se han suscitado por la “simplicidad” del lenguaje y que han influido en muchas personas a tal grado de rechazarlo por la falsa imagen que se ha creado en algunos círculos de programadores. Ruby no es un lenguaje para novatos o peor aún como lo expresan muchos: “*un lenguaje para niñas*” (lo que evidencia el poco criterio de los que lo comentan pues las niñas también pueden ser excelentes programadoras), no, Ruby es un lenguaje muy maduro enfocado en la productividad. Es cierto que se recomienda aprender a programar con lenguajes como PHP, Python o Ruby para luego pasar a lenguajes más “serios” como C++ o Java pues con estos lenguajes de programación, prácticamente el mismo día que lo instalas puedes estar escribiendo código muy fácilmente. Esto nos habla de lo mucho que podemos lograr en muy poco tiempo, no de una simpleza.

**Con buena documentación y una comunidad activa:** Uno de los temores al iniciarse en algún lenguaje, es avanzar y llegar a un punto sin salida. En más de una ocasión nos ha pasado esto ¿verdad? ¡No hay nada que temer! La API de Ruby es una de las mejores documentadas y cuenta con una gran cantidad de ejemplos.

Puedes echar un vistazo a la documentación oficial de Ruby en la siguiente dirección: <https://ruby-doc.org/>

Generalmente la documentación oficial resolverá un montón de dudas, pero, si aún hubiese algo que no entendemos, en *ruby-talk* (es la lista de correos más popular donde se tratan temas en general de Ruby) siempre habrá alguien dispuesto a responder a tus preguntas. Para poder suscribirte a la lista de correo *ruby-talk* escribe la siguiente url en tu navegador.

<https://www.ruby-lang.org/es/community/mailling-lists/>

**De propósito general:** Aunque su framework Rails se ha encargado de que sea mayormente utilizado para aplicaciones web, Ruby puede tener diferentes usos y aplicaciones. Si quieres saber al respecto, te dejo un enlace que te llevará a una lista de historias de éxito. Algunas de estas historias tienen que ver con simuladores, modelado 3D, robótica, redes, entre otras más. Solo por ponerte un ejemplo, el centro de Investigación Langley de la NASA y Motorola usaron Ruby para crear simuladores. Además, el recientemente popular google Sketchup utiliza Ruby como motor para el diseño y modelado 3D.

<https://www.ruby-lang.org/en/documentation/success-stories/>

# 2

## PREPARACIÓN DEL ENTORNO DE TRABAJO

Las instalaciones en general son una de las cosas que muchos de nosotros odiamos, pero sea que queramos o no, las debemos hacer 😞. Sin embargo, si sigues los pasos que describiré a continuación según el sistema operativo de tu preferencia, en máximo 20 minutos tu computadora estará ansiosa por trabajar 😊.

Posterior a la instalación de Ruby y Rails, tendrás tiempo para ir probando con libertad las herramientas de edición de código que te mencionaré u otra que desees conocer. Al final usa la que más te agrade o en la que mejor te desempeñes.

### INSTALACIÓN DE RUBY Y RUBY ON RAILS

---

Los ejemplos que se incluyen en el presente libro en su totalidad fueron escritos usando **Ruby 2.4.4** (la versión estable y con mayor cantidad de gemas compatibles poco antes de la publicación de este libro) y **Rails 5.2.2**. Nosotros estamos usando estas dos versiones, pero no debe haber cambios significativos en una versión mayor. Aun así, mi deseo es que me sigas a la par sin ningún problema hasta el final del libro y para eso, te mostraré cómo lo hice yo.

En la siguiente dirección podrás elegir cualquier versión de Rails:

<https://rubygems.org/gems/rails/versions>

Una cosa es cierta, no todos contamos con los mismos equipos, ni sistemas operativos (de ahora en adelante SO) en nuestras PCs, por esa razón mostraré la instalación del lenguaje, de su framework y de todo lo necesario, tratando de cubrir

la mayor cantidad de SOs. No es necesario que leas todo, ve directamente a la explicación de la instalación del SO que uses, ya sea Windows, alguna distribución de GNU/Linux basada en Ubuntu u OS X.

## Instalación en Windows

---

A pesar de que existe un empaquetado para Windows que nos haría la vida de maravilla llamado *RailsInstaller*, en este apartado te enseñaré a instalar Ruby y Rails en las versiones propuestas (2.4.4 y 5.2.2, respectivamente). La ventaja de hacerlo de esta manera sin duda alguna es que podrás instalar la versión que consideres más conveniente.

A pesar de lo dicho, si quieres hacerlo por medio de *RailsInstaller*, el ejecutable lo encontrarás en la página oficial, *RailsInstaller*.

<http://railsinstaller.org/en>

La instalación es sumamente sencilla, prácticamente es siguiente, siguiente, siguiente y ¡ya! (aunque por lo general todas las instalaciones así son).

Para hacerlo de la segunda forma primeramente debes dirigirte a la página oficial de *RubyInstaller* y dar clic en “Download” (un botón que resalta a primera vista). Te redireccionará a una nueva página donde se encuentran los instaladores de Ruby en sus diversas versiones.

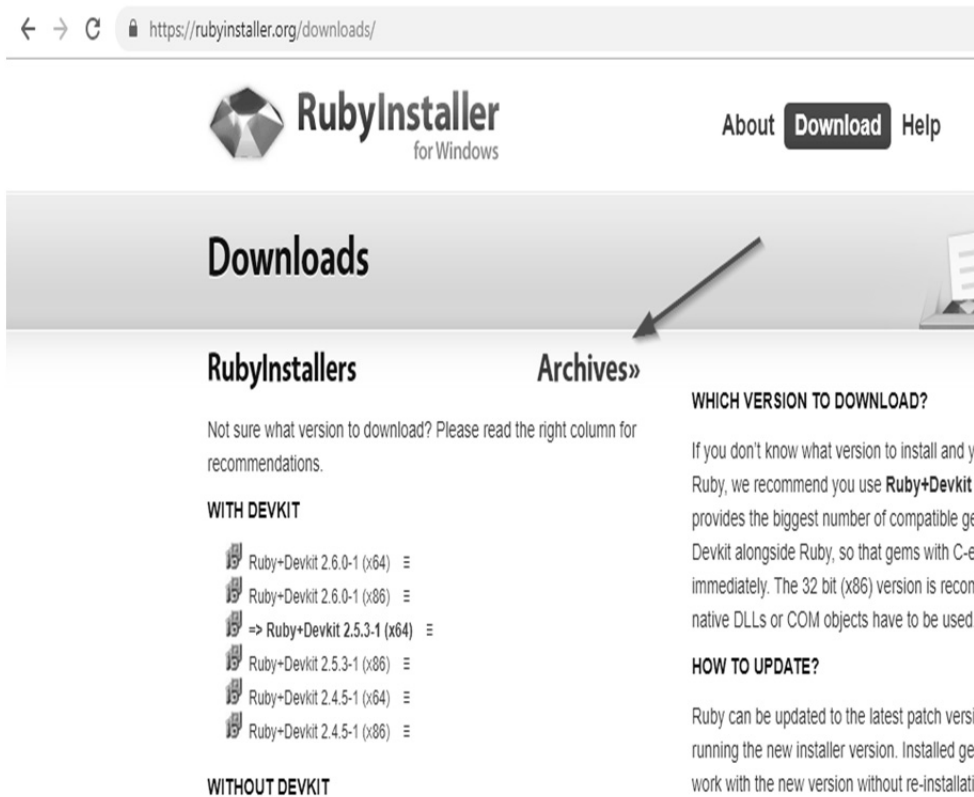
El enlace a la página para realizar la descarga es el siguiente:

<https://rubyinstaller.org/>

En la lista de versiones disponibles, encontrarás dos grupos: *Ruby sin Devkit* y *Ruby con Devkit*, deberás descargar: *Ruby + Devkit 2.4.4* con terminación en *x64* si tu computadora es de 64 bits o con terminación *x86* si tu computadora es de 32 bits. Si no estás seguro, puedes ir al panel de control y ver la información de tu SO.

Toma en cuenta que la página se actualiza constantemente ya que salen nuevas versiones. Por lo tanto, en la página del enlace anterior solo mostrará las versiones más recientes, pero sería bueno que leyeras lo que en ella se encuentra, te será de mucha ayuda. Si deseas ver alguna versión más antigua deberás hacer clic en el enlace “Archives” y te direccionará a un listado más amplio donde encontrarás desde la versión 1.8.7 hasta la más actual y estable.

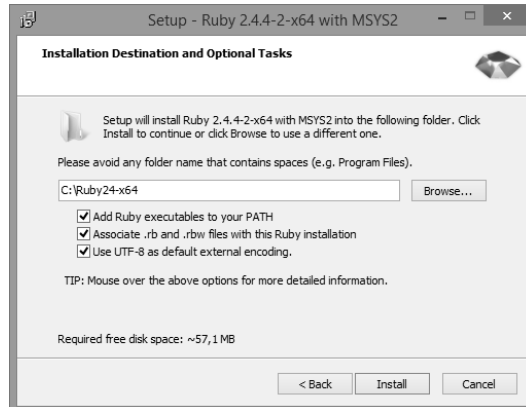
La página de RubyInstaller es la que mostraré a continuación. En ella está señalada con una flecha el enlace mencionado.



**Fig. 1** Página oficial para la descarga del instalador de Ruby para Windows

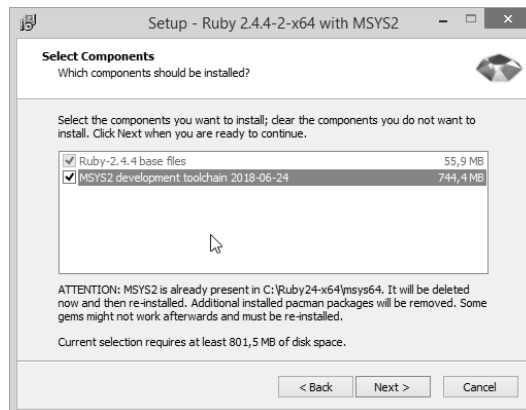
Cuando se haya descargado el instalador de la página web por completo, deberás de ejecutar el instalador haciendo doble clic sobre él, esto hará que muestre el asistente de instalación. Cuando aparezca, deberás de aceptar los términos de licencia y dar clic en “Next”.

Deja que el asistente instale Ruby en la ruta que se muestra, asegurándote de que las casillas de verificación “Add Ruby executable to your PATH”, “Associate.rb and .rbw files with this Ruby instalation” y “Use UTF-8 as default external encoding” estén marcadas para continuar con la instalación.



**Fig. 2 Destino de instalación de RubyInstaller**

Debido a que descargamos Ruby con Devkit debes de dejar seleccionada la casilla “MSYS2 development toolchain 2018-06-24” y dar clic en “Next”.



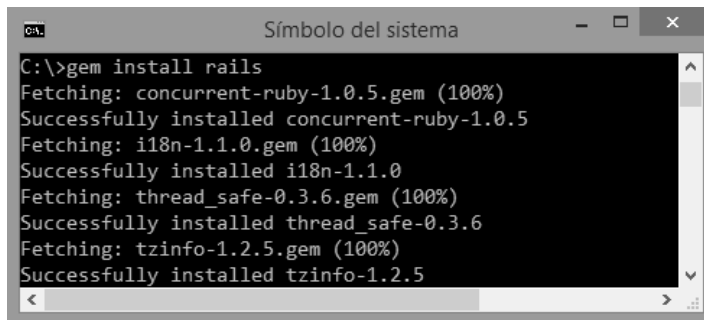
**Fig. 3 Instalación del componente MSYS2**

Después, habrá una barra de progreso para informarnos el tiempo faltante de forma gráfica. Espera a que esta termine. La instalación concluida merece el par de minutos de espera.

Hemos instalado Ruby en Windows, eso nos podría bastar, sin embargo, puedes comprobarlo con el comando `ruby -v` en el símbolo del sistema o CMD.

Para la instalación de Rails simplemente debes de escribir en el símbolo del sistema el comando `gem install Rails` y esperar pacientemente hasta que finalice la instalación. Sin embargo, te podrás dar cuenta en la imagen que sigue después de esta que este comando instala la versión de Rails más reciente y estable.

Si lo haces de esta manera, después de ingresar el comando anterior tu terminal se verá similar o igual a lo siguiente:



```
C:\>gem install rails
Fetching: concurrent-ruby-1.0.5.gem (100%)
Successfully installed concurrent-ruby-1.0.5
Fetching: i18n-1.1.0.gem (100%)
Successfully installed i18n-1.1.0
Fetching: thread_safe-0.3.6.gem (100%)
Successfully installed thread_safe-0.3.6
Fetching: tzinfo-1.2.5.gem (100%)
Successfully installed tzinfo-1.2.5
```

**Fig. 4 Instalación de Ruby on Rails (Windows)**

Durante la instalación hay un momento en el que pareciera que no avanza, pero no interrumpas la instalación, te comento que esto puede retrasarse un par de minutos, sé paciente. 😊

Al igual que para Ruby, puedes comprobar la versión de Rails que se instaló con el comando `rails -v`.

Si quieres instalar una versión en concreto, por ejemplo, la versión 5.2.2, entonces escribe el comando **`gem install rails -v 5.2.2`**. Donde 5.2.2 es la versión a instalar, sustitúyela por la que desees. Ya sea que instales precisamente esta o la más reciente y estable en teoría no deberías de tener ningún problema, pero si quieres sentirte seguro, nosotros usaremos la **versión 5.2.2**.

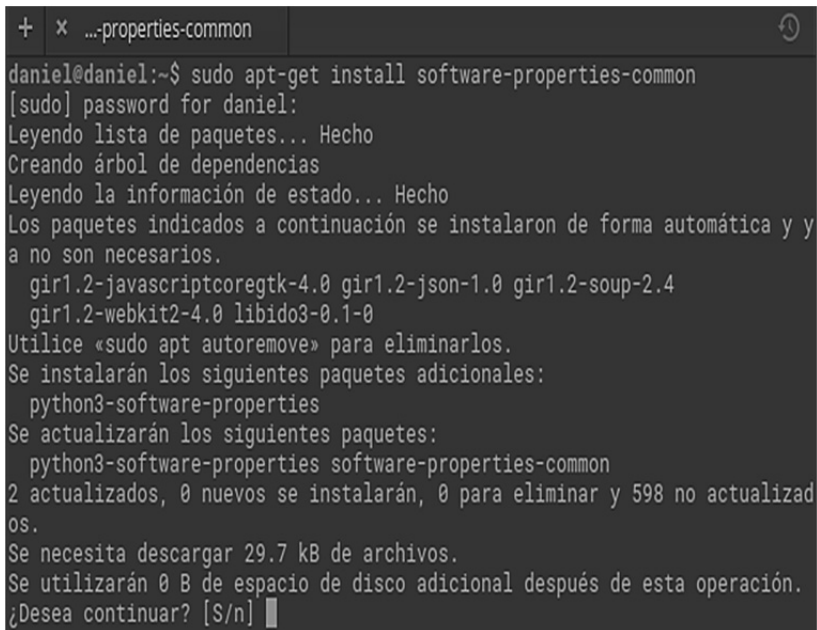
## Instalación en GNU/Linux

Linux es un SO muy amado por los desarrolladores y probablemente ya te lleves muy bien con la *terminal*, por lo tanto, escribir comandos no debería hacerte entrar en pánico. Si tienes Windows y a pesar de ello quieres mejor trabajar en Linux, te mostraré cómo instalar todo lo necesario para poder trabajar con Ruby on Rails en *Elementary OSX 0.4.1 Loki*. Este SO está basado en *Ubuntu 16.04.2* y es una buena opción para introducirse en el mundillo de Linux. Además, le va a caer muy bien a tus ojos ya que tiene un buen diseño.

Pero lo anterior no quiere decir que la siguiente guía de instalación esté dedicada sola para los que tengan *Elementary OS*. Si tu distribución favorita está basada en Ubuntu, podrás escribir los mismos comandos en tu terminal y tendrás con toda seguridad Ruby y Rails “al cien”.

Ahora bien, para la instalación de Ruby bastaría instalar algunas dependencias y escribir en la terminal un par de comandos y ¡listo!, pero antes de instalar Ruby instalaremos *RVM (Ruby Version Manager)* para poder administrar múltiples versiones de Ruby. El tiempo que ahora inviertas se lo agradecerás después a *RVM*. *RVM* te permitirá tener más de una versión de Ruby en tu computadora y usar la que tú le indiques en un momento dado. De esta manera podrías tener un proyecto basado en una versión “X” y otro en otra versión sin que estas interfieran.

Antes de instalar *RVM* deberás teclear el comando `sudo apt-get install software-properties-common` en la terminal, por cierto, la terminal de Elementary OS se llama *Pantheon* y te comento esto porque puede resultarte un poco extraña. Esto es necesario para poder agregar un repositorio PPA. PPA significa archivo de paquete personal y esos archivos están alojados en un repositorio de software específico.

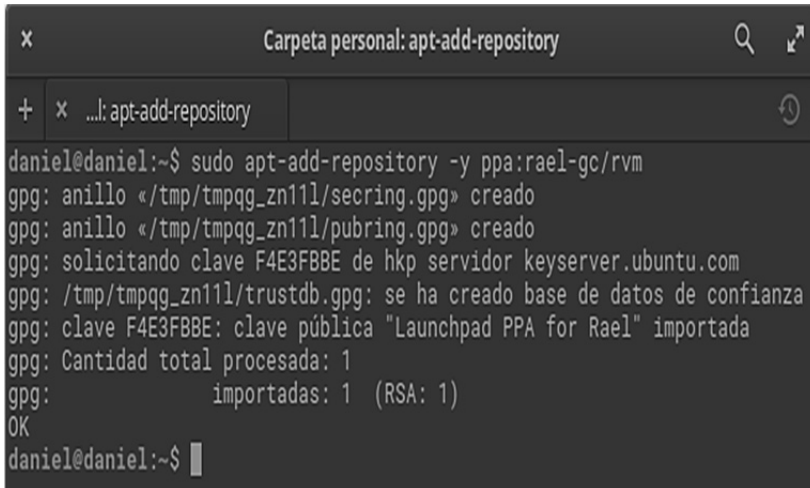
A terminal window titled "...-properties-common" showing the execution of the command "sudo apt-get install software-properties-common". The output includes prompts for a password, progress messages like "Leyendo lista de paquetes... Hecho", and a list of additional packages to be installed: "python3-software-properties". It also shows the disk space requirements and asks for confirmation to continue with "[S/n]".

```
daniel@daniel:~$ sudo apt-get install software-properties-common
[sudo] password for daniel:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
 gir1.2-javascriptcoregtk-4.0 gir1.2-json-1.0 gir1.2-soup-2.4
 gir1.2-webkit2-4.0 libido3-0.1-0
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
 python3-software-properties
Se actualizarán los siguientes paquetes:
 python3-software-properties software-properties-common
2 actualizados, 0 nuevos se instalarán, 0 para eliminar y 598 no actualizados.
Se necesita descargar 29.7 kB de archivos.
Se utilizarán 0 B de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n]
```

**Fig. 5** Instalación de *software-properties-common*

Después de escribir el comando y de presionar enter, te preguntará si deseas continuar. Para confirmar que sí, deberás de ingresar una *S* y pulsar nuevamente la tecla *Intro*.

Para poder agregar el PPA de RVM deberás escribir `sudo apt-add-repository -y ppa:rael-gc/rvm`.



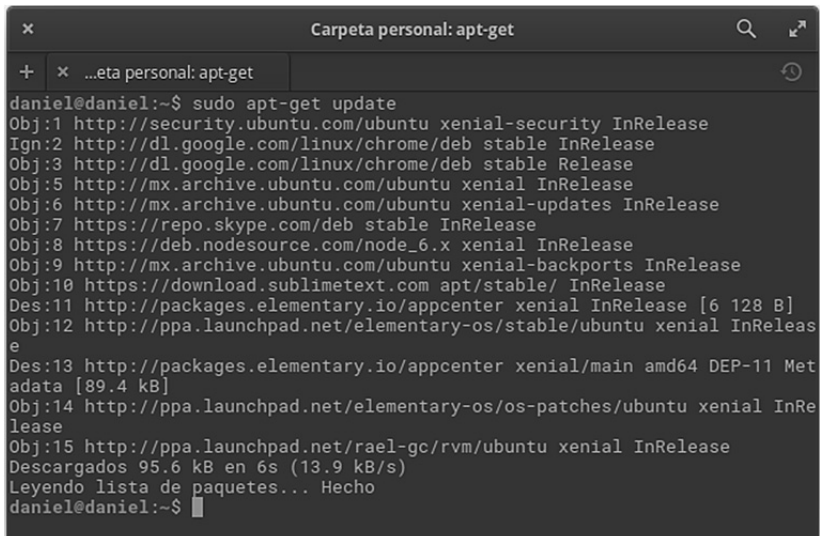
```

x          Carpeta personal: apt-add-repository
+ x ...: apt-add-repository
daniel@daniel:~$ sudo apt-add-repository -y ppa:rael-gc/rvm
gpg: anillo «/tmp/tmpqg_zn11l/secring.gpg» creado
gpg: anillo «/tmp/tmpqg_zn11l/pubring.gpg» creado
gpg: solicitando clave F4E3FBBE de hkp servidor keyserver.ubuntu.com
gpg: /tmp/tmpqg_zn11l/trustdb.gpg: se ha creado base de datos de confianza
gpg: clave F4E3FBBE: clave pública "Launchpad PPA for Rael" importada
gpg: Cantidad total procesada: 1
gpg:          importadas: 1 (RSA: 1)
OK
daniel@daniel:~$

```

**Fig. 6 Comando para agregar el PPA de RVM**

Posteriormente deberás actualizar los repositorios con el comando bien conocido por todos, el comando `sudo apt-get update` como verás a continuación:



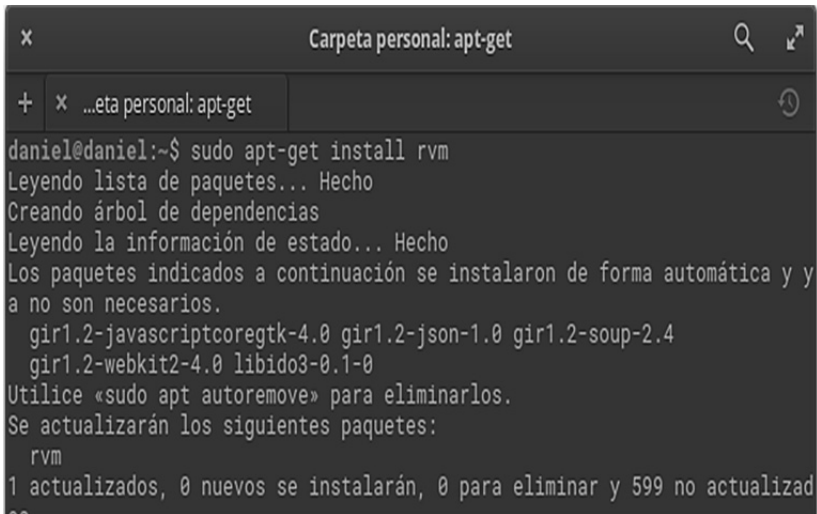
```

x          Carpeta personal: apt-get
+ x ...eta personal: apt-get
daniel@daniel:~$ sudo apt-get update
Obj:1 http://security.ubuntu.com/ubuntu xenial-security InRelease
Ign:2 http://dl.google.com/linux/chrome/deb stable InRelease
Obj:3 http://dl.google.com/linux/chrome/deb stable Release
Obj:5 http://mx.archive.ubuntu.com/ubuntu xenial InRelease
Obj:6 http://mx.archive.ubuntu.com/ubuntu xenial-updates InRelease
Obj:7 https://repo.skype.com/deb stable InRelease
Obj:8 https://deb.nodesource.com/node_6.x xenial InRelease
Obj:9 http://mx.archive.ubuntu.com/ubuntu xenial-backports InRelease
Obj:10 https://download.sublimetext.com apt/stable/ InRelease
Des:11 http://packages.elementary.io/appcenter xenial InRelease [6 128 B]
Obj:12 http://ppa.launchpad.net/elementary-os/stable/ubuntu xenial InRelease
Des:13 http://packages.elementary.io/appcenter xenial/main amd64 DEP-11 Metadata [89.4 kB]
Obj:14 http://ppa.launchpad.net/elementary-os/os-patches/ubuntu xenial InRelease
Obj:15 http://ppa.launchpad.net/rael-gc/rvm/ubuntu xenial InRelease
Descargados 95.6 kB en 6s (13.9 kB/s)
Leyendo lista de paquetes... Hecho
daniel@daniel:~$

```

**Fig. 7 Actualización de repositorios**

Una vez actualizados los repositorios, procedemos a instalar *RVM* con *sudo apt-get install rvm*.

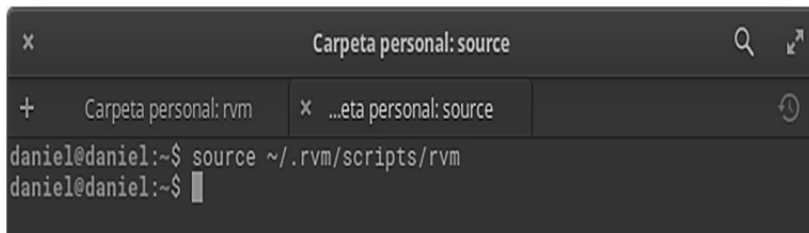


```
x          Carpeta personal: apt-get
+ x ...eta personal: apt-get
daniel@daniel:~$ sudo apt-get install rvm
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
 gir1.2-javascriptcoregtk-4.0 gir1.2-json-1.0 gir1.2-soup-2.4
 gir1.2-webkit2-4.0 libido3-0.1-0
Utilice «sudo apt autoremove» para eliminarlos.
Se actualizarán los siguientes paquetes:
 rvm
1 actualizados, 0 nuevos se instalarán, 0 para eliminar y 599 no actualizados
```

**Fig. 8 Instalación de RVM**

En este punto deberás de reiniciar tu computadora por los cambios efectuados para que todo funcione correctamente.

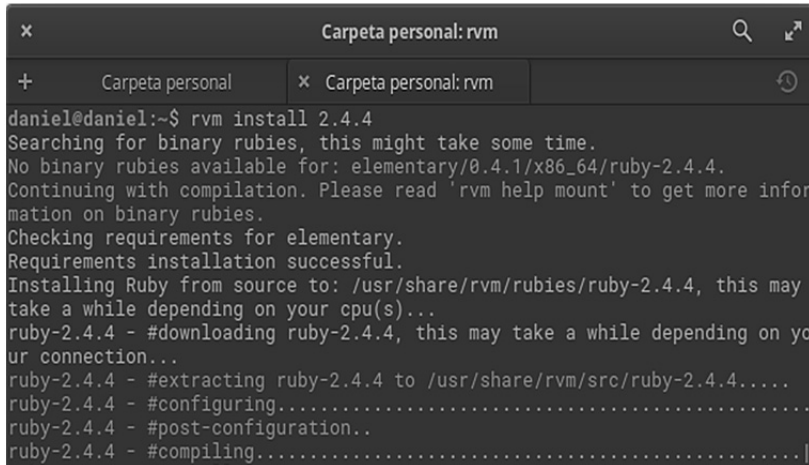
Abre nuevamente la terminal y escribe *source ~/.rvm/scripts/rvm* para cargar RVM.



```
x          Carpeta personal: source
+ Carpeta personal: rvm x ...eta personal: source
daniel@daniel:~$ source ~/.rvm/scripts/rvm
daniel@daniel:~$
```

**Fig. 9 Carga de RVM como función**

Ahora ya podrás instalar las versiones de Ruby que gustes. Para la versión de *Ruby 2.4.4* escribe: *rvm install 2.4.4*. Mientras se instala puedes ir por un café y tomarte unos cinco o diez minutos, en ese tiempo seguramente ya habrá terminado.



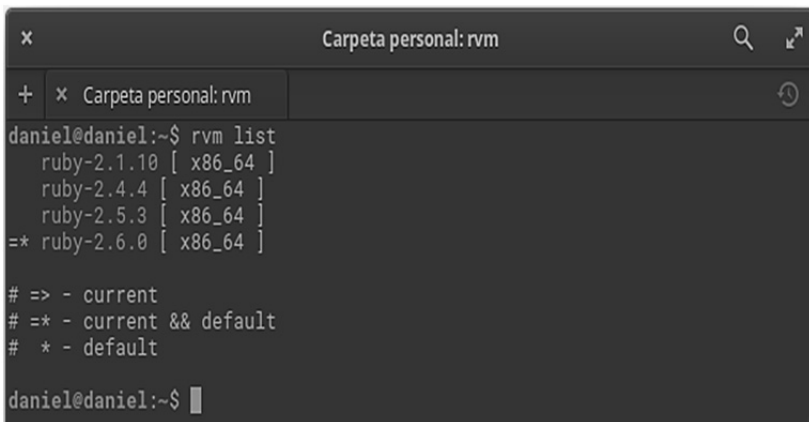
```

x          Carpeta personal: rvm
+          Carpeta personal  x Carpeta personal: rvm
daniel@daniel:~$ rvm install 2.4.4
Searching for binary rubies, this might take some time.
No binary rubies available for: elementary/0.4.1/x86_64/ruby-2.4.4.
Continuing with compilation. Please read 'rvm help mount' to get more information on binary rubies.
Checking requirements for elementary.
Requirements installation successful.
Installing Ruby from source to: /usr/share/rvm/rubies/ruby-2.4.4, this may take a while depending on your cpu(s)...
ruby-2.4.4 - #downloading ruby-2.4.4, this may take a while depending on your connection...
ruby-2.4.4 - #extracting ruby-2.4.4 to /usr/share/rvm/src/ruby-2.4.4....
ruby-2.4.4 - #configuring.....
ruby-2.4.4 - #post-configuration..
ruby-2.4.4 - #compiling.....

```

**Fig. 10** Instalación de Ruby 2.4.4 con RVM

Si quieres ver las versiones de Ruby que se han instalado en el directorio de RVM, *rvm list* te mostrará todas ellas. Por ser tu primera vez solo te mostrará una, la que hayas instalado. Sin embargo, te mostraré las que yo tengo instaladas.



```

x          Carpeta personal: rvm
+          Carpeta personal: rvm
daniel@daniel:~$ rvm list
ruby-2.1.10 [ x86_64 ]
ruby-2.4.4 [ x86_64 ]
ruby-2.5.3 [ x86_64 ]
=* ruby-2.6.0 [ x86_64 ]

# => - current
# =* - current && default
# * - default

daniel@daniel:~$ █

```

**Fig. 11** Lista de las versiones de Ruby instaladas

Al ser la primera vez que se instala RVM y la única versión de Ruby en su lista, no hace falta indicarle que use una versión “X”, pero en cuanto añadas una segunda versión (como yo), sí será necesario especificar la que quieres usar. Para hacer esto, escribe el comando *rvm use*, seguida de la versión que quieres usar. Haré esto con la versión 2.5.3.