



# Test de infraestructura “Teuton”



# Agradecimientos

AgileCanarias, comunidad Teuton, Software Libre.



# Presentación

David Vargas (@dvarrui)

*Ruby, OpenSUSE, software libre y Starwars*

# El camino



# OpenSUSE

- GNU/Linux OpenSUSE, y la comunidad del Software Libre
- Ruby para scripting, YAST, etc.
- **Si te encuentras un problema y lo puedes solucionar... ¡adelante!**

# Concurso de programación - PROGRAMAME

[ProgramaMe](#) [Información general](#) [Problemas](#) [Regionales](#) [Nacional](#) [Ediciones anteriores](#) [Contacto](#)

 ProgramaMe: Concurso de Programación para Ciclos Formativos

Estás aquí: / [Nacional 2022](#) / Inicio



## Concurso Nacional!

Organizado desde la Facultad de Informática de la UCM

28 de abril de 2022

[Inicio](#)  
[Ayuda a Ucrania](#)  
[Información para equipos](#)  
[Información para centros](#)  
[Inscripción](#)  
[Reglamento](#)

## ProgramaMe en tiempos de pandemia

Ponemos en marcha la final de ProgramaMe con dos elementos a nivel mundial que nos obligan a adaptarnos a las circunstancias:

- *Pandemia de la COVID-19*: la realización de actividades presenciales multitudinarias sigue sin estar completamente despejada, por lo que, como [el curso pasado](#), esta edición tendrá el modelo de *final online multisitio*.
- *Invasión rusa a Ucrania*: el 24 de febrero fuerzas terrestres rusas entraron en Ucrania y desde entonces se vive una crisis humanitaria. Desde ProgramaMe queremos ayudar [Acnur](#), la agencia de la ONU para los refugiados, a [recaudar fondos](#).



[Prueba tus soluciones](#)

Puedes probar tus soluciones en [¡Acepta el reto!](#), el juez on-line con [problemas de ediciones anteriores](#).

[Programame 2.0](#)

Visita nuestro [blog](#) con

# Correcciones automáticas basadas en tests

- 🚦 TDD: Desarrollo guiado por pruebas de software.
- 🚦 Lograr código limpio que funcione.
- 🚦 Garantizar que el software cumple con los requisitos que se han establecido.

“Clean Code” (Robert C. Martin)

“Código limpio” (Carlos Ble)

# Code testing - Example 1 - Recursive

```
def factorial(n)
  if (n == 0 || n == 1)
    return 1
  else
    return n * factorial(n - 1)
  end
end
```

```
require 'test/unit'
require_relative 'factorial_recursive'

class TestFactorial < Test::Unit::TestCase
  def test_base_case
    assert_equal(1, factorial(0))
    assert_equal(1, factorial(1))
  end

  def test_2_3_4
    assert_equal(2, factorial(2))
    assert_equal(6, factorial(3))
    assert_equal(24, factorial(4))
  end
end
```



# Code testing - Example 1 - Recursive

- Los paréntesis son opcionales cuando son redundantes.
- El if admite la forma de lenguaje natural.
- Toda sentencia es una expresión. Esto es, devuelven algo. El último return sobra.

```
def factorial(n)
  return 1 if n == 0 || n == 1
  n * factorial(n - 1)
end
```

```
require 'test/unit'
require_relative 'factorial_recursive'

class TestFactorial < Test::Unit::TestCase
  def test_base_case
    assert_equal(1, factorial(0))
    assert_equal(1, factorial(1))
  end

  def test_2_3_4
    assert_equal(2, factorial(2))
    assert_equal(6, factorial(3))
    assert_equal(24, factorial(4))
  end
end
```

# Ruby Lover

- Tipado dinámico
- OO real (todo es un objeto) y de primera clase
- Metaprogramación
- “Parece” lenguaje natural (inglés).
- Legible y fácil de escribir.

# Code testing - Example 2 - Object methods

```
def factorial(n) = (1..n).to_a.reduce(:*)
```

- Una función de una sentencia se puede poner en una línea.
- (1..n) es azúcar sintáctico de Range.new(1,n). Es un objeto.
- Los métodos de los objetos se pueden encadenar. El método to\_a devuelve un objeto Array
- El método reduce del Array reduce la lista usando el método indicado.
- Los simbolos existen en muchos lenguajes.

```
require 'test/unit'
require_relative 'factorial_recursive'

class TestFactorial < Test::Unit::TestCase
  def test_base_case
    # assert_equal(1, factorial(0))
    assert_equal(1, factorial(1))
  end

  def test_2_3_4
    assert_equal(2, factorial(2))
    assert_equal(6, factorial(3))
    assert_equal(24, factorial(4))
  end
end
```

# Code testing - Example 2 - Object methods

- Esto es lo mismo de antes pero poniendo los métodos de forma más “bonita” o “estilo funcional”.
- reduce devuelve un valor que es el return de la función.

```
def factorial(n)
  (1..n)
  .to_a
  .reduce(:*)
end
```

```
require 'test/unit'
require_relative 'factorial_recursive'

class TestFactorial < Test::Unit::TestCase
  def test_base_case
    # assert_equal(1, factorial(0))
    assert_equal(1, factorial(1))
  end

  def test_2_3_4
    assert_equal(2, factorial(2))
    assert_equal(6, factorial(3))
    assert_equal(24, factorial(4))
  end
end
```

# EL CICLO DE LA EVALUACIÓN



**FIN. HASTA LA PRÓXIMA.**

# Problema

- Reducir las tareas repetitivas y aburridas
- DevOps: Procesos reproducibles y automatizados
- Jugar es divertido (Gamificación)

# Idea

- Ruby: programar un nuevo proyecto
- Aplicar buenas prácticas al proceso
- Herramienta para facilitar las tareas
- ¡¡¡Premio asegurado!!!

# Principios

- TDD (Test-Driven Development)
- DRY (Don't repeat yourself)
- DevOps (Automatización de procesos)
- RubyLover!



# Teuton: Caso de uso



¿Qué es Teuton?

- Programa multiplataforma.
- Licencia Software libre.
- Test de infraestructura.



¿Qué resuelve?

1. Revisar nuestra infraestructura como si fuera código.
2. Automatizar las correcciones de las MV remotas de los alumnos.

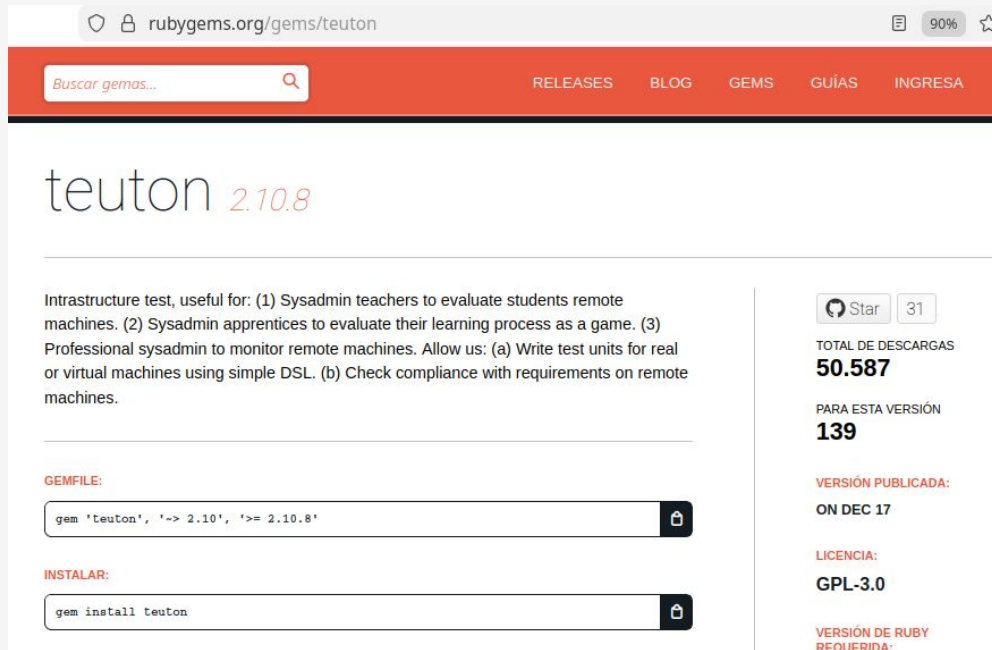


# RubyGems - “gem install teuton”

Aplicar tests para mantener la calidad de...

- El código.
- ¿La infraestructura?
- ¿Los scripts?
- ¿Configuraciones?
- ¿Entornos?

**¡Cualquier entorno CLI!**



The screenshot shows the RubyGems page for the 'teuton' gem. The browser address bar shows 'rubygems.org/gems/teuton'. The page has a red header with a search bar and navigation links: RELEASES, BLOG, GEMS, GUÍAS, and INGRESA. The main content area displays the gem name 'teuton' with the version '2.10.8'. Below this is a description: 'Infrastructure test, useful for: (1) Sysadmin teachers to evaluate students remote machines. (2) Sysadmin apprentices to evaluate their learning process as a game. (3) Professional sysadmin to monitor remote machines. Allow us: (a) Write test units for real or virtual machines using simple DSL. (b) Check compliance with requirements on remote machines.' To the right of the description is a 'Star' button with a count of 31. Below the description are two input fields: 'GEMFILE:' with the text 'gem 'teuton', '~> 2.10', '>= 2.10.8'' and 'INSTALAR:' with the text 'gem install teuton'. On the right side of the page, there is a sidebar with the following information: 'TOTAL DE DESCARGAS: 50.587', 'PARA ESTA VERSIÓN: 139', 'VERSIÓN PUBLICADA: ON DEC 17', 'LICENCIA: GPL-3.0', and 'VERSIÓN DE RUBY REQUERIDA:'.

# Instalación - CLI

- Instalar Ruby
- `gem install teuton`, instalar la gema

```
$ teuton help
```

Commands:

```
teuton [run] [OPTIONS] DIRECTORY    # Run test from directory
teuton check [OPTIONS] DIRECTORY    # Check test and config file content
teuton config [OPTIONS] DIRECTORY   # Suggest configuration.
teuton help [COMMAND]                # Describe available commands or one specific command
teuton new DIRECTORY                 # Create skeleton for a new project
teuton readme DIRECTORY               # Show README extracted from test contents
teuton version                       # Show the program version
```

# ❖ **EVALUACIÓN AUTOMÁTICA CON TEUTON** ❖

**1. ¡A PROGRAMAR!**



**2. ¡AYUDANDO EN CLASE!**



**3. ¡TEUTON AL RESCATE!**

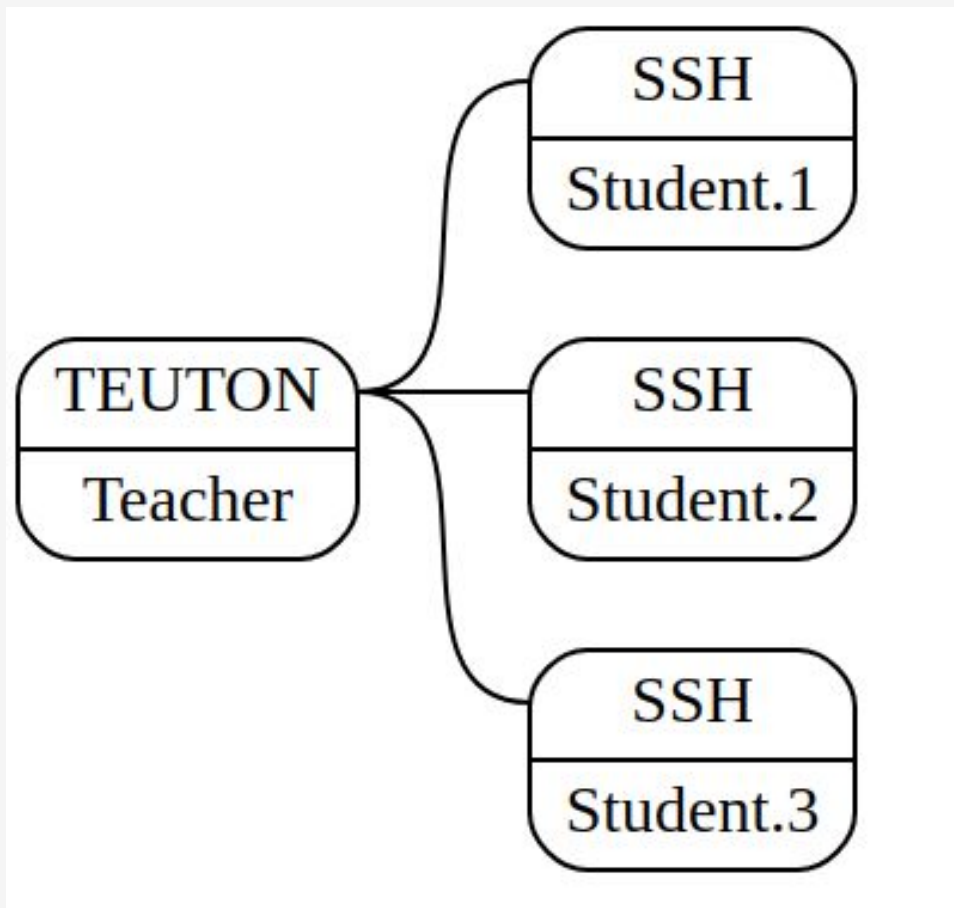


**¡EVALUACIÓN RÁPIDA Y JUSTA!**

# Entorno de trabajo

Un test consiste de:

- **config.yaml**: Fichero de configuración.
- [start.rb](#): Test (target, run ,expect, etc.)



# Ruby Lover

- Tipado dinámico
- POO (todo es un objeto). “Ciudadanos de primera clase”
- Metaprogramación
- Escribir de forma “natural” y leer de forma “natural”.
- ¡Programar poesía!





# DSL (Domain Specific Language)

Beneficios:

- Creado para resolver problemas en un campo concreto.
- Permite a los expertos del dominio crear y entender código (modelar problemas del dominio) más fácilmente
- Ejemplos: SQL, CSS, el álgebra en matemáticas. Mayor expresividad y concisión para su dominio específico.

¿Y si creamos un DSL para resolver nuestro problema?



# DSL de Teuton

- **target:** Descripción del objetivo
- **run:** Comando que ejecutamos para comprobar.
- **expect:** Lo que esperamos “ver” en la salida del comando anterior.

```
group "Learn about targets" do
  target "Create user obiwan"
  run "id obiwan"
  expect ["uid=", "(obiwan)", "gid="]

  target "Delete user vader"
  run "id vader"
  expect_fail
end
```

```
> systemctl status libvirtd
● libvirtd.service - libvirt legacy monolithic daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; disabled; preset: disabled)
   Active: active (running) since Sat 2026-01-03 19:29:58 WET; 8s ago
     Invocation: 994cbd2a3dd84a9292c104052cd1e017
  TriggeredBy: ● libvirtd.socket
                ● libvirtd-ro.socket
                ● libvirtd-admin.socket
        Docs: man:libvirtd(8)
              https://libvirt.org/
    Main PID: 10117 (libvirtd)
      Tasks: 23 (limit: 32768)
        CPU: 1.104s
    CGroup: /system.slice/libvirtd.service
            └─10117 /usr/sbin/libvirtd --timeout 120
              └─10269 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --l
                └─10270 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --l

lines 1-16/16 (END)
```

1. Si tienes el comando para comprobar el target
2. Y sabes lo que se debe "mirar" en la salida. **¡Ya tienes el test!**

# Ejemplos de tests

Repos de ejemplos:

Ejemplos documentados:

- <https://github.com/teuton-software/teuton/examples>

Repo de ejemplos:

- <https://github.com/dvarrui/teuton-tests>

Tutorial: Crear un test para una práctica Nginx apoyándonos en la IA.

- <https://github.com/dvarrui/charlas/blob/main/teuton/nginx/docs/index.md>

```
# File: config.yaml
# Desc: configutarion file
---
global:
  host1_username: root
cases:
- tt_members: student_1
  host1_ip: 192.168.1.201
  host1_password: secret_1
- tt_members: student_2
  host1_ip: 192.168.1.202
  host1_password: secret_2
- tt_members: student_3
  host1_ip: 127.0.0.1
  host1_password: secret_3
```

```
# File: start.rb
# Desc: test definitions
group "Remote host" do
  target "Create user root"
  run "id root", on: :host1
  expect ["uid=", "(root)", "gid="]

  target "Delete user vader"
  run "id vader", on: :host1
  expect_fail
end

play do
  show
  export
end
```

```
expect ["obiwan", "kenobi"]  
expect_first "episode"  
expect_last "the end"  
expect_one "rogue"  
expect_one ["obiwan","kenobi"]  
expect /Obiwan|obi-wan/
```

```
expect_none "vader"  
expect_none ["darth", "vader"]  
expect_none  
expect_nothing
```

```
expect_ok  
expect_fail  
expect_exit NUMBER  
expect_exit MIN..MAX
```

```
expect_sequence do  
  find "A"  
  find "B"  
  find "C"  
end
```

```
expect_sequence do  
  find "A"  
  next_to "B"  
  next_to "C"  
end
```

```
expect_sequence do  
  find "A"  
  next_to "B"  
  ignore 2  
  next_to "C"  
end
```

# DEMO!

nginx/v03.custom

# Future roadmap

1. Documentación
2. Difusión
3. Feedback
4. teuton-server
5. Orientar a monitorización

# Proyectos y Repositorios

[github.com/teuton-software/teuton](https://github.com/teuton-software/teuton)

[github.com/teuton-software/teuton-get](https://github.com/teuton-software/teuton-get)

[github.com/dvarrui/teuton-tests](https://github.com/dvarrui/teuton-tests)





# Ahora, pregunto yo! ;-)

github: @dvarrui  
telegram: @dvarrui  
email: dvarrui@proton.me



**¡Muchas gracias!**

