Universidad de La Laguna

ghedsh: Un intérprete de comandos para GitHub Education
GitHub Education Shell: ghedsh.

Autor: Carlos de Armas Hernández
Director: Casiano Rodríguez León

Escuela Superior de Ingeniería y Tecnología
Departamento de Ingeniería Informática y de Sistemas
Universidad de La Laguna

12 de julio de 2018

Índice

Introducción I

¿Qué es "ghedsh"?

Es una gema Ruby que consiste en un intérprete de comandos desarrollado para integrar las metodologías de GitHub Education, viendo las organizaciones como aulas y los repositorios como las asignaciones de los alumnos.

Introducción II

En cuanto a herramientas similares, existen las siguientes:

- *Teachers Pet*.
- *GitHub Classroom*.
- *ghi* (GitHub Issues).
- *ghs* (GitHub Search).

Introducción III

Desarolladas por *GitHub*:

- **Teachers Pet**. *CLI* desarrollado previamente a *Classroom*.
  - Inconveniente: los comandos se hacían excesivamente largos en determinados casos. Cayó en desuso y se dejó de desarrollar.
- **GitHub Classroom**. Una plataforma web que simplifica la configuración de las aulas y las asignaciones. Sin embargo, existen algunas limitaciones:
  - Actualmente no dispone de alguna funcionalidad que permita al profesor crear un repositorio de evaluación.
  - No da soporte a herramientas de integración continua (*Travis CI, CircleCI, Jenkins ...*).
  - El sistema para añadir información adicional del alumno es incómodo de usar.

Introducción IV

Desarrolladas por la comunidad:

- **ghi** (GitHub Issues). Permite gestionar las incidencias (issues) de los repositorios desde la terminal del usuario.
- **ghs** (GitHub Search). Permite realizar búsquedas de repositorios alojados en GitHub.

Objetivos

Esta segunda versión de *ghedsh* busca mejorar el código fuente de la primera versión, teniendo en cuenta aspectos como la mantenibilidad del código y facilitar la incorporación de nuevas funcionalidades.

Por otro lado, una de las prioridades de esta herramienta es dar soporte al proceso de evaluación.

Tecnologías empleadas

Lenguaje de programación: **Ruby**

Gestión de dependencias: **Bundler**

Testing: **RSpec**

Desarrollo del proyecto

Dividimos el desarrollo del proyecto en dos fases bien diferenciadas:

- Análisis. Identificar aquellas partes mejorables del diseño e implementación iniciales.
- Refactorización. Proceso llevado a cabo para solucionar las debilidades anteriores.

Primera fase. Análisis

Tras estudiar el código de la primera versión de la gema se han detectado diversos *code smell*.

¿Qué es un code smell?

Se define como cualquier característica del código fuente que, posiblemente, indica un problema más profundo. No son considerados como bugs.

Primera fase. Análisis

Switch Smell

```
USER=1
ORGS=2
USER_REPO=10
ORGS_REPO=3
TEAM=4
ASSIG=6
TEAM_REPO=5
```

```
case
  when op == "exit" then ex=0
    @sysbh.save_memory(config_path,@config)
    s.save_cache(config_path,@config)
    s.remove_temp("#{ENV['HOME']}/.ghedsh/temp")
  when op.include?("help") && opcd[0]=="help" then self.help(opcd)
  when op == "orgs" then self.orgs()
  when op == "cd .."
    if @deep==ORGS then t.clean_groupsteams() end ##cleans groups cache
    self.cdback(false)
  when op.include?("cd assig") && opcd[0]=="cd" && opcd[1]=="assig" && opcd.size==3
    if @deep==ORGS
      self.cdassig(opcd[2])
    end
  when op == "people" then self.people()
  when op == "teams"
    if @deep==ORGS
      t.show_teams_bs(@client,@config)
    end
  when op == "commits" then self.commits()
  when op == "issues"
    if @deep==ORGS_REPO || @deep==USER_REPO || @deep==TEAM_REPO
      @issues_list=r.show_issues(@client,@config,@deep)
    end
  when op == "col" then self.collaborators()
  when op == "forks" then self.show_forks()
```

Long Method se clasifica a nivel de método. Como su propio nombre indica, consiste en un método que ha crecido demasiado y dificulta saber qué es lo que realmente hace.

Large Class se clasifica dentro de los smells a nivel de clases. Indica que una clase ha crecido excesivamente en tamaño (God Object). Su funcionalidad puede descomponerse en clases más pequeñas.

Refactorización

Resultados obtenidos I

Caso de uso I

Conclusions and future work lines I

Bibliografía I


"Node JS."
https://nodejs.org/es/docs/.


"Github REST API V3."
https://developer.github.com/v3/.


"Express JS."
http://expressjs.com/es/.


"Pug JS."
https://pugjs.org/api/getting-started.html.


"Mongoose JS."
http://mongoosejs.com/.


"Heroku."
https://devcenter.heroku.com/.


"NPM."
https://www.npmjs.com/.

Fin de la presentación

Gracias por su atención