E.S. Ingeniería y Tecnología

Ingeniería Informática

Lenguajes y Sistemas Informáticos

Lenguajes y Paradigmas de Programación

Práctica de laboratorio **#7**

Esta práctica de laboratorio se ha de realizar utilizando el paradigma de **Programación Orientada a Objetos**, la metodología de **Desarrollo Dirigido por Pruebas**, el sistema de *control de versiones git*, el lenguaje de programación Ruby, la herramienta de gestión de dependencias Bundler, la herramienta de comprobación continua Guard, las herramientas de generación de documentación rdoc o YARD.

1. Aceptar la tarea asignada en `GitHub Classroom` denominada "Gema".

   (`https://classroom.github.com/a/ePZukRWX`)

2. El proyecto se ha de desarrollar haciendo uso del control del versiones, por lo tanto, ha de contar con más de una confirmación. Además, se ha de trabajar con al menos una rama para el desarrollo y una para la documentación, aparte de la rama principal. Finalmente, la rama remota ha de estar enlazada al repositorio que se crea al aceptar la tarea asignada en el apartado 1.

3. Crear la estructura del 'directorio de trabajo' haciendo uso de ***Bundler***.

   a) Instalar ***Bundler***.

   ```
   gem install bundler
   ```

   b) Crear la estructura de la gema.

   ```
   bundle gem <nombre-de-gema>
   ```

4. Utilizar la herramienta ***Guard*** de ***Comprobación Continua*** para el desarrollo del código Ruby. Se ha de configurar de manera que permita la ejecución de las pruebas definidas con ***Rspec*** cuando se modifiquen de la siguiente manera:

   a) Añadir ***Guard*** como una dependencia de desarrollo en el fichero de especificaciones de la gema `.gemspec`.

   ```
   spec.add_development_dependency "guard"
   spec.add_development_dependency "guard-rspec"
   spec.add_development_dependency "guard-bundler"
   ```

   b) Instalar ***Guard*** a través de *Bundler*.

   ```
   bin/setup
   ```

   c) Generar un fichero de guardia por defecto (`Guardfile`).

   ```
   bundle exec guard init
   ```

   d) Ejecutar `Guard` a través de *Bundler* en una terminal.

   ```
   bundle exec guard
   ```

5. Desarrollar una gema Ruby para representar un aparcamiento. Ha de contar con una clase para representar los datos del aparcamiento propiamente dicho, constantes para representar dos estados de ocupación y una función para mostrar el estado de ocupación de un aparcamiento dado.

   Para el desarrollo de la clase utilizar la metodología de desarrollo dirigido por pruebas (*Test Driven Development - TDD*) y la herramienta ***RSpec***.

Empezar el desarrollo desde cero, NO RECICLAR CÓDIGO.

La clave de esta práctica está en diseñar pruebas (**expectativas**) que dirijan el desarrollo y si reutiliza otros desarrollos estará haciéndolo mal.

Un ejemplo de las ejecución de las pruebas es el siguiente:

```
Aparcamiento
  Tiene un número de version, usando la sintaxis semántica X.Y.Z
  Funcionalidades
    Se cuenta con una constante para representar si el aparcamiento está completo
    Se cuenta con una constante para representar si el aparcamiento tiene plazas libres
    Se cuenta con una funcion para mostrar el estado de un aparcamiento (completo, plazas libres)
  Aparcamiento::Datos
    Tiene una clase para almacenar los datos del aparcamiento
      Todo aparcamiento tiene el atributo de accesibilidad (1..5)
      Todo aparcamiento tiene el atributo de seguridad (1..10)
      Un aparcamiento tiene un atributo para su identificación
      Un aparcamiento tiene un atributo para su nombre comercial
      Un aparcamiento tiene un atributo para su descripción (Cubierto - Aire libre - Mixto)
      Tiene un atributo para el tipo de aparcamiento (autobuses, bicicletas, coches, motos)
      Tiene un atributo para representar el conjunto de plazas del aparcamiento (altura, longitud, anchura)
      Tiene un atributo para representar el conjunto de plazas libres y ocupadas
      Tiene un método para devolver el número de plazas del aparcamiento
      Tiene un método para devolver el número de plazas libres del aparcamiento
```

6. Documentar la gema utilizando una herramienta de generación automática de documentación (***rdoc*** o ***yard***).

7. Escribir la dirección HTTP del repositorio de la organización 'ULL-ESIT-LPP-2223/gema-XXX' en la tarea habilitada en el campus virtual.

Referencia

Visic, N., Fill, H. G., Buchmann, R. A., Karagiannis, D. (2015, May). A domain-specific language for modeling method definition: From requirements to grammar. In 2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS) (pp. 286-297). IEEE Computer Society.