IIC3745 - Testing (2020 / II)

Actividad 5 - Pruebas de mutación

Fecha de entrega: martes 17 de noviembre, 23:59

Mutant y pruebas de mutación

Las pruebas de mutación son una forma de evaluar la calidad de una batería de *tests* existente. Este tipo de pruebas funciona a través de pequeñas modificaciones en la lógica del código que deberían provocar un fallo en la batería de pruebas. Si esta no falla, entonces existe espacio para mejoras.

Estas pruebas no son tan populares dada la complejidad en su implementación y automatización, sin embargo, algunos lenguajes afortunados cuentan con las librerías necesarias. En el caso particular de *ruby* se puede utilizar la gema *mutant*, sobre la cual pueden encontrar más información y ejemplos de uso en este link.

Para esta actividad se publicará un repositorio base de código junto a una batería de *tests* que cumplen con un 100% de cobertura de líneas según *SimpleCov*. Sobre este código deben utilizar *mutant* para mejorar la calidad de las pruebas.

Repositorio base

En este link pueden encontrar el repositorio base que deben utilizar para la entrega. Consta de un directorio `app/` con toda la lógica de un programa que procesa órdenes de compra de hamburguesas en un restaurante llamado *DCCBurger*. El flujo ideal del programa es:

1. Se genera una orden a partir del tipo de hamburguesa a comprar (*custom* u *original*), la elección personalizada para cada tipo (ingredientes y tipo de pan en el caso *custom* y la hamburguesa en sí en el caso *original*) y la cantidad de dinero que se tiene actualmente.
2. Se detecta si la hamburguesa es válida, esto mediante un subconjunto de criterios para cada tipo de hamburguesa:

    a. *Original*: la elección debe ser una hamburguesa tipo `OriginalBurger` dentro de las disponibles del restaurante, las que se definen en el archivo `app/constants/original_burgers.rb`
    b. *Custom*: la elección debe ser una hamburguesa tipo `CustomBurger` con uno o más ingredientes y un tipo de pan dentro de los disponibles del restaurante. Además, la hamburguesa puede ser *veggie* o *non-veggie* y se hace la validación de que los ingredientes no contradigan esta definición. La información acerca

de tipos de pan e ingredientes se definen en `app/constants/ingredients.rb` y `app/constants/bread_types.rb`

En caso de que alguna de las validaciones falle, el programa levanta una excepción informativa. Las clases utilizadas se encuentran en `app/models/`.

3. Si la hamburguesa es *custom*, esta se valoriza mediante la elección de los ingredientes y el tipo de pan. Una hamburguesa *original* viene con un precio definido.
4. Se verifica que el dinero del usuario sea suficiente para comprar la hamburguesa procesada. Si es suficiente, el programa retorna `true`, si es que no, se levanta una excepción.

Además, se les entrega un archivo `spec/dcc_burger_spec.rb` con todas las pruebas necesarias para cumplir con un 100% de cobertura de líneas según *SimpleCov*.

En concreto, se les pide modificar y/o agregar pruebas y/o código para llegar al 100% de cobertura según *mutant*. Deben lograr esto sin hacer grandes cambios a la lógica del código descrita, es decir, **está prohibido modificar**:

- Mensajes de excepciones o las condiciones para levantarlas.
- Precios de ingredientes, hamburguesas o tipos de pan.
- Restricciones para cada tipo de hamburguesa (`CustomBurger` u `OriginalBurger`).
- Método de valorización o excepciones de la validación de dinero disponible.

Compatibilidad

Dado que *mutant* es una librería que hace modificaciones del código, tiene restricciones de versión y compatibilidad bastante rígidas. Es fundamental seguir las instrucciones descritas en el repositorio y utilizarlo como base para poder llevar a cabo la actividad sin complicaciones y evitar problemas en la corrección. Cualquier duda siempre pueden publicarla en el foro del curso.

Formato de entrega

La actividad puede ser desarrollada en parejas, pero está diseñada para ser completada individualmente. Se abrirá un cuestionario en Canvas con fecha límite martes 17 de noviembre, 23:59hrs.

Tú o tu equipo deberán subir un comprimido *zip* o *rar* con la carpeta `app` que contenga los archivos originales y/o modificados, el archivo `dcc_burger_spec.rb` que contenga las pruebas adicionales y/o actualizadas para lograr el 100% de cobertura para *mutant* y un archivo `integrantes.md` que contenga el nombre, correo y número de alumno/a de los/as integrantes. El comprimido debe respetar la siguiente estructura:

```
.
├── app/
├── spec/
│   └── dcc_burger_spec.rb
└── integrantes.md
```

No cumplir esto resulta en un descuento de **5 décimas**. Para la corrección, se hará uso del repositorio base, por lo que cualquier cambio a los archivos de configuración, versiones y/o nombres de directorios es responsabilidad de cada pareja o alumno/a.

En caso de trabajar en grupos, **solamente un/a integrante debe subir la actividad**. En caso de que ambos/as lo suban, se tomará en cuenta aquella entrega con la que el ayudante se haya encontrado primero y **no habrá opción de solicitar una corrección de otra versión (no se aplica descuento si ambos suben)**. Se aplicará un descuento de **1 décima** a aquellos que no incluyan la información de su pareja en la entrega.