Hito 3

Consideraciones

- La entrega se deberá realizar hasta el **viernes 16 de mayo a las 23:59**.
- La entrega se realizará en el repositorio del proyecto que el grupo utiliza como base en este curso y deberá actualizar toda sección que se vea afectada (ver sección Contenido).
- Por cada hora de atraso se descontarán 5 puntos de la nota final.
- La lista de grupos, números y tutores ya ha sido publicada e informada.

Contenido

El objetivo del presente Hito es diseñar y documentar pruebas unitarias, así como reportar los resultados de su ejecución. Además, como es solicitado para todas las entregas, incorporar nuevas historias de usuario con la respectiva actualización del código en la plataforma.

El detalle de lo requerido se encuentra a continuación:

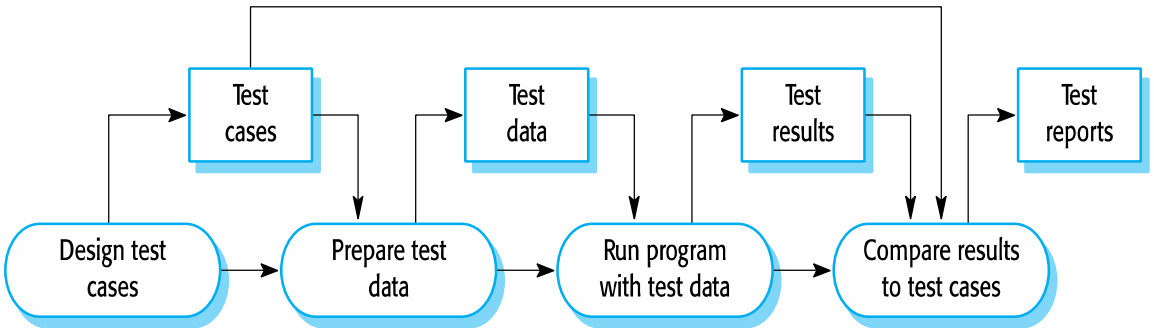
Nueva HU

- En el Hito 2 su grupo identificó 5 HU's adicionales. Escoja una nueva HU adicional de mayor prioridad (justifique) de las 4 HU's no trabajadas, y desarrolle el código correspondiente e incorpórelo a la plataforma.
  - Descomponga la HU, en tareas que conllevan a su implementación.

Pruebas Unitarias

- Para esta entrega su equipo debe diseñar cuatro (4) casos de prueba e implementarlos en Python haciendo uso del framework unittest.
- Las pruebas deben ser diseñadas a nivel unitario, donde la unidad para esta entrega es un endpoint de alguna API.
- Escoja dos endpoints de su sistema que estén funcionalmente operativos, idealmente cada endpoint implementando dos escenarios o historias de usuario distintas.
- Diseñe dos casos de prueba por cada endpoint. Presente por cada caso una tabla con inputs, salida esperada, contexto de ejecución.
- Utilice clases de equivalencia y valores frontera para la definición de estos casos, pero debe variar cada caso en términos funcionales. Es decir, no se contará como dos casos de prueba una variación dentro de la clase de equivalencia.
- Recuerde que los resultados excepcionales ("expected exception") también son verificables. Por ejemplo, usted espera que su endpoint responda con algún mensaje de error frente a inputs inválidos. En este caso, el resultado esperado es el mensaje de error y este también es un caso de prueba válido.

- Implemente con Python y el framework unittest una clase de pruebas por cada endpoint, donde cada prueba en específico se implementa como un método de prueba.
- Cada clase de pruebas debe hacer uso de las clase-métodos setUpClass() y tearDownClass() para setear los datos de prueba, recuerde proceso visto en clases:



- Ejecute las pruebas y capture la pantalla para el resultado de estas ("screenshot").
- Versione el código de las pruebas en una carpeta especial ("tests/") y en esta incorpore además el screenshot de los resultados.
- NO acomode las pruebas para que estas pasen. Si su caso de prueba ha provocado un fallo, ¡felicitaciones! su caso de prueba ha sido exitoso porque ha logrado hacer evidente un defecto en su sistema de software.

  Para más información, revise el material disponible en los siguientes enlaces:

- https://docs.python.org/3/library/unittest.html
- https://requests.readthedocs.io/en/latest/user/quickstart/

Registrar todo el trabajo realizado

- En el repositorio, ya sea a través de un Issue o dentro de una HU (issues) agregando un comentario, para cada trabajo realizado, descríbalo de manera sucinta y el tiempo empleado.

Puede incluir todo material adicional que considere de utilidad.

Dudas o consultas al correo de contacto del tutor de proyecto correspondiente, informado vía AULA.