UNIVERSIDADE DA CORUÑA

Validación y Verificación del Software

Curso 2018/2019

Validación: Análisis de las pruebas

Se analizan los casos de prueba que contiene el proyecto, y se clasifican de acuerdo a los diferentes criterios comentados en clase de teoría: dinámicas/estáticas/simbólicas; caja blanca/negra; positivas/negativas; y funcionales/estructurales/no funcionales.

David Méndez Álvarez (d.mendez.alvarez@udc.es)

Índice

1. EPLiteConnectionTest

Esta clase contiene pruebas del nivel **unidad**.

1.1. domain_with_trailing_slash_when_construction_an_api_path()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

1.2. domain_without_trailing_slash_when_construction_an_api_path()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

1.3. query_string_from_map()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

1.4. url_encoded_query_string_from_map()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

1.5. api_url_need_to_be_absolute()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja blanca, ya que es necesario conocer la estructura interna del código para comprobar un caso que da error.
- Negativa, porque se espera que produzca un fallo.
- Funcional, dado que comprueba el funcionamiento del software.

1.6. handle_valid_response_from_server()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

1.7. handle_invalid_parameter_error_from_server()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Negativa, porque se espera que produzca un fallo.
- Funcional, dado que comprueba el funcionamiento del software.

1.8. handle_internal_error_from_server()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Negativa, porque se espera que produzca un fallo.
- Funcional, dado que comprueba el funcionamiento del software.

1.9. handle_no_such_function_error_from_server()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Negativa, porque se espera que produzca un fallo.
- Funcional, dado que comprueba el funcionamiento del software.

1.10. handle_invalid_key_error_from_server()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Negativa, porque se espera que produzca un fallo.
- Funcional, dado que comprueba el funcionamiento del software.

1.11. unparsable_response_from_the_server()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Negativa, porque se espera que produzca un fallo.
- Funcional, dado que comprueba el funcionamiento del software.

1.12. unexpected_response_from_the_server()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Negativa, porque se espera que produzca un fallo.
- Funcional, dado que comprueba el funcionamiento del software.

1.13. valid_response_with_null_data()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

2. EPLiteClientIntegrationMockTest

Esta clase contiene pruebas del nivel **integración**. Se «mockean» las peticiones, pero esto implica que se simula el comportamiento del servidor, y no de las pruebas; por tanto, siguen siendo dinámicas.

2.1. validate_token()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

2.2. create_and_delete_group()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

2.3. create_group_if_not_exists_for_and_list_all_groups()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

2.4. create_group_pads_and_list_them()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

2.5. create_author()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

2.6. create_author_with_author_mapper()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

2.7. create_and_delete_session()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja blanca, ya que es necesario conocer la estructura interna del código, para saber cómo se calcula la fecha de validez de la sesión.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

2.8. create_pad_set_and_get_content()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

2.9. create_pad_move_and_copy()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

2.10. create_pads_and_list_them()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

2.11. create_pad_and_chat_about_it()

- Dinámica, pues necesita ejecutar el método o los métodos que se van a testear.
- Caja negra, ya que no es necesario conocer la estructura interna del código; sólo que, para un input concreto, devuelva el output esperado.
- Positiva, porque se busca que funcione y devuelva la salida correcta.
- Funcional, dado que comprueba el funcionamiento del software.

3. Análisis y conclusiones

Casi todas las pruebas usan la estrategia de caja negra y no existen pruebas estructurales o no funcionales, lo que implica que hay importantes carencias en el diseño de los tests: no se comprueba la calidad interna de la codificación ni su rendimiento.

No hay pruebas de sistema o de aceptación. Y las de integración se limitan a comprobar si funcionan algunas unidades combinadas; no se verifican valores aleatorios o frontera, a diferencia de las pruebas de nivel unidad. No se comprueban casos que deberían lanzar una excepción o si en caso de no recibir un valor correcto responde como se espera.

La cobertura total es del 94,01 %, si bien la clase EPLiteConnection no alcanza el 85 %, dejando sin recorrer varias líneas relativas a excepciones. Además, debería existir al menos un test para cada función del software que se está probando, y eso no se cumple para listPadsOfAuthor().