Informe Probas Etherpad-Client

Eloy Pazos Lema

1. domain_with_trailing_slash_when_construction_an_api_path:
   Comproba se o EPLite conection genera de forma correcta a URL para conectarse, comprobado a funcionalidade a través de caixa negra , executando todas as liñas do metodo, sendo probas de caixa negra , dinámica , positiva , funcional, unitaria.

2. query_string_from_map: comproba o metodo de engadir argumentos no URL usando un Map, unha proba de caixa negra funcional e unitaria, probando soamente valores válidos sen forzar o erro, deberiase comprobar casos de mandar obxectos nulos no mapa. Son probas de caixa negra , dinámica , positiva , funcional, unitaria.

3. url_encoded_query_string_from_map:comprobase o mesmo método ca no caso anterior da mesma forma esta vez comprobando o caso de enviar caracteres extraños,este e o ultimo testeo deste método, non hai ningun punto en que se consigue provocar unha excepción de encoding, nunca se chega a comprobar que se dea.Son probas de caixa negra , dinámica , positiva , funcional, unitaria.

4. api_url_need_to_be_absolute:comproba o método que pasando o path da API e enviandolle unha query, comproba o caso en que se lle manda unha query nula para provocar unha escepción. Sendo unah proba de caixa negra , dinámica , negativa , funcional, unitaria.

5. handle_valid_response_from_server: comproba o HandleResponse, funcional e unitariamente, no que se lle manda un conxunto de identificadores de Pads e comproba que se devolven correctamente, evaluando o 1º do conxunto devolto en vez de evaluar todos os devoltos, ou o 1º e o ultimo. Núnca se chega a comprobar que lance un EPLite esception. Son probas de caixa negra , dinámica , positiva , funcional, unitaria.
6. handle_invalid_parameter_error_from_server: comproba o mesmo método ca o anterior da mesma forma, esta vez comprobando o caso en que se envie co JSON con un codigo code_invalid_parameters. Sendo probas de caixa blanca , dinámica , negativa , funcional, unitaria.
7. handle_no_such_function_error_from_server: comproba o mesmo ca o anterior , esta vez enviando un code_invalid_method.Sendo probas de caixa blanca , dinámica , negativa , funcional, unitaria.
8. handle_invalid_key_error_from_server: segue evaluando este metodo , enviando un code_invalid_key, comprobando asi todos os casos de codigos enviados por JSON que non sexan CODE_OK e estean controlados, pero non comproba que se envie un numero porriba de 4 e que lance a escepcion asociada a isto.Sendo probas de caixa blanca , dinámica , negativa , funcional, unitaria. (estas probas que executan o mesmo método son de caixa branca porque intentan comprobar o funcionamento do switch case usado na implementacion)
9. unparsable_response_from_the_server: comproba un caso do metodo anterior no que o parseo do JSON non se pode dar e comproba a escepción que devolve. Sendo probas de caixa negra , dinámica , negativa , funcional, unitaria.
10. unexpected_response_from_the_server: segue comprobando o mesmo método, esta vez evaluando o caso de enviar un JSON baleiro e que a escepción devolta é a esperada.Sendo probas de caixa negra , dinámica , negativa , funcional, unitaria.
11. valid_response_with_null_data: comproba o caso de enviar un JSON válido e parseable con corpo baleiro e comprobar que o resultado é o agardado. Sendo probas de caixa negra , dinámica , positiva , funcional, unitaria.

EP LITE INTEGRATION TESTS

Probas de integración dos diferentes métodos, son todas de caixa negra e funcinais.Trata de evaluar o funcionamento normal, en ningún momento forza un error para ver se se esta correctamente controlado

1. validate_token: comproba checkToken, o que busca o token e devolve unha escepción en caso de que sexa valido o token existente, non fai ningunha asercion, non evalua a sua execución senón que o token insertado inicialmente correspondese co que hai na aplicación. Son probas de caixa negra , dinámica , positiva , funcional, de integración
2. create_and_delete_group: trata de crear un grupo, evaluando se na resposta do cliente se devolve un grupo, que non devolva un nulo e que o codigo de grupo que devolva empeza por g. Como esta especificado na API.Construindo unha resposta POST tras crear o grupo, posteriormente borrase o grupo pero non se evalúa que se borra correctamente dito grupo ou a resposta deste.Tampouco se comproba que en caso de que a creación de grupos e correcta (pode ser que devolva sempre o mesmo groupID).Son probas de caixa negra , dinámica , positiva , funcional, de integración
3. create_group_if_not_exists_for_and_list_all_groups: comproba o método de crear un grupo en caso de que non exista, comproba que a resposta devolve un JSON con un “groupID” (esta vez non comproba que o grupo devolto non comeza por g.).
   Posteriormente recuperanse todos os grupos existentes, comprobase que é 1 , despois tratan de crear outro grupo co mismo nombre e evalua se non se crea outro novo, comparando o número de respostas que daba o listAllGroups (non evalua que devolve o mesmo nome de grupo).Son probas de caixa negra , dinámica , positiva , funcional, de integración
4. create_author: trata de crear un autor, trata de recuperar o authoriD e comproba que devolve un authorID non nulo (non comproba que o authorID comece con a.).Crea un autor con nome especificado create_author(nome) e comproba que devolve un autor con nome como o indicado e asegurase que é o mesmo ca o indicado, (non comproba casos en que se mande un string baleiro).Son probas de caixa negra , dinámica , positiva , funcional, de integración
5. create_author_with_author_mapper: crea 2 autores co mesmo authorMapper con nomes diferentes e despois comproba que o identificador devolto é o mesmo. Posteriormente recupera os nomes e comproba que os nomes devoltos son os mesmos. Despois crea un 3 autor co mesmo mapper e sin indicar nome, polo que comproba que devolve o mesmo nome de autor ca o 2 autor. Non comproba nunca o caso de mandar un mapper nulo ou un nome de autor baleiro.Son probas de caixa negra , dinámica , positiva , funcional, de integración
6. create_and_delete_session: crea dúas sesións diferentes e comproba que os identificadores de sesión son diferentes, (selecciona os campos do tempo da sesión de maneira aleatoria, pero en ningún momento proba sesión.Son probas de caixa negra , dinámica , positiva , funcional, de integración
7. create_pad_set_and_get_content: Son probas de caixa negra , dinámica , positiva , funcional, de integración.
8. create_pad_move_and_copy:Son probas de caixa negra , dinámica , positiva , funcional, de integración.
9. create_pads_and_list_them : Son probas de caixa negra , dinámica , positiva , funcional, de integración.
10. create_pad_and_chat_about_it : Son probas de caixa negra , dinámica , positiva , funcional, de integración.

Resumo:

As probas unitarias son maioritariamente de caixa negra coa escepcion nas que proba os valores de erro de JSON,aquí danse probas tanto negativas como positivas deixando algunhas escepcións sen comprobar. Todas son dinámicas e funcionais.

As probas de integración son maioritariamente positivas, soamente busca a execución esperada e aceptable, non forza en case ningún momento provocar erros nin escepcións. Tampouco comproba valores límite. Hai métodos cuxa cobertura soamente se comproba a través dunha execución sen comprobar o resultado. Os test son practicamente todos de caixa negra , seria necesario comprobar os límites das estruturas de datos empregadas nos

diferentes métodos. Tamén hai múltiples escepcións que non son probadas. Todas son dinámicas e funcionais.

A cobertura da execución dos tests daría o redor do 93% pero existen moitos casos en que dita cobertura e so por execución pero non se comproba nada de esa execución (como o getToken onde so se ve que non lanza escepción)