Nombre:
Carlos Ramírez (2018-6055)
Oliver De Jesús Rosario Reyes (2017-5541)

Materia:
Introducción a la Ingenieria de Software

Tema:
Garantía de la Calidad de Software
Practica
Escenarios de Prueba

Profesor(a):
Viela Reyes Bourdierd

La Caleta, Boca Chica, Santo Domingo de Guzmán
14 de marzo del 2020
Cuatrimestre 2020-C-1

Introducción

Partiendo de la investigación previamente realizada sobre la Garantía de la Calidad de Software a continuación se propondrán a grandes rasgos los tests a implementar en el desarrollo de nuestro proyecto para así cumplir con los requisitos planteados por las normas ISO 9000, ISO 9126.

En ese orden se describirán los tests tanto de caja negra como caja blanca, los cuales a su vez engloban tanto los tests unitarios, tests de integración como también los tests funcionales en el caso de la caja negra.

El proyecto viene a resolver las necesidades de un instituto de ingles proponiendo un sistema de gestión tanto de alumnos, calificaciones, empleados, contabilidad, etc. El mismo desarrollado en Python como lenguaje de programacion sobre el Enterprise Resource Planning (ERP) Odoo por sus características. Es por esto que los tests propuestos a continuación estarán íntimamente ligados con la forma en la que funciona el ERP.

Prueba de Caja Blanca.

Las pruebas de caja blanca (también conocidas como pruebas de caja de cristal o pruebas estructurales) se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. El ingeniero de pruebas escoge distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y cerciorarse de que se devuelven los valores de salida adecuados.

En ese orden se proponen dos tipos de tests para este proyecto:

Test unitarios: prueban los componentes del código independientemente. Ej.: probar que el método de validación del archivo funcione correctamente, validar que el método de creación de estados bancarios funcione correctamente. Se hace un rollback de los cambios en la base de datos cada vez que se ejecuta uno de estos test de manera que los cambios hechos por uno no afectan el comportamiento de los demás.

De los cuales se proponen los siguientes casos:

Pruebas al ORM

A los modelos siguientes:

a) Estudiantes
b) Empleados
c) Grupos
d) Niveles
e) Pagos
f) Calificaciones
g) Configuración
h) Usuarios

Se debe probar que:

1. Los métodos básicos del CRUD crean, leen, actualicen y borren correctamente los registros con el id recibido de la tabla en la base de datos asignando los valores correctos a los campos.

2. Cada uno de los métodos mencionados aplique las restricciones de seguridad e integridad correspondientes, así como el manejo de errores y valores nulos como deberían.

3. Los métodos mencionados devuelvan el resultado esperado tanto en el caso de valores correctos como incorrectos.

4. Se gestione correctamente el pool de conexiones con la base de datos de manera que al momento de terminar la ejecución de cada uno de los métodos mencionados se libere esa conexión.

5. Se abra correctamente la conexión con la base de datos al llamar el método get_connection ()

6. El método autenticar retorne false si no se encontró un registro con las credenciales recibidas dentro de la tabla usuarios.

7. El método autenticar retorna el id del usuario si encuentra el registro con las credenciales recibidas dentro de la tabla usuarios.

8. Verificar que el balance pendiente de un pago se compute correctamente al aplicarle un avance del mismo.

Tests de integración: prueba los componentes del código en conjunto. Ej.: probar que el método de lectura del archivo de estados bancarios valide si el archivo es del banco en cuestión y cree los registros correspondientes, todo en conjunto. Se hace un solo rollback de la base de datos al terminar todo, de manera que los cambios realizados por un método afectaran el comportamiento de los demás.

De los cuales se proponen los siguientes casos:

1. El campo promedio de la tabla estudiantes se computa correctamente al modificar un registro de la tabla calificaciones el cual tenga una relación de Many2one con estudiantes.

2. El campo booleano inscrito se computa correctamente al marcar como finalizado el campo de estado del registro de la tabla grupos con el que se tiene una relación Many2many.

3. El campo float balance pendiente se computa correctamente al modificar el campo pagos que tiene una relación One2many con el modelo pagos.

4. El campo booleano inscrito de cada uno de los registros de estudiantes que pertenecen a un grupo se marca como false al presionar el botón cerrar grupo del modelo grupos.

5. El campo total se computa correctamente al modificar cualquier otro de los campos de la tabla calificaciones y se computa además el campo promedio del modelo estudiantes.

6. El campo balance pendiente del modelo estudiantes se computa correctamente al marcar cualquiera de los registros de pagos como pagado.

7. El campo sueldo se computa correctamente al modificar cualquiera de los registros del campo Many2many prestaciones.

8. Test de estrés para asegurar que el sistema se desempeñe correctamente bajo un tráfico pico de usuarios y que el mismo no colapse.

Pruebas de Caja Negra.

En pruebas de software, conociendo una función específica para la que fue diseñado el producto, se pueden diseñar pruebas que demuestren que dicha función está bien realizada. Dichas pruebas son llevadas a cabo sobre la interfaz del software, es decir, de la función, actuando sobre ella como una caja negra, proporcionando unas entradas y estudiando las salidas para ver si concuerdan con las esperadas.

En ese orden se propone el siguiente tipo de tests:

Tests funcionales: prueban el producto ya en funcionamiento e intentan hacerlo de la misma manera en la que lo haría un usuario.

Del cual se proponen los siguientes casos:

1. El estudiante es notificado al momento en que se le genera un pago.

2. Se le denega el acceso al usuario al introducir credenciales incorrectas en el login.

3. El usuario puede ingresar exitosamente al ingresar credenciales correctas en el login.

4. Se le muestra un mensaje de error si el usuario intenta crear una cuenta con un correo que ya existe.

5. Se crea un nuevo usuario correctamente al ingresar un correo que no esta en uso y el usuario es redirigido la página de login.

6. Se muestra un mensaje de error al usuario si la contraseña no coincide con su confirmación en el formulario de sign up.

7. El usuario visualiza un Kanban view de todos los grupos cuyo estado es ‘por iniciar’.

8. Al hacer clic en mis grupos el usuario visualiza solo los grupos a los que ha pertenecido.

9. El usuario no tiene acceso al sistema si tiene pagos pendientes.

10. El usuario tiene acceso al sistema al momento en que todos sus pagos están en etapas de pagado.

11. El proceso de pago de nómina se realiza automáticamente el día del mes especificado en la tabla de configuraciones.

12. Un empleado cuyo estado no es activo no tiene acceso al sistema ni se le computa el pago en la nómina.

13. Los estudiantes reciben una notificación por correo cuando el profesor modifica o agrega alguna calificación a su registro.

14. El campo inscrito de los estudiantes se marca como false si el estudiante acumula 4 faltas registradas.

15. El profesor recibe una notificación por correo una semana antes de cumplirse la fecha límite de entrega de calificaciones si el mismo no ha publicado.

16. El usuario aparece en la lista de estudiantes de un grupo al ser inscrito en el mismo mediante el wizard de inscripción.

17. El estudiante aparece en la lista de espera del grupo que coincide con su disponibilidad de horario al utilizar el wizard de registro.

18. El usuario recibe un mensaje de error al dejar un campo marcado como requerido en blanco.

19. El usuario recibe un mensaje de error si el valor introducido no pasa los parámetros de validación.