Ejercicio Guiado 3
Enunciado

Desarrollo Dirigido por Pruebas

Desarrollo de Software

Marzo 2021

Grado en Ingeniería Informática - Doble Grado en Informática y ADE

ÍNDICE

OBJETIVOS

Este ejercicio guiado tiene tres objetivos principales:

- Practicar el proceso de desarrollo guiado por pruebas.
- Conocer las herramientas necesarias para el desarrollo basado en pruebas.
- Practicar las técnicas de pruebas funcionales y estructurales.

EXPOSICIÓN DE CASO

Los objetivos específicos de este ejercicio son definir, automatizar y ejecutar las pruebas unitarias requeridas que se utilizarán para verificar si el componente para la gestión de accesos funciona correctamente una vez que se complete su desarrollo.

Tienes una versión inicial del código fuente disponible en Aula Global. Ten en cuenta que este componente, en este momento, no implementa ninguna funcionalidad. El código se proporciona para que la implementación de los casos de prueba se pueda ejecutar utilizando la herramienta de automatización **PyBuilder**. Sin embargo, se espera que ninguno de los casos de prueba funcione correctamente ya que aún no se ha realizado el diseño y la implementación de las funcionalidades del componente.

Los estudiantes deben desarrollar las funciones propuestas basándose en el código fuente proporcionado. Deben aplicar las técnicas TDD junto con técnicas de prueba funcionales y estructurales.

Este componente debe ser desarrollado en Python 3.8, usando el IDE PyCharm y configurando un entorno virtual para el proyecto. El código debe compartirse en un repositorio privado en GitHub.

Requisitos Funcionales

Los requisitos funcionales para los que se desea definir los casos de prueba funcionales y estructurales son los siguientes:

Función 1. Solicitar el código de acceso.

AM-FR-01: Quienes quieran acceder al edificio deben solicitarlo antes. El sistema recibirá información personal y devolverá un código que será necesario para obtener la clave de acceso.

Entrada:

AM-FR-01-I1: id_card. El primer parámetro es una tarjeta de identificación española (DNI) válida. Debe ser una cadena de 9 caracteres. Los primeros ocho caracteres deben ser un número y el último carácter una letra según las reglas establecidas por el gobierno español.

AM-FR-01-I2: access_type. Tipo de acceso solicitado. Este parámetro puede tomar dos valores diferentes: “Resident” o “Guest”.

AM-FR-01-I3: name_surname. Nombre y apellidos de la persona que quiere acceder al edificio. Se espera al menos un nombre y un apellido separados por un espacio en blanco.

AM-FR-01-I4: email_address. Dirección de email válida.

AM-FR-01-I5: validity. Número de días con acceso permitido desde la fecha de emisión de la clave. En caso de visitantes, este número debe estar entre 2 y 15 días. En caso de residentes, este valor debe ser 0 (el acceso siempre está permitido).

Proceso:

AM-FR-01-P1: El componente debe verificar la corrección de los datos recibidos.

AM-FR-01-P2: Si los datos recibidos son correctos, el componente obtendrá una firma mediante el algoritmo MD5. Este valor MD5 se obtiene del método __str__ de la clase AccessRequest (clase disponible en el código de Aula Global). Esta firma será el código de acceso para obtener una llave en el edificio.

AM-FR-01-P3: El sistema debe almacenar en un fichero todos los datos de cada solicitud para poder comprobarlo posteriormente.

Salidas:

- AM-FR-01-O1: Una cadena en MD5 correspondiente al código de acceso generado en formato hexadecimal..
- AM-FR-01-O2: Excepción en los siguientes casos:
    - El DNI recibido no es válido o no tiene un formato válido.
    - El tipo de acceso solicitado no es válido.
    - La cadena de nombre y apellido no es válida.
    - El formato de correo electrónico no es válido.
    - El número de días no tiene un valor válido.
    - Otros errores internos.

Interfaz

La definición del componente de interfaz que proporciona esta funcionalidad es la siguiente:

```
request_access_code (id_document,
                     access_type,
                     full_name,
                     email_address,
                     days):

// Devuelve una cadena que representa AM-RF-01-O1

// En caso de errores, devuelve una AccessManagerException representa AM-RF-01-O2
```

Función 2. Obtener acceso al edificio.

AM-RF-02: Cuando una persona quiere acceder al edificio, el sistema debe verificar la corrección del código de acceso generado por la función 1. Si el código de acceso es válido, el sistema devolverá una clave. Esta clave se utilizará para abrir las puertas cerradas del edificio.

Entradas:

AM-RF-02-I1: La ruta del archivo del archivo con los datos de acceso para los que es necesario generar una nueva clave. El archivo de entrada JSON debe cumplir con el siguiente formato:

```
{
    "AccessCode":"<String having 32 hexadecimal characters>",
    "DNI":"<valid DNI>",
    "NotificationMail":[“<Valid email address>”*]
}
*This field can contain from 1 to 5 emails separated by comma
```

Proceso

AM-RF-02-P1: el componente debe verificar que la solicitud de acceso fue almacenada en el almacén de solicitudes y que el código de acceso coincide con los datos (es decir que no se han manipulado los datos del fichero).

AM-RF-02-P2: Si el acceso solicitado es válido, el componente debe generar una instancia de la clase AccessKey. Los atributos de esa clase son:

- "alg" (String). Identifica el algoritmo utilizado para firmar la clave de acceso. Por ahora, su valor debe ser "SHA-256", pero en futuras versiones, el componente permitirá otros valores.
- "typ" (String). Por ahora, el componente solo permite el valor "DS" para este campo.
- "access_code" (String). Este valor es el valor del Código de Acceso obtenido en AM-RF-01.

- “issued_at". Contiene la fecha de emisión de la clave (hora UTC en formato de marca de tiempo).
- "expiration_date". Fecha de vencimiento de la clave (hora UTC en formato de marca de tiempo). Para los visitantes, es la suma del “issued_at” y el número de días que se especificaron en la solicitud del código de acceso (ver función 1). En caso de residentes, la fecha de caducidad es 0.
- “key” (String). Este campo contiene la llave para abrir las puertas del edificio. La clave se calcula codificando los campos anteriores usando “SHA-256”. El texto a codificar tiene el formato:

```
{alg:<value>,typ:<value>,accesscode:<value>,issuedate:<value>,
expirationdate:<value>}
```

AM-RF-02-P2: El componente debe almacenar todas las claves generadas en un fichero para verificar los intentos de acceso.

Salidas

AM-RF-02-O1: una cadena SHA-256 con la clave en formato hexadecimal, si la solicitud de acceso es correcta (se ha encontrado en las solicitudes almacenadas y los datos están bien).

AM-RF-02-O2: un AccessManagementException en los siguientes casos:

- No se encuentra el archivo de datos.
- El archivo no tiene formato JSON.
- El JSON no tiene la estructura esperada.
- Los datos del JSON no tienen valores válidos.
- La solicitud no se encontró en el archivo de solicitudes.
- Error de procesamiento interno al obtener la clave.

Interfaz

La definición del componente de interfaz que proporciona esta funcionalidad es la siguiente:

```
get_access_key (input_file):

// El archivo de entrada es una cadena con la ruta del archivo descrita en AM-RF-02-I1

// Devuelve un String en hexadecimal que representa la clave AM-RF-02-O1

// En caso de error, devuelve un AccessManagementException representa AM-RF-02-O2
```

Función 3. Apertura de puertas

AM-FR-03: Quienes quieran abrir una puerta deben introducir el valor de una clave válida. El sistema recibirá el valor de la clave y devolverá verdadero si la clave es válida o una excepción en caso contrario.

Entradas:

AM-RF-03-I1: Una cadena SHA256 que representa una clave generada por la función 2 en formato hexadecimal.

Proceso:

AM-FR-03-P1: El componente debe verificar la exactitud de la clave recibida. Con base en la experiencia de los métodos anteriores, debe determinar si la clave recibida es válida o no.

AM-FR-03-P2: Si la clave era válida, el componente debería registrar en un archivo la marca de tiempo (hora UTC) del acceso y el valor de la clave.

Salidas:

AM-FR-03-O1: El componente devolverá verdadero si la clave es válida.

AM-FR-03-O2: una AccessManagementException en los siguientes casos:

- La cadena de entrada no contiene una clave que pueda procesarse.
- La clave a verificar no está registrada.
- El valor de la clave no es válido (valores diferentes para cada situación)
- Error de procesamiento interno al procesar la clave (es decir, archivos no encontrados, etc.)

Interfaz

La definición del componente de interfaz que proporciona esta funcionalidad es la siguiente:

```
open_door (key):

// la clave es una cadena con el valor descrito en AM-RF-03-I1

// Devuelve un valor booleano definido en AM-RF-03-O1

// En caso de errores, devuelve una AccessManagementException que representa AM-RF -03-O2
```

TAREAS A REALIZAR

1. Cree un nuevo repositorio en GitHub para este ejercicio guiado. Nombra este repositorio de acuerdo con las reglas descritas en el ejercicio guiado 2 (GXX.TYY.EG3) e invita a tu profesor de prácticas.
2. Clona el proyecto e incluye el código y la estructura de carpetas disponibles en Aula Global.
3. Siguiendo el proceso TDD, define los casos de prueba e implementa la **primera función**. Aplicar en este primer método el Análisis de Clases de Equivalencia y los Valores de Límite (cuando corresponda). Dependiendo de la naturaleza de los parámetros de entrada y salida, hay varias reglas a considerar. Estas reglas están relacionadas con:
    - Rango de valores de entrada / salida
    - Cantidad de valores de entrada
    - Semántica de los valores de entrada
4. Después, define los casos de prueba e implementa la **segunda función**. En este caso, aplica la técnica de **Análisis sintáctico.** Si es necesario, **complementar** las pruebas **aplicando clases de equivalencia y valores límite**.
5. Finalmente define los casos de prueba e implementa la tercera función. Aplicar en este tercer caso técnicas de **prueba estructural**.
6. **El proyecto debe seguir el estilo de codificación PEP8**. Debe verificar el código **usando pylint con la configuración predeterminada**. Todos los errores y advertencias detectados por pylint deben resolverse.
7. Ambos miembros del equipo deben interactuar con el repositorio de GitHub, siguiendo los principios de programación por pares y propiedad colectiva del código.

8. Cree un documento PDF e incluye la definición de los casos de prueba para cada función. Para la segunda función, incluye también la gramática y el árbol de derivación. Para la tercera función se incluye el Gráfico de control de flujo, la definición de las rutas básicas y los casos adicionales necesarios para probar los bucles. La buena presentación del documento se tendrá en cuenta en la nota (portada, encabezados, formato del documento, etc.)

   Incluir también un fichero excel con la especificación de los casos de prueba descrita en el documento anterior (plantilla disponible en Aula Global).

   Estos documentos deben incluirse en la carpeta “docs”.

9. Generar un informe con los resultados de la ejecución de las pruebas para cada función. Incluir estos informes en la carpeta "docs\test_reports".

Los anexos del ejercicio guiado describen las técnicas involucradas en este ejercicio guiado.

NORMAS Y PROCEDIMIENTOS

Este ejercicio se resolverá por parejas. Una vez finalizado el ejercicio, debes subirlo en el repositorio correspondiente de GitHub, incluyendo:

- El código fuente generado para cada funcionalidad durante el proceso TDD.
- El código de las pruebas definidas para automatizar la validación de los casos de prueba seleccionados.
- Los informes con los resultados de la ejecución de todas las pruebas, en una carpeta llamada “docs\test_reports”.
- Un documento PDF que contiene:
    - Las clases de equivalencia y el análisis de valores límite para la

funcionalidad # 1.

- La gramática, el árbol de derivación y la selección de casos de prueba a desarrollar para la funcionalidad # 2 utilizando la técnica de análisis de sintaxis.
- El diagrama de flujo de control, las rutas básicas y los casos adicionales para probar los bucles para la funcionalidad # 3 usando la técnica de análisis estructural.
- Este documento debe guardarse en el directorio “doc” del proyecto.

- Un documento excel con la definición de los casos de prueba, de acuerdo con la plantilla disponible en aula global. Este documento debe guardarse en el directorio doc del proyecto.

La fecha límite para este ejercicio es el 16 de abril de 2021 antes de las 23:59.

De acuerdo con los estándares de evaluación continua establecidos en esta asignatura, si un equipo no publica la solución del ejercicio antes de la fecha límite, el ejercicio será evaluado con una puntuación de 0 puntos.