EJERCICIO GUIADO 2

**ÍNDICE**

César Manuel Blázquez Martín 100495797

Hugo Becerra Fernández 100495717

Iñaki Feijoó Basagoiti 100495947

Contenido

[Función 1 2](#_Toc162807902)

[Función 2 3](#_Toc162807903)

[Función 3 3](#_Toc162807904)

# Función 1

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Clases de equivalencia y valores límite en los distintos parámetros de la función roomReservation:

CREDIT\_CARD:

Como clases de equivalencia válidas, tenemos 3. La primera se cumple cuando el algoritmo de Luhn es válido. La segunda es cuando el tipo de dato es válido, es decir, son números. Por último, la longitud del número de la tarjeta. En cuanto a las clases de equivalencia no válidas, tenemos 4. Uno de los 16 dígitos no es válido, la segunda es cuando no es un número. La tercera describe el caso de que sean más de 16 dígitos y por último que sean menos de 16. En los valores límite tenemos como válido el caso de que tenga 16 dígitos y como no válidos 15 y 17.

ID\_CARD:

Las clases de equivalencia válidas de Id\_card, son las mismas son las mismas que las de credit\_card cambiando la primera, ya que el D.N.I tiene 8 dígitos numéricos y una letra final. Las clases de equivalencia no válidas son aquellas que contemplan los casos de que no haya una letra final, no sea un número o tenga mas de 8 dígitos o menos. En cuanto a los valores límite, válido es el caso de 8 números y 1 letra final, y los no válidos, aquellos que tengan 7 o 9 dígitos numéricos y 0 o 2 letras.

NAME\_SURNAME:

En este caso las clases de equivalencia válidas contemplan al menos 2 cadenas de caracteres, que el tipo de dato sea un string y que tenga entre 10 y 50 caracteres. Las no válidas recogen los casos de que no sean caracteres, este fuera de rango o no se proporcionen dos cadenas. Los valores límite son el nombre mas pequeño, 10 caracteres, y los no válidos son 9 y 51 caracteres.

PHONE\_NUMBER:

La única clase de equivalencia válida del teléfono es que tenga 9 dígitos. En el caso de las no válidas es que no sean números o no sean exactamente 9 dígitos. Los valores límite no válidos contemplan que sean 8 o 10 dígitos.

ROOM\_TYPE:

La clase de equivalencia válida es que sea una palabra dentro de estas tres posibles; single, double o suite. La clase de equivalencia no válida es aquella en la que el tipo de habitación no es alguno de esos tres.

ARRIVAL:

Las clases de equivalencia de la fecha de llegada de un cliente manejan que el formato de la fecha sea el correcto; DD/MM/YYYY. En los valores límite se multiplica el número de casos, ya que contemplamos mucha variedad de situaciones que se pueden dar, que el día sea entre 1 y 31, o 1 y 30 los meses que no tengan 31 días, o 28 en el caso de febrero. Así también como que los meses no pueden ser de 13 hacia arriba o el año de llegada sea anterior al actual.

NUM\_DAYS:

El número de días que un cliente puede quedarse en el hotel es entre 1 y 10, ahí está su clase de equivalencia válida, y por consiguiente la no válida, que es que se salga de ese rango. Un valor límite válido es un número de días entre 1 y 10, y los dos tipos de valores límite no válidos es que tengamos 0 días u 11.

# Función 2

Diagrama, Esquemático

Descripción generada automáticamente

En base a la gramática definida anteriormente, se han definido una serie de casos de prueba, concretamente 21, en los cuales se comprueba la integridad del árbol. El primero es un caso de prueba que cubre todos los nodos, y por tanto es válido (se especifica en el Excel). A partir del segundo se comienzan a contemplar errores en distintos módulos o distintos nodos, los cuales se especifican también en el Excel.  
  
Centrándonos en el código, se han seguido dos estructuras:

* Para el caso válido, se hace un assertEqual básico donde compara el localizador que se ha creado en el test con el almacenado en el .json
* Para el resto de casos, al definir los tipos de errores, se comprueba con un assertEqual la excepción que genera el test con el esperado.

# Función 3

Diagrama

Descripción generada automáticamente

En cuanto al diagrama de flujo hemos creado 12 nodos. El primero sirve para comprobar la existencia de room\_key, en caso negativo lanzamos una excepción y en caso positivo pasamos a comprobar la existencia del archivo json. Si no existe este archivo, se lanza su correspondiente excepción. Si existe, comprobamos la validez de su formato y en caso de que no sea el correcto lanzamos dos excepciones, una que avisa de que el formato no es el correcto y otra de error de procesamiento. Si el archivo está en el formato correcto habrá que ver si la room\_key proporcionada se encuentra en dicho archivo, si no se encuentra se lanza una excepción y si se encuentra pues ya pasaríamos a comprobar la validez de la fecha y en caso afirmativo devolvemos True. En caso negativo salta una excepción.

Para los casos de prueba del código solo pudimos implementar 3, uno válido y dos inválidos sobre si no se introducía room\_key o si la room\_key no estaba registrada dado a que a esta función solo le entra como argumento room\_key.