|  |  |
| --- | --- |
| Descripción: Descripción: Descripción: https://moodle.lab.dit.upm.es/moodle/pluginfile.php/4552/mod_resource/content/1/ditupm.gif | **Departamento de Ingeniería de Sistemas Telemáticos**  **Ingeniería del Sistemas y Servicios Telemáticos** |

CASOS DE PRUEBAS

TC (Test Cases)

|  |  |
| --- | --- |
| Nombre del documento | Plan de Pruebas |
| Nombre del fichero: | *ISST-TC-V12.docx* |
| Versión: | *1* |
| Fecha de entrega: | *12/05/2025* |
| Caso de estudio: | *Splitit* |
| Entrega | *Sprint 3* |

|  |  |  |
| --- | --- | --- |
| Autores/as: | |  | | --- | | *Pablo Bas Iglesias, Rodrigo de la Nuez Moreno* | |
| Revisor/a: | |  | | --- | | *Javier de Ponte Hernando* | |

Resumen

|  |
| --- |
| Plantilla para el documento que describe los casos de prueba  *Su índice de contenidos es el que debe incluirse en el Documento de Casos de Prueba que el Equipo de Trabajo deber realizar y entregar, como parte de su trabajo en grupo para la asignatura Ingeniería de Sistemas y Servicios Telemáticos* |

Comentarios

|  |
| --- |
| Las secciones de este documento se han rellenado con comentarios, escritos en letra cursiva, que pueden servir de guía para la elaboración del documento real. Naturalmente en la versión redactada por los alumnos estos comentarios deben desaparecer y el texto que lo sustituya deberá estar escrito con letra normal, no cursiva. |

INDICE

1 INTRODUCCIÓN 2

1.1 Propósito del documento 2

1.2 Definiciones, acrónimos y abreviaturas 2

1.3 Materiales de referencia 2

1.4 Estructura del documento 2

2 PLAN DE PRUEBAS 2

2.1 Visión global de las pruebas 2

2.2 Recursos, personal y productos de las pruebas 2

2.3 Control y seguimiento de las pruebas 2

2.4 Métricas, herramientas y entorno de pruebas 2

2.5 Calendario de pruebas 2

3 Especificación casos DE PRUEBA 2

3.1 Caso de prueba 2

3.1.1 Descripción 2

3.1.2 Recursos 3

3.1.3 Precondiciones 3

3.1.4 Postcondiciones 3

3.1.5 Flujo de eventos 3

3.1.6 Resultado 3

# INTRODUCCIÓN

## Propósito del documento

Este documento recoge los casos de prueba definidos y ejecutados sobre el backend de la aplicación Split.it. El objetivo es validar la lógica de negocio relacionada con la gestión de gastos, deudas y grupos.

## Definiciones, acrónimos y abreviaturas

 DTO: Data Transfer Object

 JUnit: Framework de pruebas unitarias

 Mockito: Librería de mocks para simular dependencias

## Materiales de referencia

 epositorio GitHub: [jav-io/ISST-Grupo12-Split](https://github.com/jav-io/ISST-Grupo12-Split)

 Código fuente del proyecto (rama main)

 Documentos ISST entregados por el grupo

## Estructura del documento

El documento está dividido en dos grandes bloques:

* **Bloque I – Plan general de pruebas:**  
  En este apartado se detallan los objetivos generales de las pruebas, los recursos utilizados, las herramientas de testeo empleadas (JUnit, Mockito), el equipo encargado de su diseño y ejecución, También se explican las métricas empleadas para determinar si una prueba ha sido superada y el entorno técnico sobre el que se han ejecutado los tests.
* **Bloque II – Especificación de casos de prueba:**  
  Se incluyen los casos de prueba definidos para los servicios críticos de la aplicación. Para cada uno se presenta una ficha individual con:
  + Descripción del objetivo de la prueba
  + Dependencias o recursos usados
  + Precondiciones necesarias antes de ejecutar la prueba
  + Datos de entrada y salidas esperadas
  + Flujo de ejecución paso a paso
  + Resultado observado (superado o no)

El documento está diseñado para permitir la trazabilidad de cada caso probado con el código fuente, facilitando la revisión por parte del equipo docente o cualquier otro miembro del equipo de desarrollo

# PLAN DE PRUEBAS

Este apartado describe los planes definidos para la realización de las pruebas, dejando para el siguiente apartado la descripción de los casos de pruebas y los resultados de su aplicación.

## Visión global de las pruebas

Se han realizado pruebas unitarias sobre servicios con lógica de negocio crítica: gestión de gastos, creación de grupos y búsqueda de deudas. Se han cubierto tanto casos positivos como errores esperados.

## Recursos, personal y productos de las pruebas

 IDE: Visual Studio Code

 Herramientas: Maven, JUnit 5, Mockito

 Repositorio local y remoto GitHub

## Control y seguimiento de las pruebas

Los resultados de las pruebas se han controlado mediante la salida en terminal de mvn test y el uso de aserciones (assertThrows, assertEquals).

## Métricas, herramientas y entorno de pruebas

 Herramientas: Maven, JUnit, Mockito

 Pruebas ejecutadas con mvn test

 Métrica principal: cumplimiento esperado (pasa / no pasa)

## Calendario de pruebas

* Sprint 3: creación de test unitarios básicos yampliación a test con mocks y verificación de lógica de negocio

.

# Especificación casos DE PRUEBA

**3.1 Caso: testCrearGastoSinGrupoId\_lanzaExcepcion**

**Descripción**

Verifica que no se puede crear un gasto si no se proporciona el ID del grupo.

**Recursos**

JUnit, Mockito, clase GastoServiceTest

**Precondiciones**

Sistema en estado inicial, sin necesidad de base de datos real.

**Datos de entrada**

GastoDTO con grupoId = null

**Postcondiciones**

Se lanza excepción de tipo RuntimeException.

**Flujo**

1. Crear GastoDTO sin grupoId
2. Ejecutar crearGasto()
3. Verificar que se lanza la excepción

**Resultado**

Prueba superada

## Caso: testBuscarPorId\_DeudaNoExiste\_LanzaExcepcion

**Descripción**

Se espera una excepción cuando se busca una deuda que no existe.

**Recursos**

Clase DeudaServiceTest, Mockito

**Precondiciones**

Deuda con ID 99L no está presente en el mock

**Datos de entrada**

ID inexistente

**Postcondiciones**

Se lanza una excepción

**Flujo**

1. Simular repo con deuda ausente
2. Llamar a buscarPorId()
3. Capturar excepción

**Resultado**

Prueba superada

## Caso: testCrearGrupoCorrectamente\_funciona

**Descripción**

Valida que se puede crear un grupo correctamente con nombre y usuario válidos.

**Recursos**

Clase GrupoServiceTest, mocks

**Precondiciones**

El usuario con ID 1L existe en el mock

**Datos de entrada**

GrupoDTO con nombre y idCreador

**Postcondiciones**

Grupo se guarda correctamente y se registra en el sistema

**Flujo**

1. Simular usuario válido
2. Crear DTO con datos válidos
3. Ejecutar método y verificar que se guarda

**Resultado**

Prueba superada