Trabajo Práctico

Pruebas de Aceptación Automatizadas

Ingeniería de Calidad UNTREF

Dentro del proyecto Repartir se debe agregar una regla y dos o más escenarios que la ilustren, y luego implementar la regla en el código del proyecto (<https://github.com/dfontde/repartir-2025>).

La regla **a agregar es libre**, aunque no debe estar ya implementada en el proyecto. Un ejemplo que pueden utilizar es “El total de un grupo no debe ser negativo”.

**Condiciones de entrega**

* Hacer un fork del proyecto en GitHub.
* Los escenarios deben cumplir con las características BRIEF
  + Orientado al **negocio**/dominion (B de Business)
  + Datos **reales** cuando sea pertinentes (R de Real data)
  + Revelar **intención** (I de Intention revealing)
  + **Esencial**, sin nada que no haga falta para ilustrar el comportamiento
  + **Enfocado** ilustran una única regla (F de focused)
* Se debe escribir primero la regla, luego los escenarios, luego implementar las pruebas de aceptación automatizadas y finalmente la solución, siguiendo el ciclo de BDD.
* Las pruebas de aceptación automatizadas pueden ser del tipo *acceptanceTest* (punta a punta con Selenium) o fastA*cceptanceTest* (sobre los objetos de dominio). Se recomienda este último tipo de prueba para quienes hayan tenido problemas de entorno (por ejemplo, Docker o WebDriver).
* Se debe entregar link al repositorio público de GitHub con el código del proyecto pasando todas las pruebas (./gradlew check), e incluyendo la nueva funcionalidad implementada (la excepción la constituyen pruebas que no puedan correr por problemas de entorno como se describe en el ítem anterior.

Se recomienda probar manualmente la solución cuando las pruebas automatizadas estén pasando.

Regla propuesta:

El total de un grupo no debe superar un límite máximo permitido (por ejemplo, $10.000,00).

* **Escenario 1:** Intenta agregar un gasto que haría que el total supere el límite → debe ser rechazado.
* **Escenario 2:** Agrega un gasto que no supera el límite → debe ser aceptado.

Ambos escenarios son:

* **Business-oriented:** Reflejan una restricción de negocio clara.
* **Real data:** Usan montos concretos y nombres reales.
* **Intention revealing:** Se entiende claramente la intención de cada paso.
* **Essential:** Solo lo necesario para ilustrar la regla.
* **Focused:** Ambos ilustran la misma regla.

Las pruebas de aceptación automatizadas son del tipo fastA*cceptanceTest.*

Modificaciones al codigo:

**LimitesDeGrupo.java(ar.com.grupoesfera.repartir.config):**

Cree una clase para centralizar el limite del grupo en una clase de constantes (asi evitamos cambiarlo manualmente en todos los archivos que se utilice). La constante se llama “LIMITE\_TOTAL”

**MontosService.java (ar.com.grupoesfera.repartir.services):**

Utilizo la constante de “LIMITE\_TOTAL” para verificar que el máximo presupuesto del grupo no fue superado, si lo fue lanza una excepción (**LimiteDeGastoExcedidoException)**

**LimiteDeGastoExcedidoException.java (ar.com.grupoesfera.repartir.exceptions):**

Excepción creada que indica si se excede del máximo presupuesto un grupo al agregar un gasto (formateado el mensaje de la excepción con separador de miles).

**Limite\_grupo.feature(resources.feature.grupos):**

Feature de Cucumber creada. Contiene la regla y los escenarios propuestos.

**TotalGrupoNoSuperaLimiteSteps.java(ar.com.grupoesfera.repartir.steps.grupos**

**):**

Archivo Java creado con los steps referidos al nuevo feature creado (limite\_grupo.feature)