Universidad ORT Uruguay

Facultad de Ingeniería

Bernard Wand Polak

Diseño de Aplicaciones 1

Obligatorio 1

Martín Robatto **Nro Est. 240935**

Santiago Ferreiro **Nro Est. 231691**

Grupo: M4AN

Docente:

- Ricardo Samuel Szyfer Sabath
- Bruno Ricardo Balduccio Méndez

Estudiantes

Nro. Estudiante	240935
Nombre:	Martín
Apellido/s:	Robatto Otegui
Grupo / Turno:	M4A

Nro. Estudiante	231691
Nombre:	Santiago
Apellido:	Ferreiro Raina
Grupo / Turno:	M4A

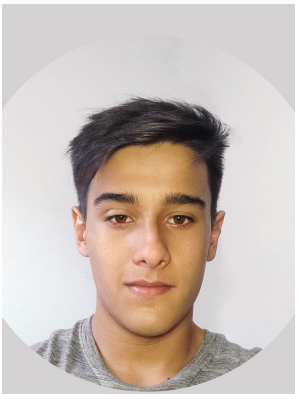


Tabla de contenido

Importante:

Se adjuntará un archivo “Diagramas” conteniendo todos los diagramas para una mejor visualización en caso de que se necesite, entendemos que tal vez en este archivo se dificulte su lectura debido a las limitaciones del documento.
También adjuntamos otro archivo llamado “Backups” el cual contendrá los scripts de los datos de prueba y vacíos, con sus respectivos backups para lograr levantar la base de datos.

Link al repositorio de GitHub: https://github.com/ORT-DA1/240935-235695.git

Descripción general del trabajo:

En ésta oportunidad, el objetivo del sistema es brindar apoyo a aquellas personas que necesitan de una herramienta la cual se encargue de facilitar una buena planificación financiera. Esto es sumamente recomendado por los expertos en finanzas, quienes argumentan que con un sistema como este, se logran identificar patrones de consumo, reconocer categorías en las que se gasta más y por ende establecer objetivos de ahorro de acuerdo a toda la información recabada.

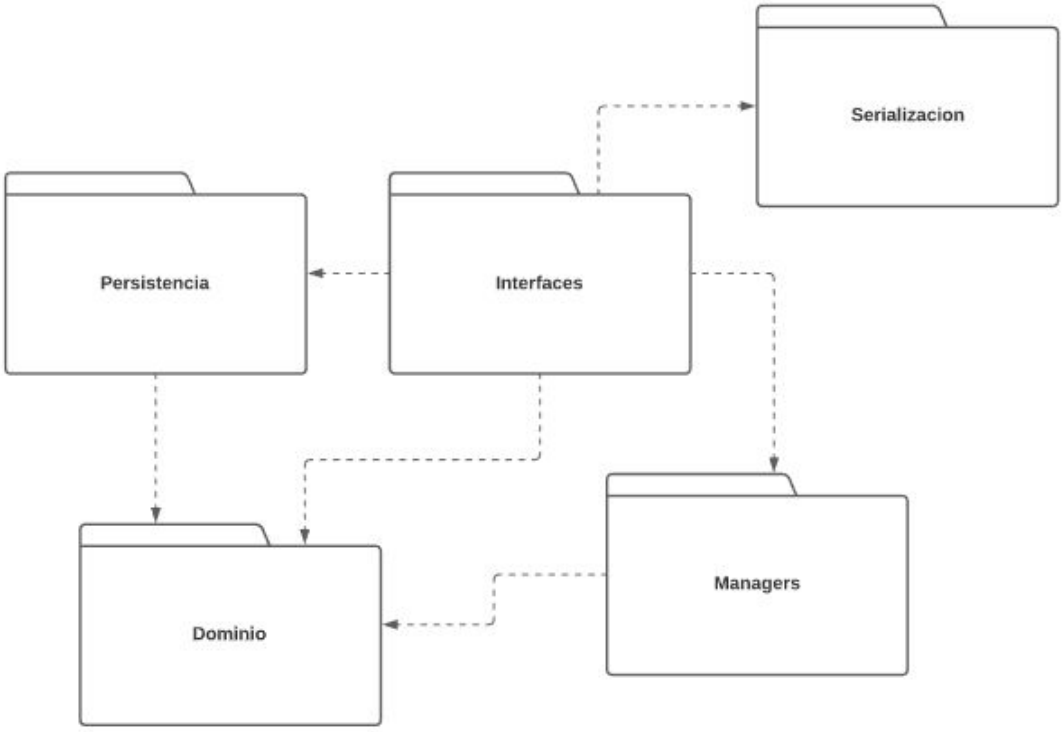
Las principales características requeridas fueron:

- La aplicación debe mantener un registro de categorías de gastos.
- Se debe permitir agregar transacciones individuales con una categoría y monto.
- Se debe poder establecer un presupuesto para cada mes, en el que cada categoría tiene un monto estimado de gasto.
- Se debe mostrar un reporte por mes de todos los gastos realizados.
- Se debe ver un reporte por mes, en el que se muestra para cada categoría el monto
- estimado y el real.
- Persistencia de los datos: En esta nueva versión, la empresa requiere que todos los datos del sistema sean persistidos en una base de datos.
- Soporte Multimoneda: A partir de ahora la aplicación deberá soportar el ingreso de gastos en monedas diferentes.
- Exportar Reporte de Gastos: La aplicación deberá permitir exportar la lista de gastos que se presenta en el Reporte de Gastos a diferentes tipos de archivos.

Descripción general del sistema:

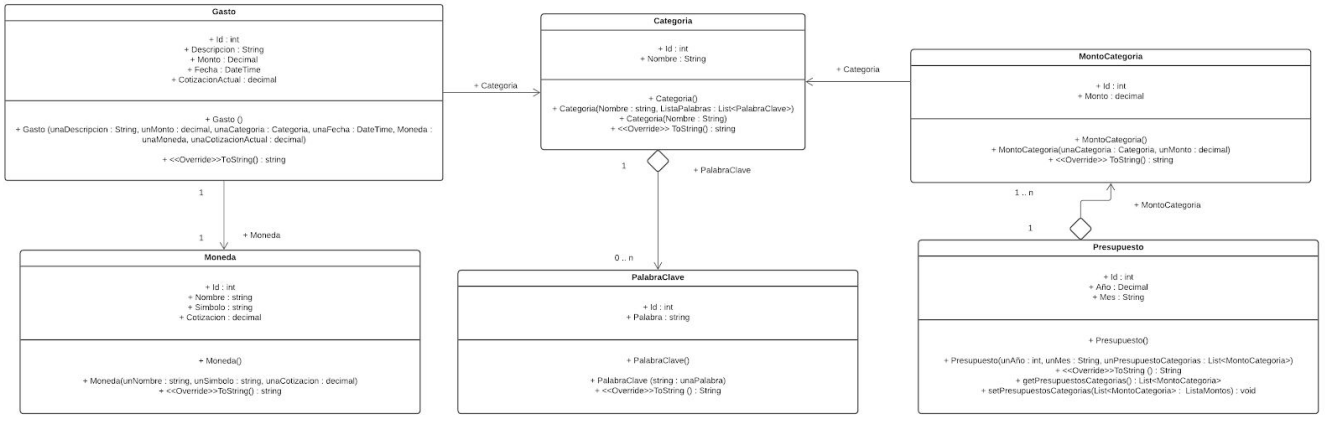
El sistema presenta un menú inicial en el cual brinda la posibilidad de acceder a registro, modificación de una categoría, registro, modificación y eliminación de un gasto y registro o modificación de un presupuesto. También podemos ingresar a dos secciones más, donde se encuentran los reportes de gastos y el reporte de presupuestos.
Se utilizó la metodología de TDD (Test Driven Development) como medio de diseño del sistema.

Diagrama de paquetes:

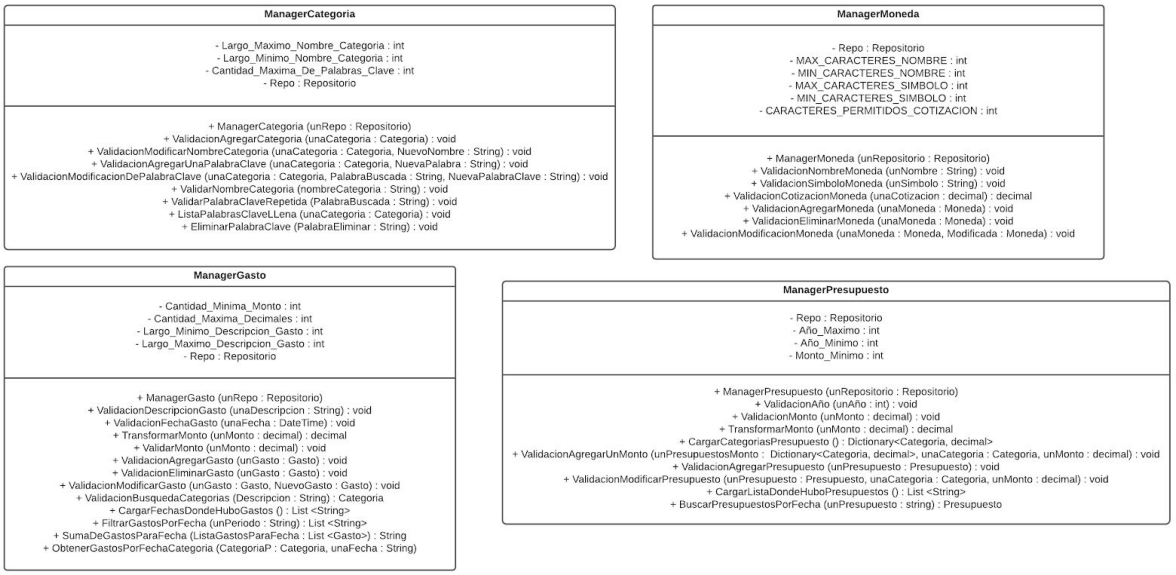


UML:

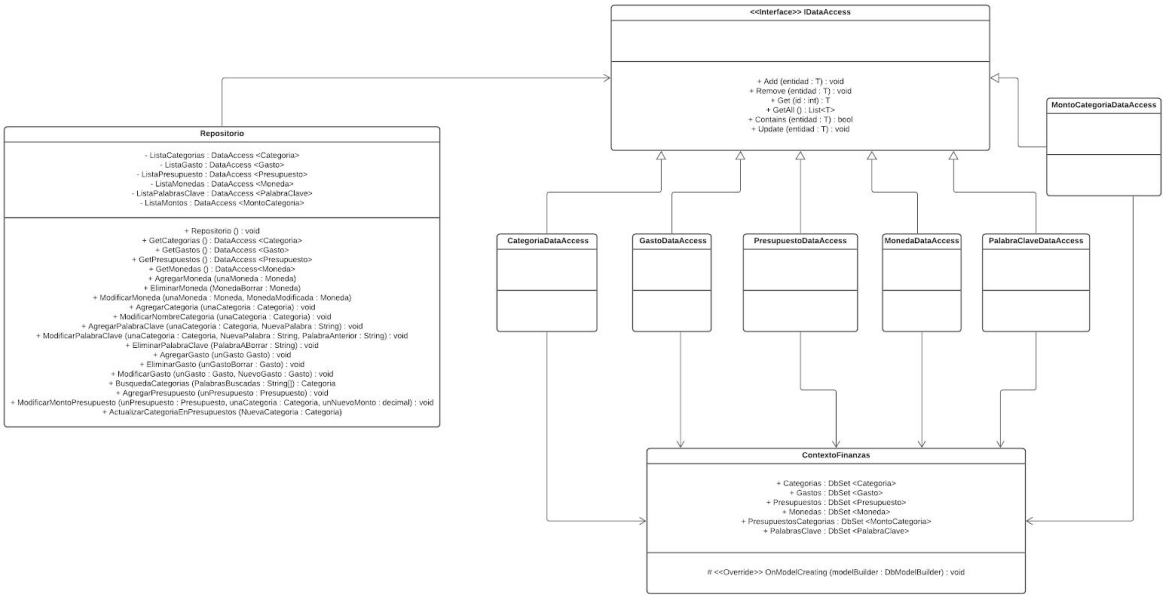
Dominio:



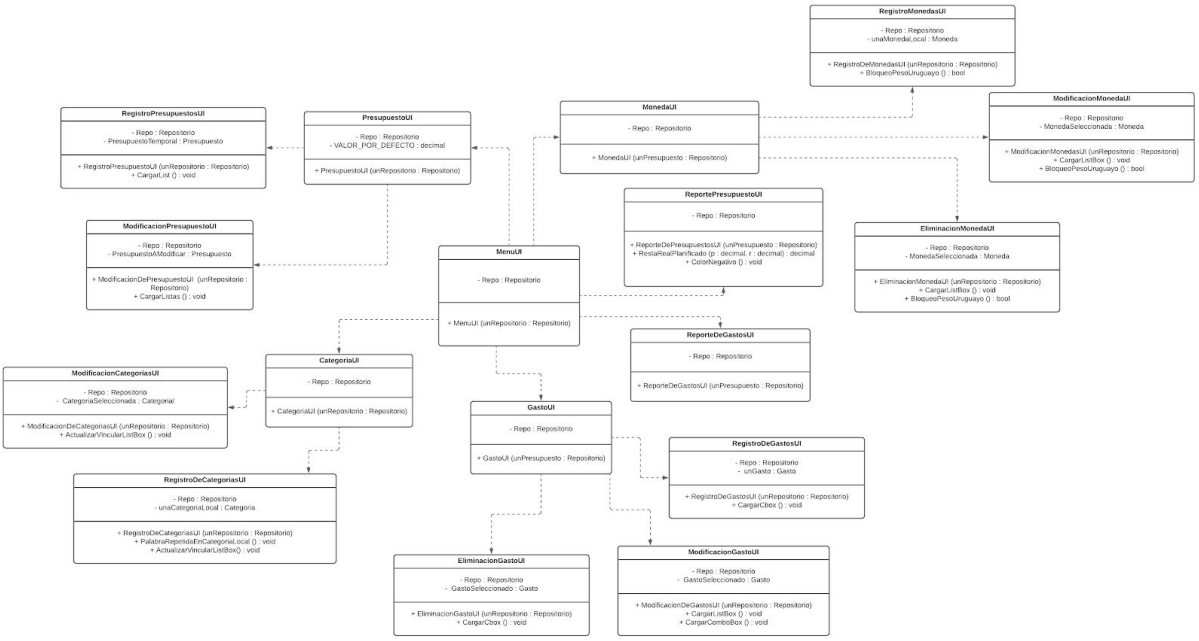
Managers:



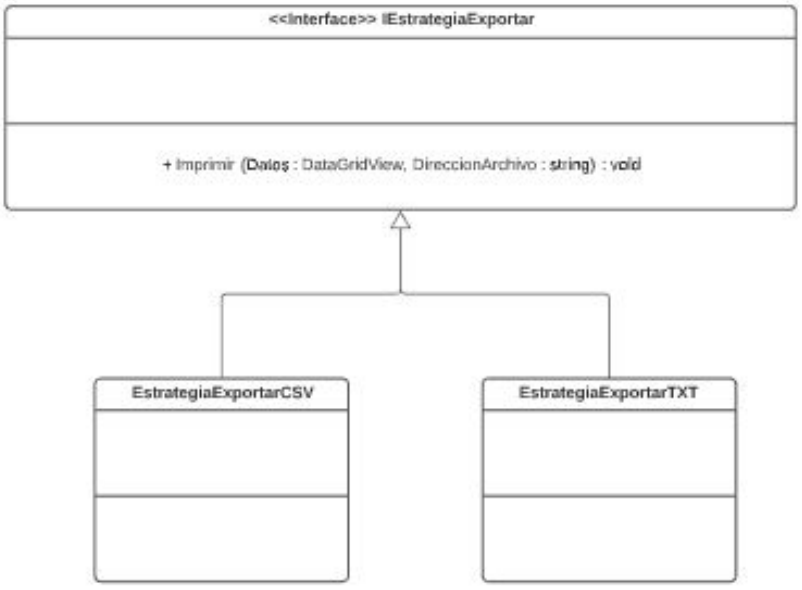
Persistencia:



Interfaz:



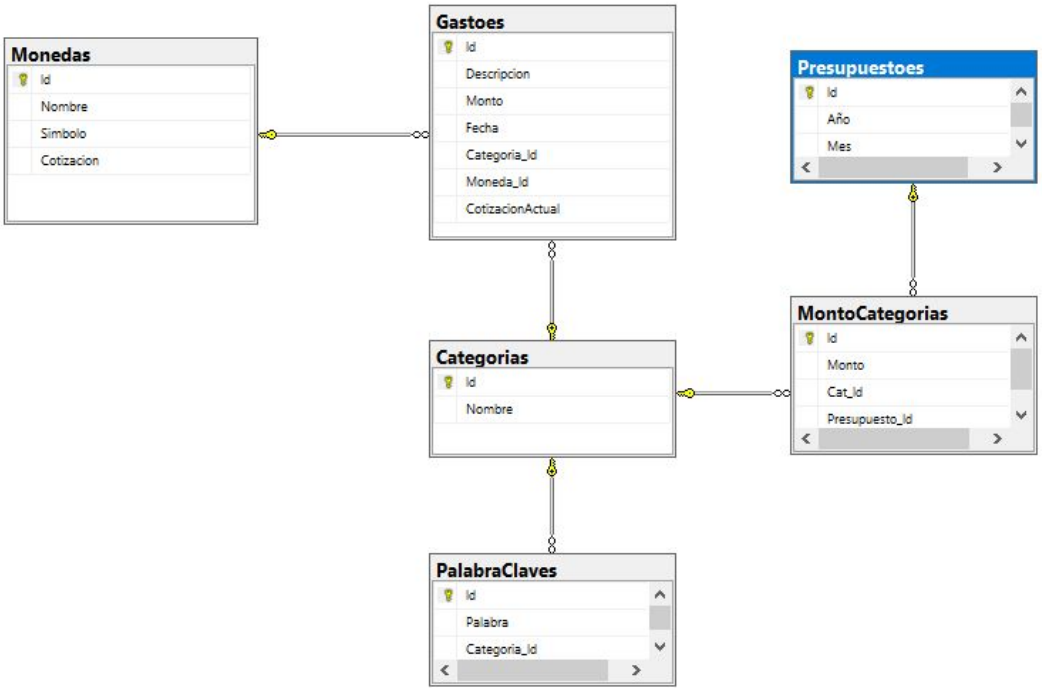
Serialización:



Diagramas de interacción:

Véase Anexo

Diagrama de tablas en la base de datos:



Explicación de los mecanismos generales y descripción de las principales decisiones de diseño tomadas:

El usuario es el que se encarga de navegar por las interfaces según la información que quiera registrar, eliminar o consultar. Para ello el sistema presenta una funcionalidad de persistencia de datos en una base de datos, basada en una clase llamada repositorio. Dicha clase es la que se encarga de almacenar todos los datos.
A su vez implementamos “managers” como enlace entre las clases del dominio y la persistencia de datos. Se optó por incluir las restricciones de negocio dentro de dichos managers ya que se concordó que este proyecto en concreto no alcanzaba a tener una gran magnitud como para separar dichas validaciones en clases como “validators”. Igualmente tenemos en cuenta que, desde el punto de vista de los patrones GRASP, separarlos en estas clases. sería muy bueno desde el punto de vista del bajo acoplamiento ya que se generaría una indirección entre las clases.
Creamos excepciones para los casos en donde el usuario viole las restricciones detalladas en la letra del proyecto y las capturamos, seguido de un mensaje que informe al usuario de dicha situación. Creemos que el feedback siempre debe estar presente en cualquier interfaz.

Al hacer el análisis de las responsabilidades que iba a tener cada entidad de nuestro proyecto, logramos concluir que el paquete de persistencia con su clase repositorio era el indicado para almacenar los datos de todo el sistema. Como ya mencionamos anteriormente el uso de managers de cada categoría del dominio son los encargados de previamente de hacer la persistencia de los datos, aplicar las validaciones necesarias. Las

clases de dominio, presentan la funcionalidad de describir los actores de la realidad planteada en la letra del proyecto. Cabe destacar que, como mencionamos antes, este paquete de Persistencia se creó para separar la responsabilidad de almacenar los datos de las clases del dominio e interfaces.

Para esta nueva entrega, se solicitó en la letra que el sistema presente la funcionalidad de exportar los reportes de gastos a diferentes archivos, que pueden ser de tipo txt, csv, etc. Para ello, generamos un nuevo paquete llamado “Serialización”, en donde están las estrategias para cumplir la funcionalidad. La forma en la que se implementa esta funcionalidad es debido a que seguimos el principio de Abierto-Cerrado perteneciente a SOLID, esto se logró utilizando el patrón de diseño “Strategy”,permitiendo que en caso de que se quiera agregar un nuevo tipo de archivo en donde exportar los reportes de gasto, no sea necesario modificar código que ya está implementado. Queda abierto para agregar nuevas funcionalidades.

Cobertura de pruebas unitarias:

El proyecto fue diseñado a lo largo del tiempo a través de la metodología de TDD (Test Driven Development) por lo que la cobertura de las pruebas no fue un gran problema para nosotros. De igual manera, nos vimos obligados a eliminar el registro de las migraciones a la base de datos, debido a que las mismas bajaban en un gran porcentaje nuestros resultados de cobertura de código. Habiendo hecho esto, logramos alcanzar un promedio de 91.25 % de bloques cubiertos.

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
robak_MARTIN-PC 2020-11-26 1...	266	8.75%	2775	91.25%
managers.dll	5	1.38%	357	98.62%
obligatorio 1 - da1.dll	2	1.15%	172	98.85%
persistencia.dll	231	21.81%	828	78.19%
testing.dll	28	1.94%	1418	98.06%

Claramente los dos paquetes que bajo nuestro punto de vista son más importantes que estén mayoritariamente probados son las clases de dominio y los managers que también utilizan las clases del dominio.

Costo de instalación:

Primero que nada, para lograr correr este sistema adecuadamente, necesitamos tener instalado Sql Server Express dado a que es una parte fundamental del programa. Sin él, el sistema no logrará persistir los datos.

Luego, se debe levantar la base de datos a partir de un backup, que puede ser sin datos o con datos de prueba. En la carpeta Backups en el repositorio se guardarán los scripts con datos y sin datos con sus correspondientes backups para lograr levantar la base de datos con facilidad.
Simplemente con estos pasos, estamos accediendo a trabajar con el programa. Cabe destacar que no es necesario incluir la moneda por defecto para que el programa funcione, ya que ni bien se ejecute la aplicación, dicha moneda se generará automáticamente.

Aclaraciones del sistema:

En esta sección, se detallan algunas funcionalidades del sistema que por un motivo u otro es necesario mencionarlos para que queden documentados.

1) En el código de exportar los reportes de gasto, específicamente en el paquete serialización, la función que se encarga de generar los gastos en las tablas ("Imprimir") recibe en los parámetros un tipo de dato "Datagrid", por lo que es necesario hacer un llamado a las librerías de Windows.Forms. Somos conscientes que esto no es correcto, pero fue la única forma que encontramos de implementar la funcionalidad. Lo mejor sería que dichas clases, no usen librerías de interfaces.
2) Los scripts vacíos de la base de datos, no es necesario que tengan el registro de la moneda por defecto, se puede correr sin tener dicho registro.

Análisis de Gasto:

Entrada / Variable	Clases válidas	Clases inválidas
Descripción	• Más de 3 y menos de 20 caracteres(1)	• Menos de 3 caracteres(5) • Más de 20 caracteres(6)
Monto	• Mayor a 0 con dos decimales(2)	• Menor a 0(7) • Menos de dos decimales(8) • Más de dos decimales(9)
Fecha	• Entre 01/01/2018 y 31/12/2030(3)	• Menor a 01/01/2018(10) • Mayor a 31/12/2030(11)
Categoría	• Ya existente(4)	• Vacía(12)

Registro de gasto:

Caso	Descripción	Monto	Fecha	Categoría	Resultado obtenido	Clases de equivalencia cubiertas
CP 1	“Carrera de autos”	100.50	31/03/2020	Carreras	Se ingresa el gasto	**1,2,3,4**
CP 2	“Do”	100.50	31/03/2020	Carreras	“La descripción tiene que tener entre 3 y 20 caracteres”	**5,2,3,4**
CP 3	“Esto tiene más de 20 caracteres”	100.50	31/03/2020	Carreras	“La descripción tiene que tener entre 3 y 20 caracteres”	**6,2,3,4**
CP 4	“Carrera de autos”	-59.68	31/03/2020	Carreras	”Monto debe ser mayor a 0”	**1,7,3,4**
CP 5	“Carrera de autos”	59.6	31/03/2020	Carreras	La interfaz redondea a dos decimales y se ingresa	**1,8,3,4**
CP 6	“Carrera de autos”	59.589	31/03/2020	Carreras	La interfaz redondea a dos decimales y se ingresa	**1,9,3,4**
CP 7	“Carrera de autos”	59.58	19/03/2001	Carreras	“La fecha debe de estar comprendida entre 2018 - 2030” (Anexo 1.c)	**1,2,10,4**

CP 8	“Carrera de autos”	59.58	30/10/2031	Carreras	“La fecha debe de estar comprendida entre 2018 - 2030” (Anexo 1.c)	**1,2,11,4**
CP 9	“Carrera de autos”	59.58	31/03/2020		“Hay campos Vacíos” (Anexo 1.d)	**1,2,3,12**

Eliminación de Gasto

Caso	Gasto	Monto	Fecha	Categoría	Resultado obtenido
CP 10	“Carrera de autos”	100.50	31/03/2020	Carreras	Se elimina el gasto
CP 11					No se elimina el gasto, la interfaz no hace nada

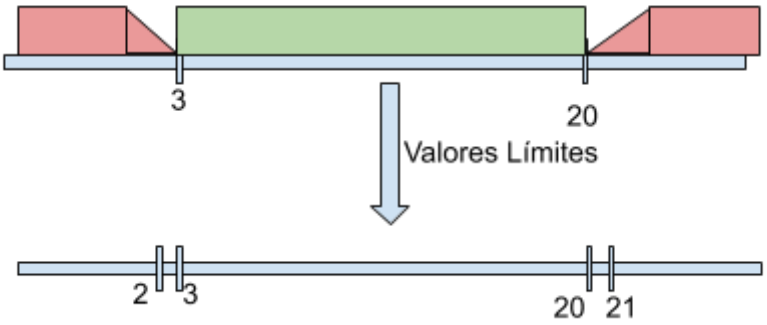
Modificación de Gasto (Igual a registro de gasto)

Caso	Descripción	Monto	Fecha	Categoría	Resultado obtenido	Clases de equivalencia cubiertas
CP 12	“Carrera de autos”	100.50	31/03/2020	Carreras	Se modifica el gasto	**1,2,3,4**
CP 13	“Do”	100.50	31/03/2020	Carreras	“La descripción tiene que tener entre 3 y 20 caracteres”	**5,2,3,4**

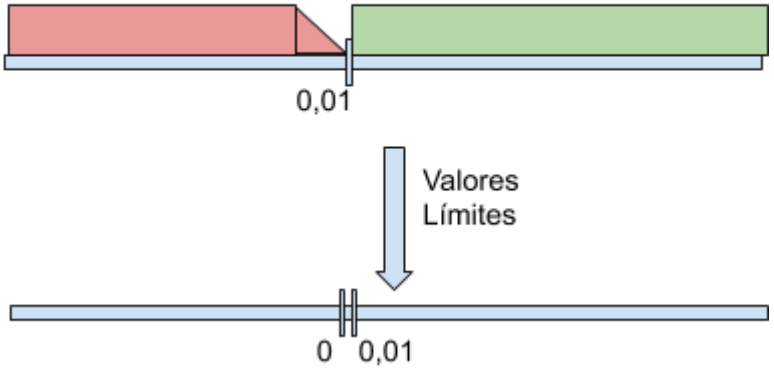
CP 14	“Esto tiene más de 20 caracteres”	100.50	31/03/2020	Carreras	“La descripción tiene que tener entre 3 y 20 caracteres”	**6,2,3,4**
CP 15	“Carrera de autos”	-59.68	31/03/2020	Carreras	”Monto debe ser mayor a 0”	**1,7,3,4**
CP 16	“Carrera de autos”	59.6	31/03/2020	Carreras	La interfaz redondea a dos decimales y se modifica	**1,8,3,4**
CP 17	“Carrera de autos”	59.589	31/03/2020	Carreras	La interfaz redondea a dos decimales y se modifica	**1,9,3,4**
CP 18	“Carrera de autos”	59.58	19/03/2001	Carreras	“La fecha debe de estar comprendida entre 2018 - 2030”	**1,2,10,4**
CP 19	“Carrera de autos”	59.58	30/10/2031	Carreras	“La fecha debe de estar comprendida entre 2018 - 2030”	**1,2,11,4**
CP 20	“Carrera de autos”	59.58	31/03/2020		“Hay campos Vacíos”	**1,2,3,12**

Valores límites para Registro y Modificación

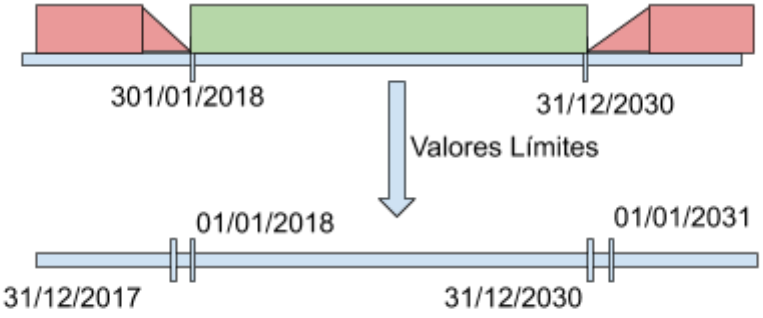
- Cantidad de caracteres de descripción



- Valor Monto

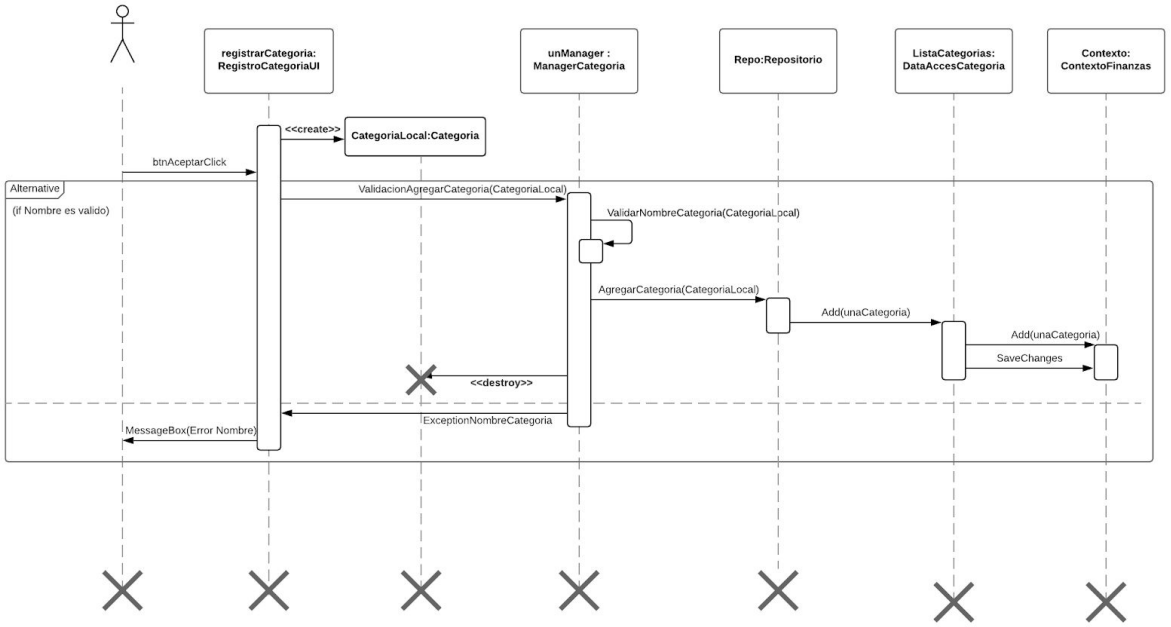


- Fecha

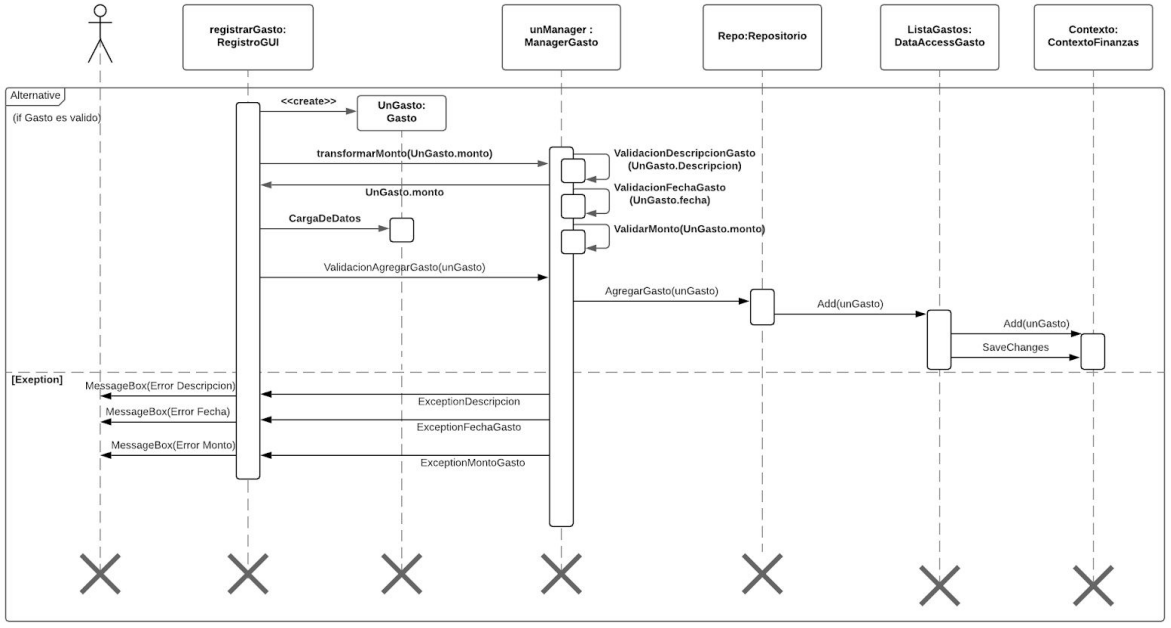


Anexo

Registro Categoria:



Registro Gasto:



Registro Moneda: