

# Cash Manager

Enunciado, Especificación de requerimientos funcionales y Mockups

Danilo Vicente Erazo Meza - A00369481

Jhan Carlos Carvajal Bastidas - A00368888

Facultad de Ingeniería, Universidad ICESI

Algoritmos y Programación II

Juan M. Reyes

Mayo 2021

## Enunciado

Debido a los vacíos de educación financiera que se pueden generar dentro de los procesos educativos y de formación de secundaria básica y media, gran parte de las personas coinciden en que no tienen una educación financiera aceptable o satisfactoria que les permita hacer un seguimiento de sus finanzas personales. Con esto en mente, un programa que permita llevar un registro contable preciso y específico de todos los ingresos y gastos de una persona representa una herramienta para gestionar de mejor forma el dinero.

El programa debe permitirle al usuario hacer un registro de sus gastos, ingresos, deudas y ahorros de acuerdo a categorías que pueden estar definidas por él o solo usar las categorías por defecto. Cada categoría tiene un nombre, un monto total acumulado, un monto máximo en caso de gasto o una meta en caso de ahorro. Para el caso de las deudas, estas tienen el capital a pagar, el saldo y el número de pagos realizados. También el usuario debe poder modificar los datos registrados en el programa (ingresos, gastos, deudas y ahorros), los únicos campos que no se verán modificados son la fecha y hora en la que se realizó el registro del dato, haciendo clic sobre el dato y seleccionando "editar". El programa también cuenta con una opción para eliminar cualquier dato registrado haciendo clic sobre el dato y seleccionando "eliminar".

Además, debe permitirse gestionar (editar, añadir y eliminar) los tipos de cuenta que maneja el usuario, es decir, débito, crédito y bolsillo. Cada cuenta posee un nombre, un total de dinero, un acumulado de movimientos (retiros, abonos, paso entre cuentas). Para el caso de la cuenta de crédito, debe calcularse la cuota mensual en función de los gastos registrados y del interés de la misma y posee el cupo máximo.

Para cada cuenta, el sistema debe mostrar los movimientos registrados de forma similar a un extracto bancario, cada movimiento contiene la fecha del mismo, el monto y el tipo de movimiento. Cada movimiento realizado se debe poder modificar o eliminar.

El sistema debe generar un reporte por ingreso, ahorro y gastos donde se pueda saber el total de movimientos que corresponden a cada uno, con su categoría, monto y fecha. Adicionalmente el programa debe tener una opción para generar un reporte donde se muestre en un gráfico de pastel la distribución de los ingresos, ahorros o gastos en función de su categoría, un gráfico donde se relacionan los ingresos, ahorros y gastos, y otras gráficas enfocadas a cada una de las categorías previamente mencionadas. Por ejemplo, una gráfica para ahorros, esta gráfica permite ver qué porcentaje ocupan las diferentes subcategorías de ingresos en el total de ingresos. Cada categoría en la gráfica debe estar representada con distinto color.

El usuario debe poder importar y exportar un archivo el cual contiene datos(ingresos, gastos, ahorro y deudas), donde además se especifica la cuenta a la cual pertenece cada dato, esto básicamente para permitir al usuario pasar sus datos a otro dispositivo en caso

de requerirlo. El programa, de forma permanente debe mostrar al usuario un reloj con fecha y hora actuales

## Requerimientos funcionales

El sistema debe estar en la capacidad de:

**Req1. Realizar un respaldo de todos los datos del programa.** El programa debe realizar un respaldo mediante serialización cuando el usuario lo desee (mediante una opción). El usuario puede también decir al sistema que realice un respaldo cada día, cada semana o cada mes.

**Req2. Añadir categorías.** El programa debe tener unas categorías preestablecidas, sin embargo, si el usuario quiere se debe permitir, mediante una o varias opciones, añadir nuevas categorías, las cuales son nombres para identificar y agrupar los datos ingresados. Las categorías sólo tienen nombre.

**Req2.1. Añadir categorías de ingresos.**

**Req2.2. Añadir categorías de gastos.**

**Req2.3. Añadir categorías de deudas.**

**Req3. Modificar categorías.** El programa debe permitir, mediante una o varias opciones, modificar los nombres de las categorías de gastos, ingresos y deudas que se encuentran ya registradas, incluso las categorías que se encuentran por defecto.

**Req4. Añadir tipos de cuentas.** El programa debe permitir añadir nuevos tipos de cuentas al listado ya existente, el programa debe tener un listado base de tipos de cuenta (por defecto). Por ejemplo, tarjetas, billetera online, efectivo, etc.

**Req5. Modificar los tipos de cuentas.** El programa debe permitir mediante una opción modificar el nombre de los tipos de cuenta.

**Req6. Mostrar el capital total con el que cuenta el usuario.** El programa debe mostrar la suma del monto que poseen todas las cuentas del usuario, esto con el fin de informar acerca del capital que posee al momento, este debe actualizarse cada vez que se registren ingresos o gastos.

**Req7. Añadir una contraseña de acceso.** El programa debe presentar una opción para que el usuario registre un número de 4 dígitos a modo de contraseña de acceso, para posteriormente acceder al programa.

**Req8. Mostrar gráficas con resumen de los datos.** El programa debe tener una o varias opciones para mostrar una gráfica de pastel donde se verá reflejado como influye cada categoría en un total. Además, toda categoría dentro del gráfico debe presentarse de un color distinto.

**Req8.1. Mostrar gráfica de gastos e ingresos juntos.** Una de las gráficas desplegadas por el programa debe mostrar los gastos e ingresos y el porcentaje que ocupan dentro del total. En este caso los gastos e ingresos se toman como categorías completas, es decir sin sus subcategorías.

**Req8.2. Mostrar gráfica de ingresos.** Otra de las gráficas debe mostrar la información referente a ingresos, tomando esta categoría como un total y representando todas sus subcategorías en el gráfico. Además, se debe mostrar el porcentaje que estas subcategorías ocupan dentro del total de ingresos.

**Req8.3. Mostrar gráfica de gastos.** Otra de las gráficas debe mostrar la información referente a gastos, tomando esta categoría como un total representando todas sus subcategorías en el gráfico. Además, se debe mostrar el porcentaje que estas subcategorías ocupan dentro del total de gastos.

**Req8.3. Mostrar gráfica de deudas.** Otra de las gráficas debe mostrar la información referente a deudas, tomando esta categoría como un total representando todas sus subcategorías en el gráfico. Además, se debe mostrar el porcentaje que estas subcategorías ocupan dentro del total de deudas.

**Req9. Eliminar datos del programa.** El programa debe contar con una opción para eliminar todos los datos del programa en el caso de que el usuario lo desee. Si el usuario tiene habilitada la opción de contraseña para el programa entonces se le debe solicitar esta contraseña antes de realizar la eliminación.

**Req10. Listar datos del programa.** Se debe contar con una pantalla o varias pantallas en la/s cual/es se presenten los datos correspondientes a ingresos, gastos, deudas y ahorros. El usuario debe tener acceso a una opción para ordenar cualquiera de estas categorías de datos por fecha o por monto.

**Req11. Registrar un nuevo dato.** Cada dato, es decir, un ingreso, gasto, ahorro o deuda, tiene una fecha (incluye el formato AM/PM) , ambas son tomadas del sistema justo en el momento en el que el usuario ingresa a la opción para registrar el dato. Por otro lado, tiene una cuenta, es decir de donde se gastó el dinero o hacia que cuenta entra como ingreso, una categoría, un monto en COP (peso colombiano) y una nota, aquí el usuario de así quererlo, da más información sobre la acción realizada.

**Req11.1. Registrar un ingreso.**

**Req11.2. Registrar un gasto.**

**Req11.3. Registrar una deuda.** Cuando se registra una deuda el usuario puede poner la cantidad de cuotas a las que desea pagar esta deuda y el programa le mostrará el monto para cada cuota, esto con el fin de mostrar al usuario cuantas cuotas le quedan por pagar.

**Req11.4. Registrar un ahorro.**

**Req12. Editar un dato registrado.** El programa debe permitir al usuario modificar un dato ya registrado, dando clic sobre el dato y seleccionando editar. Los únicos campos que no pueden ser modificados son la fecha y hora en la que se realizó el registro de dicho dato.

**Req13. Eliminar un dato.** El programa debe contar con una opción para eliminar un dato dando clic sobre este y seleccionando eliminar.

**Req14. Mostrar un reloj.** El programa debe mostrar en todo momento un reloj con fecha y hora actuales.

**Req15. Crear bolsillos.** Si la pantalla en la que se encuentra el usuario es la de cuenta débito entonces el programa debe presentarle la opción para crear un bolsillo. Un bolsillo tiene un nombre, y una opción para añadir fondos al bolsillo, este dinero sale de la cuenta en la que se creó el bolsillo. Se resta del total de la cuenta el monto que ingresa al bolsillo, pero es necesario mostrar el monto que posee el bolsillo bajo el monto de la cuenta en sí, ya que este dinero no desaparece solo fue reubicado.

**Req16. Guardar toda la información del programa.** La información del programa debe ser persistente, es decir, ante un nuevo ingreso los datos de la sesión previa deben seguir en el programa.

**Req17. Importar datos.** El programa debe tener una opción que le permite importar al usuario importar datos de una copia de respaldo que este posea. Este documento no incluye configuraciones del programa o algo similar, sólo información de gastos, ingresos, deudas y ahorros.

**Req18.** El sistema debe generar un reporte por ingreso, ahorro y gastos donde se pueda saber el total de movimientos que corresponden a cada uno, con su categoría, monto y fecha, este reporte se realiza mediante una pantalla dentro del programa.

## Justificación:

El problema presentado abarca una serie de gestiones extensas de diferentes instancias del mundo del problema. Adicionalmente la parte de análisis gráfico pretende ser manejada a través de gráficos 2D como herramienta. Todas las instancias están relacionadas a los conceptos de persistencia y deserialización. La solución planteada pretende abarcar un gran abanico de opciones para el usuario, esto con el fin de hacer el programa lo más útil posible para el manejo de las finanzas de quien lo use y maximizar su control del dinero.

## Requerimientos no funcionales:

**ReqNF1.** Los datos guardados por el usuario serán transparentes y su persistencia debe ser asegurada siempre y cuando las condiciones de memoria del dispositivo que ejecuta el programa lo permitan.

**ReqNF2.** La interfaz gráfica de usuario debe contar con la internacionalización en todos los textos y mensajes que maneja.

**ReqNF3.** El sistema debe ser capaz de gestionar al menos 5000 registros de movimientos monetarios en total, es decir, sumando los de cada cuenta registrada.

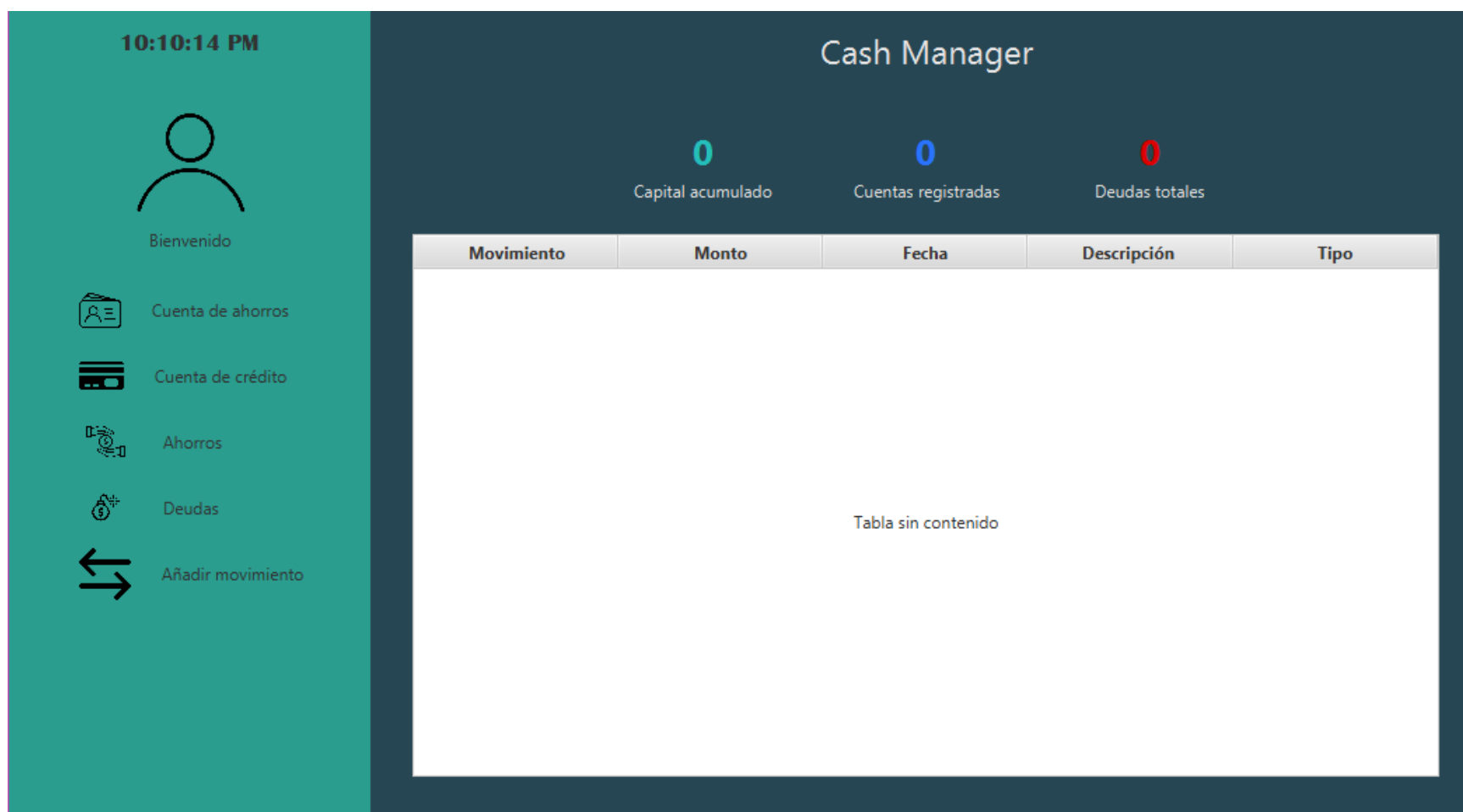
**ReqNF4.** La seguridad de los movimientos monetarios debe estar garantizada por una clave de acceso si el usuario así lo requiere.

**ReqNF5.** El importe y exporte de los datos requeridos por el usuario no deben interferir con la disponibilidad de la interfaz gráfica y demás acciones contenidas en la misma.

**ReqNF6.** El importe y exporte de datos requerido por el usuario debe estar en la capacidad de procesar al menos 1000 datos por cada vez que se realice la acción.

## Mockups:

Esta es la pantalla principal del programa, permite el movimiento a través de las diferentes cuentas que el usuario posee así como la interacción de dinero entre estas y añadir ingresos y gastos de forma rápida






Estas dos pantallas son para crear cuentas, es decir, instancias de cuentas de ahorros, crédito, ahorros como tal y deudas. El ejemplo muestra la creación de un ahorro con sus atributos. Por otro lado se muestra la creación de un movimiento




The screenshot displays the 'Crear movimiento' (Create movement) form. On the left is a teal sidebar with a clock showing 10:13:52 PM, a user icon, and a 'Bienvenido' (Welcome) message. Below these are five menu items: 'Cuenta de ahorros' (Savings account), 'Cuenta de crédito' (Credit account), 'Ahorros' (Savings), 'Deudas' (Debts), and 'Añadir movimiento' (Add movement), each with a corresponding icon. The main area has a dark blue background. At the top, a teal bar contains the title 'Crear movimiento'. Below this, the text 'Cuentas disponibles' (Available accounts) is followed by a dropdown menu currently set to 'Cuentas'. The form fields include: 'Monto' (Amount) with a text input; 'Fecha y Hora' (Date and Time) with a date/time picker showing '2021/05/30 22:13:49'; 'Descripción' (Description) with a text input; 'Tipo' (Type) with a dropdown menu set to 'Tipos'; and 'Categoría' (Category) with a dropdown menu set to 'Categorías'. At the bottom, there are two teal buttons: 'Añadir movimiento' (Add movement) and 'Eliminar movimiento' (Delete movement).


11:13:07 PM



Bienvenido




Cuenta de ahorros




Cuenta de crédito



Ahorros



Deudas



Añadir movimiento

### Crear cuenta nueva

Tipo de cuenta Deuda

Nombre

Interés

Número de cuotas

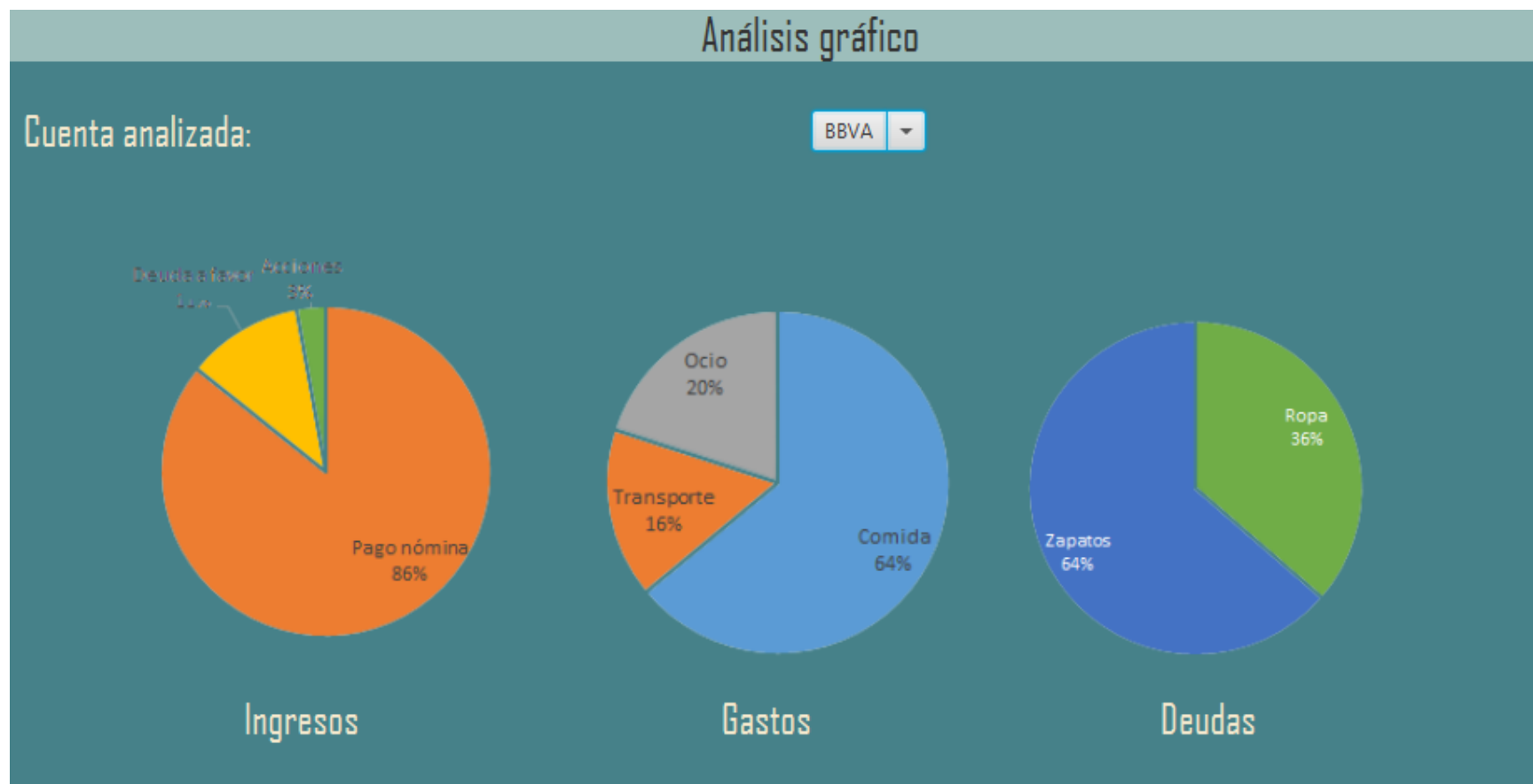
Dinero a pagar

Crear cuenta nueva

Esta pantalla es la presentación general de los tipos de cuenta, al seleccionar una del menú desplegable se cargan las demás opciones. La tabla permite visualizar los movimientos realizados en la cuenta y los botones auxiliares permiten gestionar movimientos o generar análisis.



Es la pantalla desplegada al seleccionar el análisis gráfico de una cuenta en específico



```

classDiagram
    class Movement {
        <<enumeration>>
        INCOME
        SPEND
        WITHDRAWAL
    }
    class Account {
        <<abstract>>
        +name: String
        +accountName: String
    }
    class SavingAccount {
        +availableMoney: double
        +changeAccountName: String, money: double
    }
    class CreditAccount {
        +interest: double
        +monthlyFee: double
        +monthlyRate: double
        +creditAccountName: String, interest: double, maxQuota: double
        +creditLimitLeft: double
    }
    class MoneyManagement {
        +moneyManagement: String
        +cashAmount: double
        +moneyManagementName: String, max: double
        +getMoneyAmount: double
        +setMoneyAccountBalance: double
    }
    class CreditManager {
        +creditManager
        +creditSavingAccountName: String, money: double
        +creditWithdrawalAccountName: String, interest: double, quota: double
        +creditWithdrawal: String, interest: double, fee: money: double
        +creditWithdrawal: String, money: double
        +creditWithdrawal: int, name: String, loanId
        +addMovementCreditMovement: Movement, creditMovement: Movement
    }
    class User {
        +name: String
        +password: String
        +login: String, password: String
        +logout: String
    }
    class Base {
        +name: String
        +password: String
        +login: String, password: String
        +logout: String
    }
    class Main {
        +main()
    }

    Movement --> Account : spend
    Account <|-- SavingAccount
    Account <|-- CreditAccount
    MoneyManagement --> Movement
    MoneyManagement --> Account
    CreditManager --> User
    CreditManager --> Base
    CreditManager --> Movement : getMovement
    User --> Base
    Base --> Movement
    Main --> Base
    Main --> Movement
  
```