UNIVERSIDADE DA CORUÑA

Facultade de Informática

Trabajo fin de grado

Grado en Ingeniería Informática

Mención en Tecnologías de la Información

Aplicación para el análisis de carteras de fondos de inversión

---

Autor: López López, Ángel
Director: Castro Castro, Paula María
Director: González Coma, José Pablo

A Coruña, diciembre de 2016

Índice

Capítulo I
Introducción

1. Introducción al mundo financiero

Para poder llevar a cabo este proyecto, ha sido necesario realizar un primer paso de búsqueda de información acerca del mundo de las finanzas, mas concretamente sobre los fondos de inversión, para poder conocer su funcionamiento, sus métricas y los tipos de datos que en ellos se utilizan.

1.1. Fondos de inversión

Para comenzar empezaremos definiendo que es un fondo, como funciona y los elementos que en el intervienen:

Un **fondo de inversión** es un capital compuesto por la suma de las aportaciones monetarias realizadas por varias personas. Este capital se invertirá en una serie de activos con el objetivo de obtener la máxima rentabilidad posible. Dependiendo de la evolución de estos activos, el fondo arroja resultados positivos o negativos, los cuales se repartirán entre cada inversor según la proporción que represente su inversión sobre el total del patrimonio del fondo. Cada fondo se encuentra identificado por un International Securities Identification Number (ISIN), este código identifica unívocamente un valor mobiliario a nivel internacional.

Los fondos de inversión se dividen en partes proporcionales llamadas **participaciones** y sus propietarios se denominan **participes**. El número de participaciones no es fijo, sino que depende de las compras y ventas de las mismas. Su valor, denominado **Valor Liquídativo (VL)** de la participación, se calcula diariamente de la siguiente manera:

$$Valor\ liquidativo = \frac{Patrimonio\ del\ fondo}{N\ de\ participaciones\ en\ circulacion} \tag{1}$$

Este valor depende, por tanto, de la evolución diaria de los valores que componen el patrimonio del fondo y será uno de los indicativos fundamentales que utilizará la aplicación a la hora de realizar los históricos de los diferentes fondos. Otra medida importante es la **rentabilidad del fondo**, esta se calcula mediante el porcentaje entre el VL en la fecha de compra de la participación (suscripción) y la fecha de venta (reembolso), de la siguiente manera:

$$Rentabilidad = \frac{Valor\ liquidativo\ final - Valor\ liquidativo\ inicial}{Valor\ liquidativo\ inicial} * 100 \tag{2}$$

El resultado no es percibido de manera efectiva hasta que no se produzca el reembolso de las participaciones y será en ese momento en el que el partícipe deberá tributar por el resultado de su inversión.

Otro aspecto importante es que las decisiones de la inversión las toma una **gestora**, que administra y representa el fondo, mientras que la función de custodiar y vigilar los activos la realiza el llamado **depositario**, generalmente una entidad financiera. Normalmente la gestora cobra una serie de comisiones de gestión que se restan al fondo, lo cual disminuye el VL de cada participación.

Los siguientes puntos se centrarán en ver los distintos tipos de fondos que podemos encontrar, los criterios que se deben de utilizar para su elección y las operaciones que podemos realizar sobre ellos.

1.1.1. Tipos de fondos

En el mercado existen una amplia gama de fondos de inversión, es tarea del propio inversor elegir aquel que más se adapte a sus necesidades.

- **Fondos de renta fija:** Son fondos donde la mayoría de sus activos son de renta fija (obligaciones y bonos, letras, pagarés, etc). Normalmente, la rentabilidad de estos fondos va ligado al plazo de vencimiento de dichos activos, es decir, a menor plazo, menos riesgo y por lo tanto menos rentabilidad prevista y viceversa.
- **Fondos de renta variable:** Son fondos donde la mayoría de sus activos son de renta variable (acciones). Por lo general, los fondos de renta variable reportan ganancias o rendimiento a largo plazo, a cambio de un mayor riesgo.
- **Fondos Mixtos:** Son fondos en los que sus activos se encuentran divididos entre activos de renta fija y renta variable. Cuanto mayor sea el porcentaje de activos de renta variable mayor sera el riesgo y la rentabilidad potencial.
- **Fondos globales:** Son fondos que suelen incluir renta variable, fija y activos monetarios en diferentes localizaciones geográficas, en determinados porcentajes dependiendo de la política del fondo, de forma que sus inversiones estén muy diversificadas.
- **Fondos garantizados:** Son fondos que aseguran la recuperación del capital inicialmente invertido más una rentabilidad fija o variable, en una fecha futura determinada.
- **Fondos monetarios:** Son fondos basados en la adquisición de activos a corto plazo para minimizar el riesgo de la inversión obteniendo la máxima rentabilidad posible.

1.1.2. Criterios para elegir un fondo de inversión.

Como hemos visto en el apartado anterior, existen varios tipos de fondos de inversión adaptados a diferentes necesidades. A la hora de elegir un fondo en particular existen varios ratios e indicadores que pueden ayudar a determinar cual es el mas adecuado a las preferencias del inversor.

Normalmente, a la hora de seleccionar un fondo, el inversor debe considerar cual es su capacidad de asumir de pérdidas (pues cuanto mayor es el riesgo también lo es la rentabilidad) así como el horizonte temporal durante el cual desea mantener la inversión, pues, dependiendo de la política del fondo, puede ser aconsejable estar dispuesto a mantener la inversión un determinado período de tiempo.

Otro aspecto a tener en cuenta son las comisiones que se cargan a los fondos de inversión, puesto que pueden afectar a la rentabilidad. Es posible que un fondo aplique distintos tipos de comisiones a las diferentes tipos de participaciones que emita.

También hemos de considerar el comportamiento histórico que ha tenido un fondo a lo largo del tiempo. Es importante conocer las rentabilidades obtenidas en el pasado, aunque esto no signifique que se siga una línea similar en el futuro. En la aplicación a desarrollar se incluirán históricos de las rentabilidades referidas a un determinado período (trimestre, semestre ...) para que al comparar distintos fondos se puedan contrastar las rentabilidades en los mismos períodos. Cabe mencionar que es necesario que los fondos sigan una misma política de inversión para que la comparación sea significativa.

Es posible que durante la vida de un fondo este cambie su política de inversión e incluso de grupo gestor, por lo que al consultar rendimientos pasados hay que tener en cuenta que puede que estes hayan cambiado, es importante conocer la fecha de dicho cambio y tener en cuenta sólo las rentabilidades a partir de ese momento.

Por último, algunas métricas o indicadores que se deben utilizar para elegir un fondo de inversión son los siguientes:

- **Volatilidad:** es una medida de variación (cambios) en el precio de un activo. Mide cuanto varía el precio de un activo respecto a su precio medio y cuantifica el riesgo del activo financiero.
- **Alfa:** mide la capacidad o habilidad que tiene el gestor de generar valor al fondo de inversión.
- **Beta:** mide la sensibilidad del VL de un fondo a los movimientos de su índice de referencia.
- **Ratio de Sharpe:** nos dice lo bueno que es un fondo de inversión en la relación rentabilidad-riesgo.
- **Ratio de Información:** es una medida que se emplea para determinar la influencia que ha tenido un gestor en la rentabilidad del fondo en comparación con el comportamiento del mercado.
- **Máximo Drawdown:** se define como la máxima caída experimentada por un fondo en el periodo comprendido desde que se registra un máximo, hasta que vuelve a ser superado.

1.1.3. Operaciones y seguimiento de fondos

En este último punto hablaremos sobre las operaciones de suscripción, reembolso y traspaso de un fondo de inversión así como de como realizar el seguimiento de su rentabilidad.

El método para realizar una inversión en un fondo consiste en la **suscripción** de participaciones, la entidad gestora emite una serie ellas y cada inversor obtiene tantas como el resultado de dividir el capital invertido entre el VL (1.1) aplicable a la operación. Normalmente el VL aplicable es el del mismo día de la solicitud o el del día siguiente a la solicitud. Algunos fondos pueden estar sujetos a comisiones de suscripción,de hasta hasta un 5 % de la inversión.

Si un inversor quiere recuperar su dinero debe solicitar un **reembolso** de todas o parte de sus participaciones, recibiendo el resultado de multiplicar el el VL(1.1) de la participación por el número de participaciones que quiera reembolsar. El VL aplicable es el mismo que en el caso anterior, el del mismo día o el del día siguiente. El plazo en el que el inversor recibe su dinero es de un máximo de 3 a 5 días, pudiendo tener dicho reembolso una comisión de hasta el 5 % como en el caso anterior. El inversor conocerá el resultado de la inversión(positivo o negativo) cuando se le abone el reemboslo.

En el caso de querer realizar un **traspaso** de un fondo a otro se produce un reembolso del primero y la inmediata suscripción al segundo. Este método tiene una ventaja, pues se conserva la antigüedad de la primera inversión a efectos fiscales, por lo que las plusvalías no se tributan hasta que se produzca el reembolso definitivo.

Existen cuatro partes que intervienen en un traspaso:

- **Fondo de origen:** fondo en el que se mantiene la inversión antes del traspaso.
- **Fondo de destino:** fondo en el que quiere invertir el capital que se reembolse del fondo de origen.
- **Entidad de origen:** la que comercializa o gestiona el fondo de origen.
- **Entidad de destino:** la que comercializa o gestiona el fondo de destino.

Sin embargo, al tratarse de de una operación de reembolso y suscripción, se deberán abonar las respectivas comisiones que tengan establecidas ambos fondos.

El proceso de **seguimiento** de un fondo de inversión puede realizarse principalmente a través de dos fuentes:

- La documentación que proporcione la entidad gestora o depositaria. Pues es obligatorio que se proporcione a los partícipes información periódica acerca de la evolución de sus inversiones.
- La divulgación de datos sobre fondos de inversión que proporcionan periódicos o diversos portales de internet. De esta última fuente obtendremos los datos necesarios para el funcionamiento inicial de la aplicación.

Capítulo II
Metodología

2. Proceso Unificado de Desarrollo Software

Para la realización de este proyecto se utilizará la metodología de **Proceso Unificado de Desarrollo Software (PUD)**. El PUD * Es un marco de desarrollo extensible, dirigido por casos de uso, iterativo e incremental, en el cual los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones.

El PUD presenta las siguientes características:

- **Esta dirigido por casos de uso:** Cada caso de uso representa un requisito funcional y su conjunto forma el modelo de casos de uso.
- **Esta centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo y describe los elementos del modelo que son mas importantes para poder desarrollarlo.
- **Iterativo e incremental:** El trabajo es dividido en tareas mas pequeñas o iteraciones. El resultado de cada iteración es un sistema ejecutable, una nueva versión del producto final. Cada una de estas iteraciones resulta en un incremento en el proyecto y se divide a su vez en: análisis de requisitos, diseño, implementación y prueba.

Como lenguaje de representación visual el PUD utiliza el Lenguaje Unificado de Modelado (UML) y se ha seleccionado para este proyecto porque esta concebido para la programación orientada a objetos, acelera el ritmo del desarrollo y reduce el coste del riesgo a un solo incremento.

3. Planificación y evaluación de costes

En este apartado se detalla como se ha aplicado el PUD para gestionar el desarrollo de la aplicación. Debido a que se utiliza un marco de desarrollo incremental, en cada iteración se presentará un nuevo caso de uso que conformará una nueva versión del producto final.

El objetivo de este Trabajo de Fin de Grado (TFG) es implementar una aplicación en la que los usuarios puedan obtener gráficos, datos numéricos y resultados de una o varias carteras de fondos de inversión, por lo cual las fases en las que se divide el proyecto son las siguientes:

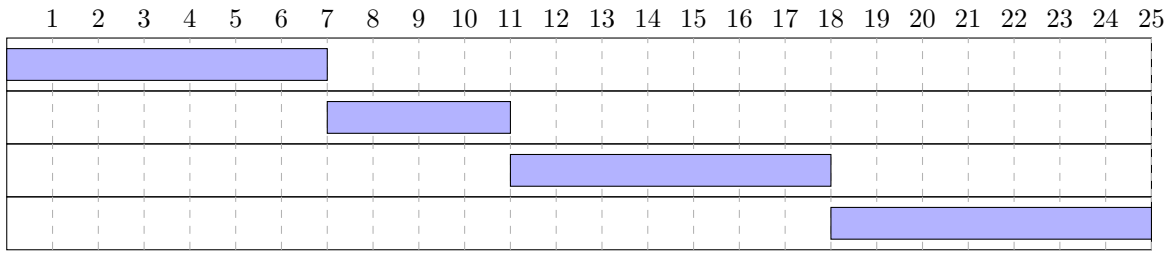
Tabla 1: Planificación del proyecto

Fase	Iteración	Tareas
Inicial	Preliminar	Inmersión en el mundo financiero
		Elección de indicadores, medidas y criterios más significativos
		Localización y selección de las fuentes de datos
		Definir las diferentes iteraciones en las que se realizará el proyecto
Diseño	1	Búsqueda de requisitos funcionales
	2	Realizar los modelos de los casos de uso
Desarrollo	1	Desarrollo de la IT 1
	2	Desarrollo de la IT 2
	3	Desarrollo de la IT 3
	4	Desarrollo de la IT 4
Documentación	Final	Redacción del manual de usuario

Para una mayor monitorización de las tareas a realizar se incluye a continuación un diagrama de Gantt con la planificación temporal estimada para cada tarea.

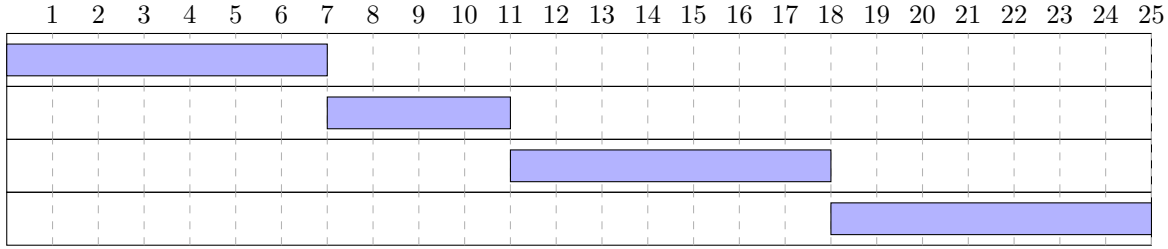
Tareas

1. Introducción al mundo financiero
2. Metodología, Planificación y costes
3. Formato de la base de datos
4. Tecnologías empleadas



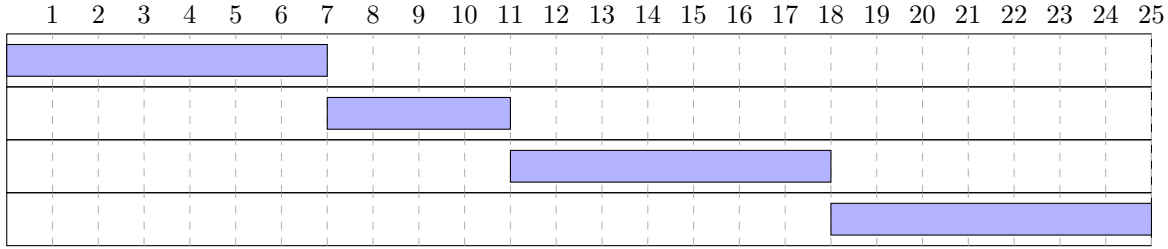
Tareas

1. Introducción al mundo financiero
2. Metodología, Planificación y costes
3. Formato de la base de datos
4. Tecnologías empleadas



Tareas

1. Introducción al mundo financiero
2. Metodología, Planificación y costes
3. Formato de la base de datos
4. Tecnologías empleadas



4. Revisión de fundamentos tecnológicos

En este apartado se describen las herramientas y tecnologías usadas para el desarrollo del proyecto.

4.1. Herramientas para la gestión de proyectos

- **Git:** Es un software de control de versiones distribuido que permite trabajar en grupo y mantener accesibles las diferentes versiones de la aplicación del proyecto. Se ha elegido git como sistema de gestión de versiones del proyecto por encima de su principal alternativa, subversion porque proporciona un repositorio local sobre el que se puede trabajar off-line.

4.2. Herramientas para el modelado de software

- **Visual Paradigm Community Edition:** Es una herramienta para el desarrollo de aplicaciones utilizando UML recomendada para la aplicación y el seguimiento del PUD. Proporciona asistencia para realizar los análisis, diseño, casos de uso y modelos UML del proyecto. Se trata de un software con licencia pero se utiliza su versión gratuita (Community Edition).

4.3. Herramientas para el desarrollo del proyecto

- **Eclipse IDE Neon:** Es una plataforma de desarrollo de software de código abierto y multiplataforma basada en Java. Proporciona Entornos de Desarrollo Integrados (IDEs) prácticamente para cualquier lenguaje, siendo el mas utilizado el de Java. Provee también de una serie de plugins para el control de versiones y frameworks para el desarrollo de aplicaciones gráficas.

  Se ha elegido Eclipse como plataforma de desarrollo de este proyecto en lugar de Netbeans debido a que ha sido la plataforma utilizada en la mayoría de las asignaturas del grado, por lo que su funcionamiento es mas conocido. Para la sincronización con el repositorio git se utiliza el plugin EGit.

- **Apache Maven:** Es un software de gestión de proyectos software desarrollado por Apache. Esta basado en el concepto de Project Object Model (POM), maven permite gestionar dependencias, módulos, componentes y el orden de construcción.

4.4. Herramientas de bases de datos

- **MySQL:** Es un sistema de gestión de bases de datos relacional desarrollado por Oracle Corporation y está considerada como la base datos open source más popular del mundo. Se utiliza en este proyecto para guardar los datos de los diferentes fondos de inversión de tal forma que puedan estar disponibles off-line.

  Su principal alternativa es PostgreSQL, pero se ha decidido utilizar MySQL porque nuestra aplicación utilizará principalmente consultas sencillas, normalmente de lectura y MySQL esta orientado a este tipo de tareas proporcionando un mayor rendimiento que su competidor.

- **Hibernate:** Es un framework para el mapeo objeto-relacional de código abierto para la plataforma Java. Su uso facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos denominados eXtensible Markup Language (XML) o anotaciones en los beans de las entidades para poder establecer estas relaciones.

4.5. Herramientas para pruebas

- **JUnit:** Es un framework utilizado para realizar pruebas unitarias a aplicaciones Java, permitiendo realizar estas pruebas de forma controlada y evaluar el funcionamiento de cada uno de los métodos de las clases.

4.6. Herramientas de documentación

- **LaTeX:** Es un software gratuito de composición de textos para la elaboración de documentos de índole científica. Latex proporciona una serie de características que proporcionan una gran calidad tipográfica en sus documentos.

Capítulo III
Desarrollo

5. Primera iteracción: Capa modelo

Esta primera iteracción de desarrollo tiene como objetivo sentar las bases de la aplicación, proporcionando una capa modelo que exponga las funcionalidades necesarias para el acceso a los datos de los diferentes fondos de inversión.

La base de datos utilizada en la aplicación es MySQL junto con el Framework Hibernate.

5.1. Diseño de la base de datos

Para comenzar el diseño de la base de datos debemos analizar cuales son las funcionalidades y las estructuras de datos que necesitamos para la aplicación.

Por un lado, hemos de ser capaces de guardar los principales campos que contiene un fondo de inversión: ISIN del fondo, nombre de la entidad gestora, el tipo de fondo que es, la categoría, la divisa con la que operan sus inversiones y sus comisiones de apertura y cancelación.

Por otro, cada fondo puede tener una cantidad variable de VLs, normalmente uno por día durante el período en el que el fondo se encuentra activo.

Una primera aproximación del Modelo entidad-relación (ER) es el siguiente:

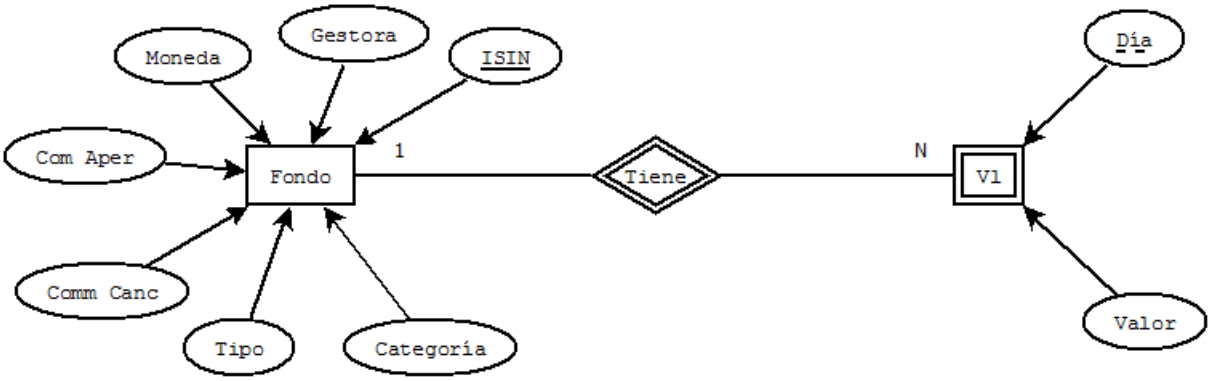


Figura 1: Diagrama ER inicial

Este modelo está compuesto por una entidad fuerte encargada de guardar la información referente a los fondos y una débil, que presenta una dependencia de identificación con la primera, encargada de guardar los VLs de cada fondo. Por tanto, cada fondo puede tener N VLs y cada VL solo puede pertenecer a un fondo.

Este modelo permite almacenar los datos básicos tanto de un fondo como el histórico de sus VLs, pero no permite realizar operaciones de compra/venta de participaciones. Para poder realizar operaciones sobre los diferentes fondos es necesario añadir un poco de complejidad al modelo:

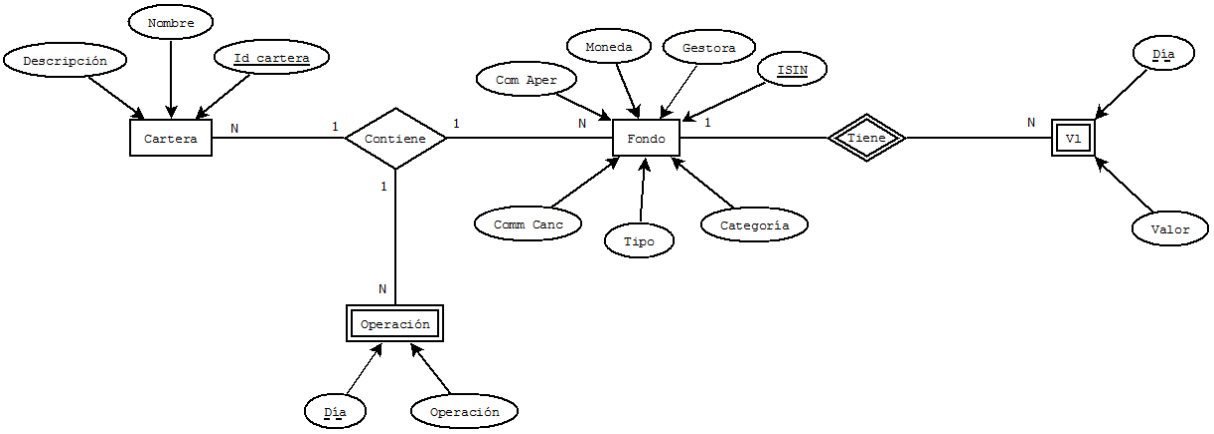


Figura 2: Diagrama ER final

De esta forma incorporamos la posibilidad de tener carteras de fondos y realizar operaciones de compra/venta de participaciones en diferentes fondos pertenecientes a diferentes carteras.

Cada fondo de inversión puede pertenecer a varias carteras distintas y cada cartera estará compuesta por varios fondos. Las operaciones sobre ellos solo estarán permitidas si pertenecen a una cartera de inversión, en caso contrario solo se podrá acceder a su histórico de VLs.

Una vez terminado el diagrama ER el siguiente paso es realizar su traducción? al modelo relacional. El resultado es el siguiente:

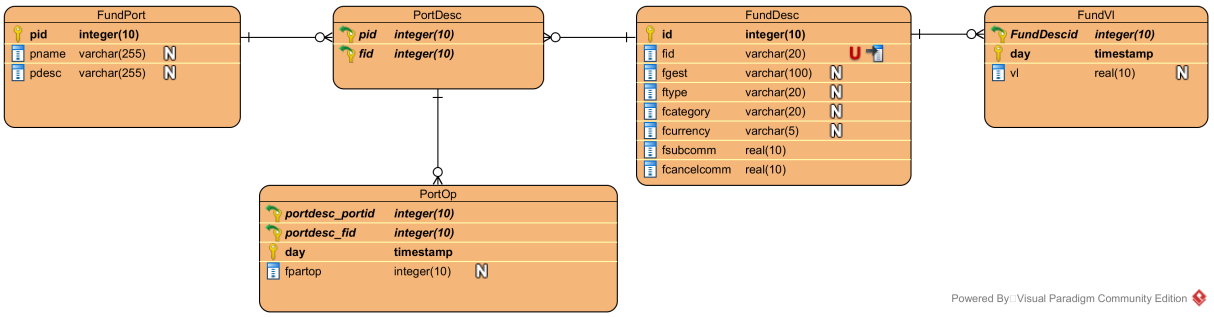


Figura 3: Diagrama del modelo relacional(eliminar los nulos)

5.2. Implementación de la base de datos mediante Hibernate

Para implementar la base de datos usando Hibernate, primero debemos modelar las clases Java correspondientes a las tablas del modelo relacional:

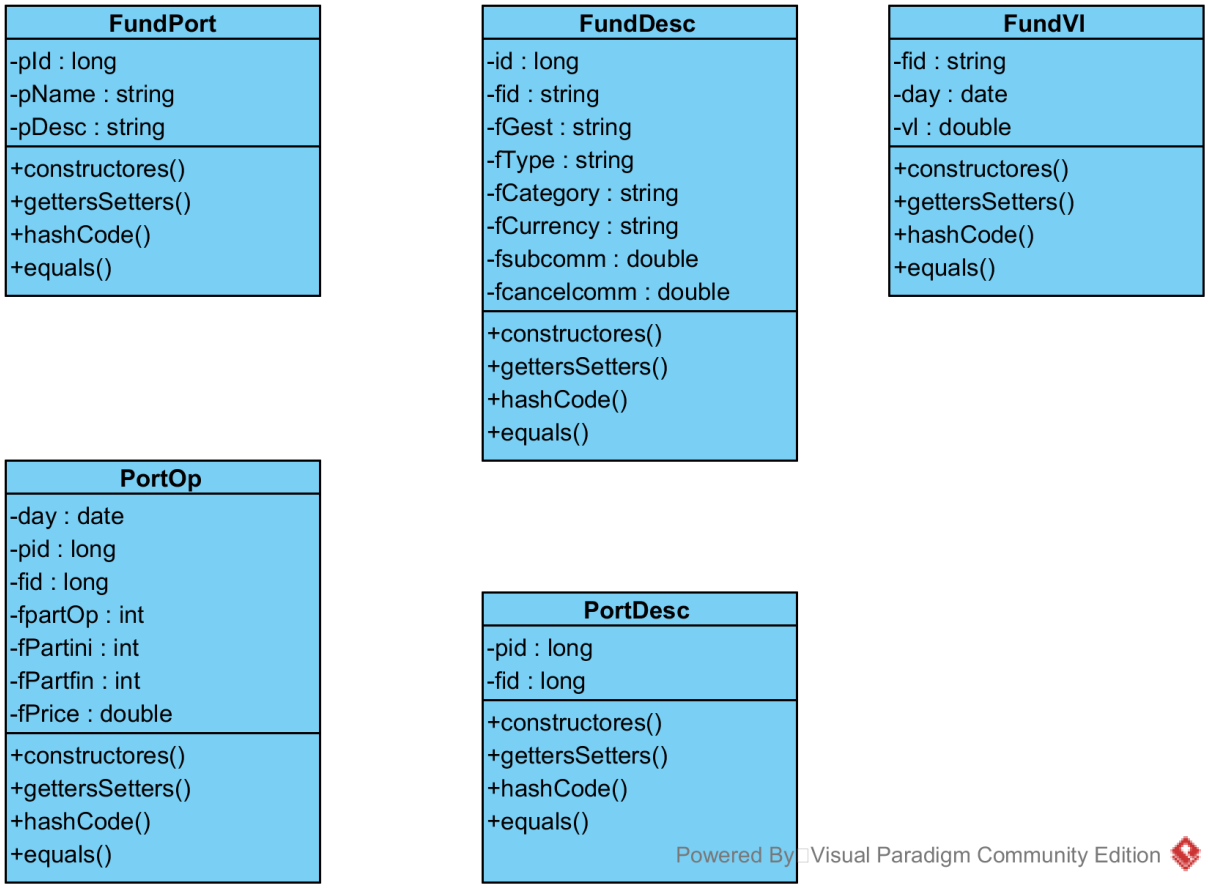


Figura 4: Diagrama de clases del modelo

Los atributos de FundDesc, **id** y de FundPort **pId**, están formados por valores auto generados por la base de datos mediante la estrategia de generación *"GenerationType.IDENTITY"* y actúan como claves primarias de las tablas. En el caso de FunDesc el atributo fId (que se corresponde con el ISIN del fondo) es una clave candidata que se guardará en base de datos como un Varchar de longitud 12, indicando a Hibernate que es un valor único y no puede ser nulo. Este atributo podría haberse utilizado como clave primaria de la tabla FundDesc, pero al tratarse de un tipo de dato Varchar que se repetirá, como norma general, más de un millar veces por cada fondo en la tabla FundVl, se ha decidido utilizar un id autogenerado por motivos de rendimiento.

Para que Hibernate mapee las relaciones es necesario incluir en cada una de las clases con relación 1:N una lista con los objetos relacionados. Por ejemplo, cada objeto FunDesc tiene, a mayores de los atributos mostrados en el UML, uno llamado fundVls y otro llamado portDescs, que contienen una lista de los FundVls y PortDescs relacionados con ese fondo respectivamente.

En el caso de las relaciones N:1 basta con incluir un atributo del tipo del objeto con el que se relaciona. Por ejemplo, cada objeto FundVl tiene un atributo llamado fundDesc que contiene el fondo al que pertenece ese FundVl. Por este motivo, ha sido necesario modificar la implementación por defecto de los métodos equals y hashcode, para que su uso no provocase una llamada recursiva infinita que causase una excepción por stack overflow.

Para poder implementar las clases correspondientes a tablas con claves compuestas en Hibernate, es necesario implementar clases serializables que contengan únicamente los atributos que componen la clave primaria de la tabla. En este caso se han implementado las tres necesarias:

- **FundVlPk:** Con los atributos fundDesc y day.
- **PortDescPk:** Con los atributos funDescId y fundPortId.
- **PortOpPk:** Con los atributos portDesc y day.

En el caso de PortOpPk el atributo portDesc es mapeado en base de datos como el conjunto de las ids del fondo y de la cartera.

A continuación, se definen las anotaciones de Hibernate utilizadas para crear la base de datos.

- **@Id:** Indica que el siguiente atributo es la clave primaria de la tabla.
- @**IdClass**: Indica la clase que contiene los atributos que componen la clave primaria de la clase/tabla actual.
- **@Column:** Indica que el siguiente atributo se modela como una columna en la base de datos.
- **@Transient:** Indica que el siguiente atributo no se modela como una columna en la base de datos.
- **@GeneratedValue:** Indica que el siguiente atributo es generado automáticamente por la base de datos, utilizando la estrategia asignada.
- **@OneToMany:** Indica que el objeto tiene una relación 1:N con otra tabla.
- **@ManyToOne:** Indica que el objeto tiene una relación N:1 con otra tabla.
- **@JoinColumn:** Indica que el objeto obtiene una columna de otra tabla , normalmente se utiliza junto con la anotación anterior.

La anotación **@Transient** se utiliza en la clase PortOp para indicar que los atributos fPartIni, fPartFin y fPrice no se guardan en la base de datos y serán calculados bajo demanda cuando se realice una operación de búsqueda.

Las anotaciones @OneToMany y @ManyToOne tienen algunas opciones interesantes:

- La opción **fetch** indica si las listas de objetos relacionados se cargaran bajo demanda (EAGER) o inmediatamente (LAZY).
- La opción **cascade** hace referencia a las acciones que se deben de llevar a cabo tras la eliminación de un objeto, CascadeType.REMOVE indica que se deben eliminar todos los objetos relacionados con el (Por ejemplo si se elimina un FundDesc se eliminan también todos sus FundVls).
- La opción **mappedBy** se utiliza para definir el atributo que mapea la relación entre las tablas.

El último paso para crear la base de datos consiste en indicar a Hibernate cuales son las clases que debe mapear en su fichero de configuración *"src/main/resources/hibernate.cfg.xml"*.

De esta forma, cuando realicemos la llamada de inicio de sesión *"sessionFactory = new Configuration().configure().buildSessionFactory()"* Hibernate generará las tablas en la base de datos, si no existían previamente.

La propiedad *"hbm2ddl.auto"* nos permite controlar el comportamiento de hibernate cuando se inicia la sesión. Podemos, desde eliminar las tablas y volverlas a crear en cada inicio, hasta simplemente actualizarlas si hay cambios.

5.3. Implementación del servicio del modelo

El servicio del modelo será el encargado de exponer las funcionalidades necesarias para el funcionamiento lógico de la aplicación.

Como funcionalidades mínimas necesitamos tener la capacidad de añadir, buscar, actualizar y eliminar elementos de cada una de las tablas de la base de datos, pero a la hora de realizar gráficas es muy interesante, por ejemplo, la posibilidad de obtener los VLs y las operaciones sobre un fondo en un rango de fechas determinado, por lo que el servicio consta de los siguientes métodos:

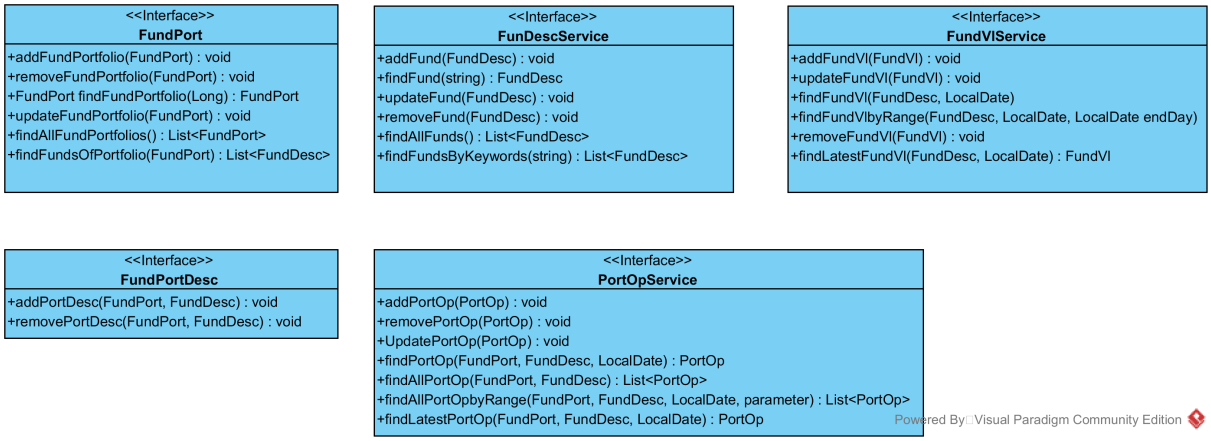


Figura 5: Diagrama del servicio

Aunque en este diagrama existe una interfaz del servicio para cada clase, en la implementación se han agrupado todos en una única interfaz denominada **FundService**.

Cada uno de los métodos del servicio lanza una **RunTimeException** si se produce algun error a la hora de conectarse a la base de datos. A mayores, se han implementado dos excepciones más:

- **InputValidationException:** Salta cuando los datos introducidos por el usuario no son correctos.
- **InstanceNotFoundException:** Salta cuando no se puede encontrar el objeto solicitado en la base de datos.

Para llevar a cabo la validación de los datos de entrada se ha incorporado un validador:

uml del validador

Existen una serie de métodos en el servicio que utilizan este validador para asegurar que los datos introducidos son correctos:

- **validateFundVl:** se encarga de validar que el campo vl del FundVl es mayor o igual que cero.
- **validateFund:** se encarga de validar que el ISIN del fondo es correcto y llama a su vez a validateFundVl para validar cada uno de sus FundVl.
- **validateFundPort:** se encarga de validar que el nombre de la cartera de fondos no sea un string vacío.
- **validatePortOp:** se encarga de validar que la operación añadida, eliminada o actualizada no deje en ningún momento a la base de datos con un total de participaciones negativo.

Ahora analizaremos cada método por separado:

- FundDesc
  - **addFund:** Añade un FundDesc y toda su lista de FundVLs a la base de datos. La forma más común de representación de un fondo es mediante un fichero excel (.xls) que contenga un conjunto de pares día-VL, en un futuro la aplicación permitirá importar los VLs a partir de estos ficheros, por lo que se podrán guardar en la base de datos simplemente llamando a esta operación.
  - **findFund:** Devuelve el FundDesc a partir de su ISIN. A pesar de que la clave primaria de la tabla sea la clave subrogada *"id"*, es más natural buscar un fondo en concreto por su ISIN.
  - **updateFund:** Actualiza los campos de un FundDesc.
  - **removeFund:** Elimina un FundDesc y toda su lista de FundVls de la base de datos.
  - **findAllFunds:** Obtiene todos los FundDesc de la base de datos.
  - **findFundsByKeywords:** Obtiene los fundDesc que coinciden con una serie de caracteres en alguno de sus campos. Esta función es útil a la hora de buscar varios fondos de un mismo tipo, moneda o gestora, por ejemplo.
- FundPort
  - **addFundPortfolio:** Añade una cartera de fondos.
  - **removeFundPortfolio:** Elimina una cartera de fondos.
  - **findFundPortfolio:** Obtiene una cartera de fondos a partir de su Id.
  - **updateFundPortfolio:** Actualiza los campos de una cartera de fondos.
  - **findAllFundPortfolios:** Obtiene todas las carteras de la base de datos.
  - **findFundsOfPortfolio:** Obtiene todos los fondos de una cartera.

- FundVl
  - **addFundVl:** Añade un unico FundVl a un fondo en un día concreto.
  - **removeFundVl:** Elimina una fila de la tabla vl de un fondo en un dia concreto.
  - **findFundVl:**Obtiene el FundVl de un fondo en un dia concreto.
  - **updateFundVl:** Actualiza un unico FundVl de un fondo en un día concreto.
  - **findFundVlByRange:** Obtiene los Vl de un fondo dado en el intervalo de tiempo deseado.
  - **findLatestFundVl:** Obtiene el FundVl del día mas próximo a uno dado (Se comporta exactamente igual a findFundVl si existe un valor vl en ese día). Esta función se utiliza para calcular el valor monetario de una operación. Cuando no se disponga del VL de un fondo el dia de la operación, se utilizará en su lugar el valor del día anterior más cercano.
- PortDesc
  - **addPortDesc:** Añade un fondo a una cartera.
  - **removePortDesc:** Elimina un fondo de una cartera.
- PortOp
  - **addPortOp:** Añade una operacion (Con participaciones como unidad) sobre un fondo en una cartera en un día determinado.
  - **removePortOp:** Elimina una operacion realizada sobre un fondo en una cartera en una fecha.
  - **findPortOp:** Obtiene una operacion sobre un fondo en una cartera en un día determinado.
  - **updatePortOp:** Actualiza un PortOp (Con participaciones como unidad) sobre un fondo en una cartera en un día determinado.
  - **findAllPortOp:** Devuelve todas las operaciones realizadas sobre un fondo en una cartera.
  - **findPortOpByRange:** Devuelve todas las operaciones realizadas sobre un fondo en una cartera entre dos fechas.
  - **findLatestPortOp:** Obtiene el la operacion sobre un fondo en una cartera del dia mas proximo a un dia dado (Se comporta exactamente igual a findPortOp si existe un PortOp en ese dia). Esta función se utiliza para conocer cual ha sido la operación mas reciente sobre un fondo, lo que permite ver las participaciones que tenemos invertidas en el.

Como se ha comentado anteriormente, los campos de los objetos PortOp: fPartIni, fPartFin y fPrice se calculan bajo demanda en los métodos de búsqueda findPortOp, findAllPortOp, findPortOpByRange y findLatestPortOp. La operación encargada de calcularlos es calculatePortOp.

El método **calculatePortOp** revisa todas las operaciones anteriores y obtiene el numero de participaciones inicial y final, además del precio de la operación, utilizando para ello el vl más próximo y teniendo en cuenta las comisiones de apertura y cancelación del fondo.

Se utiliza esta operación en lugar de almacenar directamente los valores de los campos en la base de datos debido a que si se guardasen, cada inserción, actualización o borrado de una operación, implicaría actualizar todos los valores de las operaciones siguientes, además de estar guardando información redundante que se puede calcular directamente con los datos que ya se encuentran en la base de datos.

Capítulo IV
Bibliografía

Referencias

[CNMV] *Comisión Nacional del Mercado de Valores.*
https://www.cnmv.es/

[Rankia] *Todo lo que hay que saber de Fondos de inversión en un único artículo.*
http://www.rankia.com/blog/fondos-inversion/3208096-todo-que-hay-saber-fondos-inversion-unico-articulo

[Rankia] *¿Qué es un fondo de inversión y cómo funciona?.*
http://www.rankia.com/blog/fondos-inversion/952310-que-fondo-inversion-como-funciona

[Wikipedia] *Proceso Unificado de Desarrollo Software.*
https://es.wikipedia.org/wiki/Proceso_unificado

Acrónimos

ER Modelo entidad-relación. 11, 12

IDE Entornos de Desarrollo Integrado. 9
ISIN International Securities Identification Number. 2, 11, 13, 17

POM Project Object Model. 9
PUD Proceso Unificado de Desarrollo Software. 6, 7, 9

TFG Trabajo de Fin de Grado. 7

UML Lenguaje Unificado de Modelado. 6, 9, 13

VL Valor Liquídativo. 2–5, 11, 12, 16–18

XML eXtensible Markup Language. 10