|  |
| --- |
| TP Integrador |
| Documento de cátedra |
| Profesor: Luciano Straccia  Fecha de actualización: 27/10/2017 |

# Tabla de contenidos

[Tabla de contenidos 1](#_Toc496777247)

[TP Previo al Integrador 3](#_Toc496777248)

[TP Integrador: ¿Dónde invierto? 4](#_Toc496777249)

[Presentación general 4](#_Toc496777250)

[Contexto Inicial 6](#_Toc496777251)

[Dominio 7](#_Toc496777252)

[Cuentas 7](#_Toc496777253)

[Indicadores 7](#_Toc496777254)

[Metodologías 8](#_Toc496777255)

[Requerimientos generales 9](#_Toc496777256)

[Metodología 9](#_Toc496777257)

[Entrega 1 10](#_Toc496777258)

[Entrega 2: Indicadores 11](#_Toc496777259)

[Entrega 3: Metodologías 12](#_Toc496777260)

[HITO DE CONTROL – CIERRE DE PRIMER CUATRIMESTRE 14](#_Toc496777261)

[Entrega 4: Persistencia 15](#_Toc496777262)

[Entrega 5: UI Web 17](#_Toc496777263)

[Entrega 6: Despliegue y Cambios Arquitecturales 19](#_Toc496777264)

[Carga batch de cuentas 19](#_Toc496777265)

[Precálculo de indicadores 19](#_Toc496777266)

[Persistencia de indicadores 20](#_Toc496777267)

[En resumen: 20](#_Toc496777268)

# TP Previo al Integrador

Se deben diseñar e implementar los siguientes módulos de un software:

* Login para acceso a la aplicación y cambio de contraseña (excluir el registro de nuevo usuario)
* Log (reporte de auditoría) de acceso
* Pantalla en que se solicite un país y a partir de este se soliciten provincias/estados (el listado de países y provincias o estados deben ser obtenidos del listado brindado por la API de Mercado Libre[[1]](#footnote-1)).

Considerando que:

* Debe ser una aplicación web
* Debe utilizar el patrón MVC
* Persistir el log de acceso en un archivo JSON
* Persistir la información relativa al login y selección de país/provincia/estado en un Sistema de Gestión de Base de Datos

Se solicita:

* Requerimientos funcionales y no funcionales
* Requerimientos de reporte
* Requerimientos de interfaz
* Mockup UI
* Modelo de casos de uso
* Modelo de objetos
* Modelo de datos relacional
* Casos de prueba
* Implementación del diseño. La tecnología a utilizar quedará a criterio de cada grupo de trabajo debiendo ser autorizada previamente por el equipo docente.

# TP Integrador: ¿Dónde invierto?

## Presentación general

El mundo de las inversiones en la bolsa es complejo: contamos con cientos de datos históricos para miles de empresas, que podemos analizar empleando decenas de metodologías. Y todo esto para resolver una única y simple pregunta: ¿en qué empresa me conviene invertir mi dinero?

Este trabajo práctico se trata justamente de buscar con ayuda del software una respuesta aproximada a esta pregunta: diseñaremos, construiremos y probaremos un sistema que nos permita, recopilar, ordenar y analizar los balances públicos de empresas del mercado de Estados Unidos[[2]](#footnote-2).

El sistema lo construiremos iterativa e incrementalmente: partiremos de una simple aplicación de escritorio, sin persistencia, que nos servirá para entender el dominio y casos de uso, e iremos agregando complejidad tanto de modelo como arquitectónica. El resultado final será un sistema distribuido, con interfaz Web y persistencia políglota.

El recorrido que haremos no se corresponderá necesariamente con el de un desarrollo industrial, sino que estará pensado para acompañar los temas de la cursada. De todas formas, será ilustrativo de muchas decisiones de diseño realistas.

Vale aclarar que el entregable final, dadas varias simplificaciones que se harán con fines didácticos, NO servirá como un analizador de inversiones fiable, pero sí se asemeja a uno real y sus ideas podrán ser utilizadas para que el estudiante curioso explore y, con tiempo y dedicación suficiente, lo convierta en un software útil. Dicho en otras palabras: no hagan a su abuelo comprar acciones en función de los resultados de nuestro sistema.

A lo largo del proceso, pondremos en práctica los siguientes conocimientos:

* Cualidades de Diseño / Atributos de Calidad
* Arquitectura Modelo-Vista-Controlador (MVC)
* Interfaces Web y de escritorio
* Adaptación de interfaces entrantes y salientes
* Estrategias de persistencia relaciones y no relacionales
* Mapeo objetos-relacional
* Diseño avanzados con objetos
* Patrones de diseño orientados a objetos: Adapter, Builder, Observer, Command, State, Strategy, Template Method
* Testing y mocking
* Componentes arquitecturales: bases de datos, balanceadores de carga, cachés, servidores de aplicaciones, procesos calendarizados, colas de mensajes

Esperamos que durante el desarrollo del trabajo práctico puedas:

* Modelar situaciones de dominio complejas utilizando el paradigma de objetos
* Establecer un mapeo bidireccional entre el modelo de objetos y un modelo de persistencia relacional
* Tomar decisiones arquitecturales:
  + Determinar cuándo y cómo adaptar interfaces entrantes y salientes
  + Determinar cuándo y cómo es deseable incorporar una base de datos no relacional
  + Determinar qué procesos serán on-line y qué procesos se ejecutarán de modo batch
  + Determinar los principales componentes arquitecturales y su forma de comunicación

Finalmente, y antes de indagar en los aspectos técnicos del problema, vale siempre recordar que la primera decisión de un Ingeniero en Sistemas es ética: no deberíamos diseñar, construir, mantener y/ sistemas que no estén alineados con nuestros valores éticos. Decenas de dominios y modelos de negocios presentarán para muchas personas dilemas éticos, y el de las inversiones en bolsa no es una excepción. De estar ante la decisión de trabajar en un sistema de estas características en un contexto no académico, deberíamos antes plantearnos cuál es nuestra posición al respecto. El entendimiento del modelo al que este trabajo práctico nos introducirá, nos ayudará, no obstante, a tomar una decisión más fundamentada.

## Contexto Inicial

Es Martes, 28 de febrero de 2017, y el calor al mediodía en la casa de Villa Marteli es agobiante. Sólo se escuchan las chicharras del jardín, el cooler de la computadora y a Héctor tipeando frenéticamente desde temprano, describiendo todo lo que sabe sobre el mundo de las acciones en la bolsa de Estados Unidos. ¿Por qué lo hace?

Héctor, de 48 años, empezó a investigar sobre este tema algunos años atrás, por pura curiosidad. Con el tiempo, pasó de ser un mero hobby, a un ingreso extra ocasional: en su tiempo libre, asesora a algunos amigos y conocidos que hacen pequeñas inversiones. Sin embargo, las cosas están cambiando en su vida: la empresa en donde trabaja ha empezado a recortar personal, y ante la amenaza de quedarse sin empleo está evaluando dedicarse tiempo completo a la actividad de asesoramiento.

Pero hay un problema: Héctor recopila y procesa toda la información de forma mayormente manual. Consulta sitios en internet que proveen los datos que necesita, los carga en una planilla de cálculo, y realiza su análisis en parte apoyándose en fórmulas que él mismo cargó, y en parte con papel, lápiz y calculadora. Esto es tedioso y le demanda mucho tiempo, y sabe que le impide escalar su negocio.

Para colmo de males, como siempre está aprendiendo sobre la materia, es frecuente que decida modificar algunas de las fórmulas que usa, agregue nuevas o desestime otras que no le fueron útiles en sus análisis, lo que en más de una ocasión introdujo errores en sus análisis que le significaron pérdidas importantes de tiempo. Afortunadamente siempre se dió cuenta del error antes de dar un mal consejo a sus clientes, pero necesita dar una solución a esta cuestión.

Charlando el último domingo durante una cena familiar, su sobrina Sabrina, ingeniera recientemente recibida pero ya con algunos años de experiencia en el desarrollo de software, lo convenció de automatizar, en parte, lo que hace: construir una aplicación de escritorio simple que le permita cargar la información actualizada sobre las empresas que analiza, los cálculos que realiza y mostrar sus resultados. En definitiva, construir un (muy simple) Sistema de Apoyo a Decisiones. Ella se ofreció a hacerlo, pero para empezar, le pidió le envíe por correo electrónico el dominio con el mayor nivel de detalle posible.

Y esto nos trae nuevamente al 28 de febrero, día en que Héctor se encuentra escribiendo el documento que está a continuación...

## Dominio

### Cuentas

Periódicamente (típicamente cada año o cada semestre), las empresas que cotizan en bolsa generan balances públicos, que hablan del estado la empresa, expresados en cuentas.

|  |
| --- |
| **Ejemplo:** estas son algunas cuentas:   * [EBITDA](http://www.investopedia.com/terms/e/ebitda.asp) * [FDS](http://www.investopedia.com/terms/f/fullydilutedshares.asp) * [Free Cash Flow](http://www.investopedia.com/terms/f/freecashflow.asp) * [Ingreso neto en operaciones discontinuas](http://www.investopedia.com/terms/d/discontinued-operations.asp?lgl=no-infinite) e [Ingreso neto en operaciones continuas](http://www.investopedia.com/terms/c/continuingoperations.asp) |

Existen cientos de cuentas posibles, cada una con su propia fórmula de cálculo. Sin embargo, **para el inversionista poco importa calcularlas**, dado que existen bases de datos públicas que ya nos muestran sus valores en cada período.

|  |
| --- |
| **Ejemplo:** Facebook en el año 2016 la cuenta EBITDA tuvo valor de 14870 millones de dólares, y en año 2015, 8162 millones de dólares. Para el análisis que vamos a hacer el cálculo exacto de esta cuenta no es relevante, sino sólo su valor en cada año. |

### Indicadores

Si bien los datos de cuentas son la base de cualquier análisis, vamos a trabajar en su lugar con información más refinada. Para eso utilizaremos **indicadores**, que son cálculos (simples, típicamente sólo algebraicos) sobre las cuentas anteriores, que son útiles para el inversionista pero no siempre son publicados directamente por las compañías.

|  |
| --- |
| **Ejemplo:** un indicador es el ingreso neto, y puede calcularse como la suma de dos cuentas: |

Los indicadores pueden ser aún más interesantes: pueden ser cálculos que combinen cuentas y otros indicadores.

|  |
| --- |
| **Ejemplo:** el [retorno sobre el capital total](http://www.investopedia.com/terms/r/returnoninvestmentcapital.asp) (ROE), utiliza a otro indicador: el ingreso neto visto anteriormente. |

Al igual que las cuentas, un indicador puede arrojarnos diferentes valores en cada período para una cierta empresa, por eso es que nos sirven para evaluarla año a año.

|  |
| --- |
| **Ejemplo:** en 2016, las cuentas de Facebook arrojaron los siguientes resultados:   * Ingreso Neto En Operaciones Continuas: 4.273 millones de dólares * Ingreso Neto en Operaciones Discontinuadas: 0   Por tanto, el indicador Ingreso Neto (la suma de las dos cuentas anteriores) arrojó 4.273 millones de dólares |

### Metodologías

Finalmente, los inversionistas, en base a su experiencia y estudio, eligen (o incluso, crean) ciertos indicadores que les resultan significativos, y les imponen ciertas condiciones, que de cumplirse, significa que es deseable invertir en la empresa. Al conjunto de estos indicadores y condiciones se los conoce como metodologías (o métodos) de inversión.

|  |
| --- |
| Ejemplo: si bien [la metodología de Warren Buffet](http://www.investopedia.com/articles/01/071801.asp) no ha sido totalmente sistematizada, una posible interpretación nos dice que se compone de 6 indicadores, de los cuales tres son los siguientes:   * Que el ROE sea consistente durante todos los períodos anteriores * Que la tasa .... sea mayor a uno * Que las ... sea creciente |

## Requerimientos generales

Entendido el dominio y el contexto, repasemos y formalicemos a los requerimientos funcionales y no funcionales del sistema:

R1. Tendrá un único usuario, Héctor, en su rol de analista de inversiones.

R2. Será usada desde una computadora de escritorio

R3. Deberá permitir

* 1. la carga de cuentas de empresas, por período
  2. la carga de indicadores creados por el analista
  3. para una empresa y un período, la consulta de los valores
     1. de cuentas
     2. e indicadores: se podrán encontrar tanto indicadores creados por el analista, como predefinidos.
  4. para una empresa y un período, la visualización de gráficos comparativos de indicadores en el tiempo
  5. la carga de metodologías creadas por el analista
  6. el análisis de una empresa aplicando metodologías
  7. el guardado de metodologías e indicadores creados por el analista

## Metodología

Tras reunirse nuevamente Héctor y Sabrina, decidieron que encararán el proyecto de forma iterativa e incremental. ¿Por qué? Porque por un lado, Héctor necesita algo funcionando lo antes posible, y por otro lado, ambos necesitan validar sus ideas sobre qué debe hacer el sistema.

Por otro lado, además de tener en cada entrega algo funcional, esta metodología de desarrollo le dará a Sabrina la posibilidad de en cada iteración, comprender mejor el dominio a medida que diseña su modelo, y a Héctor, la oportunidad de formalizar y sistematizar lo que sabe su proceso de análisis.

## Entrega 1

Para la primera iteración, es necesario elegir la tecnología para la realización del proyecto, decisión arquitectural de gran impacto en el desarrollo del sistema. Parte fundamental de esta entrega es la elección de una tecnología y su prueba.

Además, es necesario proveer a Héctor una interfaz gráfica que sirva para verificar los principales requerimientos del sistema. El objetivo es hacerlo lo más rápido posible, para encontrar inconsistencias lo antes posible.

Se pide:

1. Realizar un diagrama de casos de uso que contemple todos los requerimientos descritos en “Requerimientos generales”
2. Realizar los bocetos para las pantallas correspondientes identificadas en el diagrama de casos de uso. [[3]](#footnote-3)
3. Diseñar[[4]](#footnote-4) e implementar los siguientes requerimientos:
   1. Carga de cuentas (Requerimiento R3.a). Se sabe que Héctor cargará los datos de empresas consultando páginas de Internet. Definitivamente sería una gran mejora si esta carga de datos se realizará de forma automatizada. Sin embargo, para mantener esta iteración simple y dado que Héctor ya estaba haciendo este proceso de carga de forma manual, se cargarán los datos de cuentas de las empresas desde un archivo.[[5]](#footnote-5)
   2. Consulta de valores de cuentas (Requerimiento R3.c.i.)

## Entrega 2: Indicadores

En la segunda iteración ampliaremos nuestra aplicación, para dar soporte a los *indicadores*. Recordemos que los indicadores pueden ser tanto:

* ***predefinidos***, es decir, estar programados dentro de nuestra aplicación. Esto sirve para los indicadores más simples y cuya definición no cambia, como por ejemplo ***Ingreso Neto*** Estos indicadores son pocos comparados con los indicadores definidos por el usuario.
* ***definidos por el usuario***:son indicadores que el usuario de alguna forma ingresará en el sistema. Los nuevos indicadores definidos por el usuario pueden ser utilizados directamente o ser guardados para un uso posterior.

Se pide:

1. Diseñar, implementar e incorporar al modelo de dominio los indicadores, de forma que puedan ser tanto cargados por el analista de inversiones como soportados nativamente por la aplicación. En cualquier caso, debe ser fácil agregar nuevos indicadores, y debe ser posible evaluarlos contra cualquier empresa en cualquier año. Es importante tener en cuenta que:

|  |
| --- |
| **Sugerencia**: Una forma usual de permitir que el usuario ingrese cálculos matemáticos simples es mediante un lenguaje específico de dominio (DSL), como lo son por ejemplo las expresiones que podemos encontrar en una hoja de cálculo como Google Docs Spreadsheet, Apache OpenOffice Calc o Microsoft Excel.  Eso implica *parsear* strings. Si bien es algo que se puede hacer (con mucho esfuerzo a mano), existen herramientas que simplifican mucho esto, como por ejemplo JavaCC o Antlr. Recomendamos utilizar alguna de estas herramientas. |
| **Advertencia**: si se utiliza un DSL para introducir indicadores, es necesario contemplar que el usuario con seguridad ingresará fórmulas incorrectas ocasionalmente, ya sea por sintaxis inválida o por emplear cuentas u otros indicadores en ellas que no existan. El sistema debe manejar adecuadamente estos ingresos incorrectos. |

1. Extender las vistas para poder soportar:
   1. la carga de indicadores definidos por el usuario
   2. el guardado y recuperación de indicadores definidos por el usuario
   3. listar a los indicadores junto con las cuentas de una empresa en un cierto período.

## Entrega 3: Metodologías

Llegó finalmente el momento de implementar las metodologías, así que el plato fuerte de esta iteración es su diseño y desarrollo.

Se pide:

1. Diseñar e implementar las condiciones y metodologías, de forma que las 4 primeras condiciones de metodología de Buffet[[6]](#footnote-6) puedan ser evaluadas, y que sea fácil definir metodologías personalizadas similares.

|  |
| --- |
| **Sugerencia**: A la hora de diseñar las condiciones, prestar atención a que:   * Hay algunas que son taxativas (*en esta empresa se puede invertir, en esta no*), otras que sirven para priorizar (*en esta empresa conviene invertir más que en otra)*, y algunas que combinan ambas ideas * Revisar las condiciones de ejemplo e identificar cuáles son primitivas y cuales son derivadas de las anteriores. Esto te ayudará a la hora de modelarlas. |

|  |
| --- |
| **Advertencia**: Releer con cuidado la especificación de las metodologías. Puede que haya aspectos no definidos o ambiguos que tengas que discutir con tu docente. |

1. Diseñar e implementar un mecanismo para que el analista pueda definir sus propias metodologías desde la interfaz gráfica.

|  |
| --- |
| **Advertencia**: No es posible implementar una funcionalidad que le permita al analista definir cualquier tipo de metodología, dado que siempre pueden surgir cierto tipo de análisis complejos no contemplados por la herramienta que estamos diseñando. Parte de esta entrega es encontrar, con ayuda de tu docente, una solución de compromiso entre contar con un alta de metodologías flexible pero realizable en el tiempo del trabajo práctico. |

1. Diseñar e implementar las pantallas necesarias para que un analista seleccione una metodología (ya sea predefinida o cargada por el usuario) y la ejecute contra todas las empresas cargadas en el sistema. El resultado debe ser un listado ordenado de empresas en las que es deseable invertir
2. Definir y generar los casos de prueba para dar una cobertura adecuada a las entregas realizadas hasta el momento.[[7]](#footnote-7)
3. Probar de forma automática el proceso de carga de archivo de la primera entrega[[8]](#footnote-8)
4. Diagramar la arquitectura del sistema[[9]](#footnote-9)

## HITO DE CONTROL – CIERRE DE PRIMER CUATRIMESTRE

Deben entregar carpeta conteniendo (en un único documento, con carátula, tabla de contenidos, encabezado, número de página):

* Diagrama de Casos de Uso
* Modelado de clases
* Modelado de datos (si ya lo tuvieran)
* Modelado de arquitectura (Diagrama de Componentes y Diagrama de Despliegue)
* Casos de prueba

## Entrega 4: Persistencia

Llegó finalmente el momento de implementar las metodologías, así que el plato fuerte de esta iteración es su diseño y desarrollo.

A la vuelta de las vacaciones de Sabrina, Héctor le cuenta que en estas cuatro semanas las cosas cambiaron mucho.

Primero, porque ahora el mismo Héctor tiene serias dudas sobre su idea original: llegó a la conclusión de que nada es tan fácil como parece, que el mundo de la bolsa es demasiado competitivo y aún hay aspectos que no domina como para poder dar asesoramiento a más personas.

Segundo, porque charlando con un amigo, éste opinó que de todas formas el software que habían creado junto a Sabrina era una herramienta interesante para pequeños inversores, y que él consideraba que muchos estarían dispuestos a pagar por éste.

Todo esto lo llevó a Héctor a plantear un nuevo modelo de negocio para su incipiente emprendimiento: en lugar de asesorar a clientes, proveer directamente un software para quienes sí lo hacen. Y tras varias reuniones más llegó a Valeria, empresaria que está dispuesta a ayudar económicamente a desarrollar este nuevo producto.

Por teléfono, Héctor le acaba de preguntar a Sabrina si ella quiere ser la arquitecta de un nuevo sistema, esta vez Web y con soporte para múltiples usuarios, basándose en el anterior. Y tras pensarlo un poco, no sólo Sabrina dice que está dispuesta a hacerlo, sino que además, considera que puede reutilizar mucho del viejo sistema.

Hasta el momento persistíamos los datos en archivos. El objetivo de esta entrega consiste en adaptar esta arquitectura para centralizar el almacenamiento. Es decir, necesitamos permitir almacenar los elementos del dominio en un medio relacional, no siendo necesario mantener la persistencia en archivos. La única excepción serán las cuentas, que por ahora seguiremos cargando tanto desde la base de datos como desde un archivo.

Además, no queremos acoplarnos a una base de datos particular, y para poder reutilizar el modelo de objetos ya construido, se pide el uso de un ORM. Si bien parece fácil, ¡cuidado!, puede que haya que realizar modificaciones al modelo original.

|  |
| --- |
| **Sugerencia**:  De utilizar Java 8 o posterior, recomendamos utilizar JPA/Hibernate (es decir, Hibernate a través de la interfaz estándar de JPA). Un buen punto de partida para adaptar el proyecto es el siguiente: <https://github.com/dds-utn/jpa-proof-of-concept-template>  De utilizar .NET, recomendamos utilizar Entity Framework, cuyos ejemplos trabajaremos en clase. |

Se pide:

1. Introducir en la solución un motor de base de datos relacional
2. Introducir en la solución un motor ORM
3. Persistir en dicha base de datos las siguientes entidades:
   1. Cuentas
   2. Indicadores
   3. Metodologías
   4. Empresas
4. Lograr que la aplicación siga funcionando sin cambios funcionales
5. Lograr que la aplicación pueda seguir cargando cuentas desde archivos. No es necesario desarrollar nuevas pantallas, sino sólo continuar soportando esta carga programáticamente.

## Entrega 5: UI Web

¡Llegó el turno de implementar las nuevas vistas! Para esta entrega, nuestro requerimiento es simple: necesitaremos implementar las pantallas Web necesarias para todas las funcionalidades existentes:

* Visualización de cuentas
* Creación de indicadores
* Evaluación de indicadores
* Creación de metodologías
* Evaluación de metodologías

Por último, hay que tener en cuenta que ahora que el almacenamiento estará centralizado y la aplicación será accedida por múltiples analistas, las metodologías e indicadores serán propias de cada usuario.

Por ejemplo:

* si Hernán crea un indicador A y B;
* y Anabel crea indicadores C y D;
* Entonces Hernán sólo verá los indicadores A y B, y Anabel, C y D.

|  |
| --- |
| **Sugerencia**: En el marco de la presentación Web, para construir componentes visuales consistentes es necesario un grado considerable de experiencia en maquetado con HTML y CSS que excede los tiempos acotados del TP Anual. Por eso, recomendamos ampliamente el uso de frameworks como Bootstrap (<http://getbootstrap.com/>), que proveen componentes visuales y sistemas disposición de componentes que cubren los requerimientos planteados. |

Se pide:

1. Implementar las vistas empleando una interfaz Web.
2. Incorporar a la aplicación el concepto de login. **No** es necesario implementar:
   1. un flujo de registración, sino que bastará con que sea posible cargar los usuarios a través de la base de datos;
   2. una política segura de control de contraseñas
   3. login con redes sociales
3. Describir mediante diagramas y prosa la arquitectura física y lógica del nuevo sistema, haciendo foco en los siguientes aspectos:
   1. Nodos de red desplegados
   2. Motores de persistencia
   3. Interfaces con sistemas internos y externos
   4. Protocolos de red utilizados
   5. Principales componentes lógicos de alto nivel

Indicar además qué acciones son necesarias para escalar horizontalmente a la nueva arquitectura.

## Entrega 6: Despliegue y Cambios Arquitecturales

¡Vamos que ya se termina! Para la sexta y última iteración, vamos a encarar algunos cambios arquitecturales: carga *batch* (en lotes) de cuentas y precálculo de indicadores.

### Carga batch de cuentas

En las primeras etapas la carga de cuentas fue disparada por los usuarios de la aplicación, que ante demanda abrían archivos de cuentas, y trabajaban con estos datos. Sin embargo, en la quinta iteración, que introdujo una interfaz Web y la posibilidad de manejar múltiples usuarios, no se contempló esta posibilidad, dado que serán los administradores del sistema quienes definan cuáles cuentas se exhibirán en la aplicación.

Aunque inicialmente Sabrina pensó en cargar manualmente la información de cuentas en la base, mediante consultas SQL, más tarde se comprendió que regularmente (al menos una vez por mes) es necesario actualizar estos datos, ya sea con correcciones, agregado de nuevas empresas o la incorporación de nuevos períodos.

Por eso, se decidió volver a incorporar una funcionalidad de carga y actualización de cuentas en el sistema, basada en el antiguo formato de archivo de cuentas de las primeras iteraciones. Sin embargo, la arquitectura será diferente: los administradores del sistema podrán dejar en un directorio especial un archivo de cuentas, y el sistema deberá procesar las mismas, procurando:

1. agregar a las cuentas existentes toda información de empresas y períodos nuevos que se encuentre en el archivo;
2. modificar las cuentas existentes si el archivo contiene correcciones;
3. no procesar dos veces el mismo archivo.

### Precálculo de indicadores

Tras unas primeras pruebas con cientos de empresas, cada una con decenas de cuentas diferentes, se llegó a la conclusión que el cálculo de los indicadores están tomando demasiado tiempo. La situación se agrava porque con la cada vez mayor cantidad de usuarios, se agregan sistemáticamente metodologías e indicadores, los cuales ralentizan la carga de varias pantallas.

Por eso, nos han pedido que precalculemos los valores de los indicadores para cada empresa y período, de forma que no haya necesidad de calcularlos dos veces mientras la información de cuentas no cambie. Estos datos precalculados deben ser persistidos apropiadamente; la elección del medio de persistencia queda a criterio del equipo. Se sugiere el uso de persistencia no relacional.

### Persistencia de indicadores

Considerando las funcionalidades ya existentes de entregas anteriores y para validar su correcto funcionamiento, se pide explicar cómo resolvieron la persistencia de indicadores en cuanto a su fórmula.

Por ejemplo, si el indicador fuera AGH + 4 / ROI, siendo AGH y ROI otros indicadores ¿persistieron esto como string? ¿o utilizaron los id de los indicadores? Por ejemplo... si el id de AGH es 3 y el id de ROI es 1, algo así como @3 + 4 / @1 (este ejemplo supone que estandarizaron que el hallazgo de @ en la fórmula implica la búsqueda del id)

Si utilizaron la persistencia en formato string, ¿cómo resuelven la posibilidad que el usuario modifique el nombre del indicador AGH y arroje luego un error al ingresar al indicador que contiene en su fórmula a este indicador que anteriormente se denominaba AGH?

**En resumen:**

Se pide:

1. Implementar los nuevos requerimientos
   1. Carga batch de cuentas
   2. Precálculo de indicadores
2. Explicar cómo resolvieron la persistencia de indicadores
3. Desplegar la aplicación final en un servidor en Internet.

|  |
| --- |
| **Alternativas**: existen múltiples proveedores de hosting de aplicaciones en internet, tanto PaaS como IaaS. Algunas opciones, a criterio del equipo, son:   * Heroku * Google Cloud Platform * Open Shift * Amazon Web Services * Digital Ocean * Azure   Varios de estos servicios ofrecen planes gratuitos para estudiantes. Ver [*Github Students Pack*](https://education.github.com/pack).  De encontrar dificultades para desplegar en alguno de estos proveedores, se aceptará también desplegar la aplicación en alguna PC o notebook que funcione como servidor.  Nota:  No todos los PaaS aceptan leer/escribir fácilmente en su sistema de archivos (por ejemplo Heroku), en caso de utilizar dicho servicio, deberá presentar a su docente una opción alternativa para la carga de cuentas. |

1. http://developers.mercadolibre.com/es/api-docs-es/ [↑](#footnote-ref-1)
2. ¿Por qué no utilizar los datos del mercado Argentino? Simplemente porque existe mucha más información fácil de obtener del primero que del segundo. [↑](#footnote-ref-2)
3. Las maquetas en papel son excelentes formas de validación de prototipos, por ser muy rápidas de construir, pero tienen la contra de no ser reutilizables y ser mucho menos interactivas. En esta entrega sugerimos realizar maquetas en papel (más adelante veremos técnicas específicas, pero por el momento pueden realizarlo de la forma que deseen) y al menos construir una pantalla utilizando tecnología de escritorio o web, a fin de que sirva como prueba tecnológica. [↑](#footnote-ref-3)
4. Recordar que diseñar implica, por ejemplo, realizar el modelo de objetos (Diagrama de Clases) [↑](#footnote-ref-4)
5. El resultado de este proceso de carga debe dejar a las cuentas de todas las empresas en el ambiente de objetos (en otras palabras: en memoria) [↑](#footnote-ref-5)
6. http://www.investopedia.com/articles/01/071801.asp [↑](#footnote-ref-6)
7. Aun no vimos este tema de diseño de casos de prueba. Si llegamos a verlo se incluye en la entrega, sino queda para entrega 3 [↑](#footnote-ref-7)
8. Originalmente previsto en Entrega 1, fue postergado a Entrega 3 [↑](#footnote-ref-8)
9. Se trata de una primera aproximación. Durante la cursada discutiremos aspectos asociados a arquitectura de sistemas, patrones específicos y podrán refinar (o redefinir) su arquitectura. Originalmente previsto en Entrega 1, fue postergado a Entrega 3. [↑](#footnote-ref-9)