BVC-MÓVIL

Bolsa de Valores de Colombia Móvil

VISTAS ARQUITECTÓNICAS

Modelos de base para la construcción del Documento de Arquitectura

**Equipo de Trabajo**

Hans Linderman Parra Padilla

Juan David Cuartas León

Mario Andrés Galeano Montoya

**Asesor**

Juan Bernardo Quintero

Historia

|  |  |  |
| --- | --- | --- |
| Versión | Fecha | Descripción de Cambios |
| 1.0 | 6 de Abril de 2013 | Vistas arquitectónicas |
| 1.1 | 10 de Mayo de 2013 | Panorama, drivers y estilos arquitectónicos. |
|  |  |  |
|  |  |  |
|  |  |  |

CONTENIDO

**Pág.**

[1. Descripción del Documento 3](#_Toc356221245)

[1.1 Propósito 3](#_Toc356221246)

[1.2 Contenido 3](#_Toc356221247)

[1.3 Audiencia 3](#_Toc356221248)

[1.4 Definiciones y acrónimos 4](#_Toc356221249)

[1.4.1 Terminología técnica 4](#_Toc356221250)

[1.5 Terminología de negocio 4](#_Toc356221251)

[2. Panorama del Proyecto 6](#_Toc356221252)

[2.1 Contexto del negocio 6](#_Toc356221253)

[2.2 Contexto del sistema 6](#_Toc356221254)

[3. Drivers de la Arquitectura 7](#_Toc356221255)

[3.1 Restricciones de negocio 7](#_Toc356221256)

[3.2 Restricciones técnicas 8](#_Toc356221257)

[4. Estilos Arquitectónicos Usados 10](#_Toc356221258)

[5. Vista Conceptual 12](#_Toc356221259)

[6. Vista Lógica 19](#_Toc356221260)

[7. Vista Física 21](#_Toc356221261)

[7.1 Descripción de la plataforma tecnológica 21](#_Toc356221262)

[7.2 Consideraciones de implementación de los componentes a construir 23](#_Toc356221263)

[7.3 Consideraciones de componentes a reutilizar 27](#_Toc356221264)

[8. Vista de Implementación 28](#_Toc356221265)

[8.1 Interfaces y Operaciones de la capa Application Layer 32](#_Toc356221266)

[8.2 Interfaces y Operaciones de la capa Domain Model Layer 36](#_Toc356221267)

[8.3 Interfaces y Operaciones de la capa Infraestructure Layer 38](#_Toc356221268)

[8.4 Interfaces y Operaciones de la capa Utility Layer 40](#_Toc356221269)

[9. Tamaño y Rendimiento 42](#_Toc356221270)

[10. Calidad 43](#_Toc356221271)

# Descripción del Documento

## Propósito

El propósito de este documento es tener un consenso entre los miembros del equipo de trabajo en el diseño de la arquitectura de software de la aplicación **BVC-MÓVIL – Bolsa de Valores de Colombia Móvil**. La arquitectura planteda en este documento se basa en 4 vistas que defienen aspectos en los sigueintes frentes:

* Vista Conceptual: visión que los usuarios tienen de la aplicación.
* Vista Lógica: visión desde los principales elementos y principios del diseño.
* Vista Física: visión desde la distribución del procesamiento entre los dispositivos.
* Vista de Implementación: visión que muestra cómo serán montados los diferentes componentes de la aplicación y la forma en que interactúan.

## Contenido

Este documento consta de 10 secciones:

* Las secciones 1 y 2 describen este documento y el proyecto.
* Las secciones 3 y 4 muestran los drivers y los estilos arquitectónicos usados.
* De la sección 5 a la 8 se ilustran las vistas arquitectónicas de **BVC-MÓVIL**.
* Las secciones 9 y 10 muestran las condiciones de tamaño, rendimiento y calidad.

## Audiencia

Este documento se crea para los miembros del equipo de la compañía **Apps & Services**. La audiencia objetivo de este documento incluye el lider del proyecto, el arquitecto, los desarrolladores y los analistas de infraestructura.

## Definiciones y acrónimos

### Terminología técnica

* **DAO**: Data Access Object
* **DDD**: Domain Driven Design
* **JSF**: Java Server Faces
* **SOA**: Software Oriented Architecture
* **SOAP**: Simple Object Acces Protocol
* **XML**: Document Object Model

## Terminología de negocio

**Acción**: Una [acción](file:///L:\\Personal%202\\Especializaci%C3%B3n\\Proyecto%20integrador\\1%20-%20Requisitos\\Requisitos_REM_Default_Spanish_CRS.html" \l "GLO-0014" \o "Una acción es la parte alícuota del capital social de una sociedad anónima. Cabe resaltar que poseer acciones de una compañía confiere legitimidad al accionista para exigir sus derechos, pero a su vez para cumplir con sus obligaciones.) es la parte alícuota del [capital](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0009) social de una sociedad anónima. Cabe resaltar que poseer acciones de una compañía confiere legitimidad al accionista para exigir sus derechos, pero a su vez para cumplir con sus obligaciones.

**BVC**: Abreviatura de la Bolsa de Valores de Colombia. Es el único mercado de acciones y otros valores de Colombia, organizado a través de la estructura de bolsa.

**Capital**: En el contexto de nuestro sistema, el [capital](file:///L:\\Personal%202\\Especializaci%C3%B3n\\Proyecto%20integrador\\1%20-%20Requisitos\\Requisitos_REM_Default_Spanish_CRS.html" \l "GLO-0009" \o "En el contexto de nuestro sistema, el capital corresponde al saldo en efectivo que tiene un portafolio, en favor del inversionista.) corresponde al saldo en efectivo que tiene un [portafolio](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0006), en favor del [inversionista](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0007).

**Comisionista**: Son los únicos entes autorizados por la [BVC](file:///L:\\Personal%202\\Especializaci%C3%B3n\\Proyecto%20integrador\\1%20-%20Requisitos\\Requisitos_REM_Default_Spanish_CRS.html" \l "GLO-0005" \o "Abreviatura de la Bolsa de Valores de Colombia. Es el único mercado de acciones y otros valores de Colombia, organizado a través de la estructura de bolsa.) para vender o comprar en el mercado accionario. Cuando un [inversionista](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0007) realiza una compra, realmente es la firma [comisionista](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0008) la que hace la compra y luego le sede estos activos a través de una transacción.

**Deposito**: Es una operación en la que un [inversionista](file:///L:\\Personal%202\\Especializaci%C3%B3n\\Proyecto%20integrador\\1%20-%20Requisitos\\Requisitos_REM_Default_Spanish_CRS.html" \l "GLO-0007" \o "Es aquella persona natural (persona física) o jurídica que es propietaria de acciones de los distintos tipos de sociedades anónimas o comanditarias que pueden existir en el marco jurídico de cada país. En el caso de nuestro sistema, es el usuario que compra o vende acciones.) le hace un ingreso de efectivo a un [portafolio](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0006).

**Dividendo**: El [dividendo](file:///L:\\Personal%202\\Especializaci%C3%B3n\\Proyecto%20integrador\\1%20-%20Requisitos\\Requisitos_REM_Default_Spanish_CRS.html" \l "GLO-0004" \o "El dividendo activo, es la parte del beneficio obtenido por las sociedades mercantiles cuyos órganos sociales acuerdan que sea repartido entre los socios de las mismas. Es decir, una vez acordado su reparto, es un crédito del socio frente a la sociedad. De otro lado, el dividendo pasivo es el crédito que ostenta la sociedad mercantil frente al socio, por la parte del capital social que suscribió y que se comprometió a desembolsar. La diferencia entre las aportaciones y el desembolso inicial de los accionistas.) activo, es la parte del beneficio obtenido por las sociedades mercantiles cuyos órganos sociales acuerdan que sea repartido entre los socios de las mismas. Es decir, una vez acordado su reparto, es un crédito del socio frente a la sociedad. De otro lado, el [dividendo](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0004) pasivo es el crédito que ostenta la sociedad mercantil frente al socio, por la parte del [capital](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0009) social que suscribió y que se comprometió a desembolsar. La diferencia entre las aportaciones y el desembolso inicial de los accionistas.

**Enrepar Acciones**: Es la [acción](file:///L:\\Personal%202\\Especializaci%C3%B3n\\Proyecto%20integrador\\1%20-%20Requisitos\\Requisitos_REM_Default_Spanish_CRS.html" \l "GLO-0014" \o "Una acción es la parte alícuota del capital social de una sociedad anónima. Cabe resaltar que poseer acciones de una compañía confiere legitimidad al accionista para exigir sus derechos, pero a su vez para cumplir con sus obligaciones.) de realizar una operación [Repo](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0002) por parte de un [inversionista](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0007): Esta operación es como un préstamo, en donde un [inversionista](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0007) le pide dinero prestado a una firma [comisionista](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0008) y esta le concede el préstamo respaldado con las acciones que el [inversionista](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0007) tenga en su [portafolio](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0006).

**Estado financiero**: Es el saldo actual en [capital](file:///L:\\Personal%202\\Especializaci%C3%B3n\\Proyecto%20integrador\\1%20-%20Requisitos\\Requisitos_REM_Default_Spanish_CRS.html" \l "GLO-0009" \o "En el contexto de nuestro sistema, el capital corresponde al saldo en efectivo que tiene un portafolio, en favor del inversionista.) que tiene un [inversionista](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0007) en un [portafolio](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0006) y la cantidad de acciones que posee, los cuales son reconocidos por la firma [comisionista](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0008) que administra el [portafolio](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0006).

**Inversionista**: Es aquella persona natural (persona física) o jurídica que es propietaria de acciones de los distintos tipos de sociedades anónimas o comanditarias que pueden existir en el marco jurídico de cada país. En el caso de nuestro sistema, es el usuario que compra o vende acciones.

**Mercado Financiero**: Es un espacio en el que se realizan los intercambios de instrumentos financieros y se definen sus precios.

**Orden de compra**: Es una operación en la que un [inversionista](file:///L:\\Personal%202\\Especializaci%C3%B3n\\Proyecto%20integrador\\1%20-%20Requisitos\\Requisitos_REM_Default_Spanish_CRS.html" \l "GLO-0007" \o "Es aquella persona natural (persona física) o jurídica que es propietaria de acciones de los distintos tipos de sociedades anónimas o comanditarias que pueden existir en el marco jurídico de cada país. En el caso de nuestro sistema, es el usuario que compra o vende acciones.) le ordena a un [trader](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0016) la compra de una cantidad determinada de acciones a un cierto valor y dentro de un determinado rango de fechas establecido por el [inversionista](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0007).

**Orden de Venta**: Es una operación en la que un [inversionista](file:///L:\\Personal%202\\Especializaci%C3%B3n\\Proyecto%20integrador\\1%20-%20Requisitos\\Requisitos_REM_Default_Spanish_CRS.html" \l "GLO-0007" \o "Es aquella persona natural (persona física) o jurídica que es propietaria de acciones de los distintos tipos de sociedades anónimas o comanditarias que pueden existir en el marco jurídico de cada país. En el caso de nuestro sistema, es el usuario que compra o vende acciones.) le ordena a un [trader](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0016) la venta de una cantidad determinada de acciones a un cierto valor y dentro de un determinado rango de fechas establecido por el [inversionista](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0007).

**Portafolio**: La cartera de inversiones o [portafolio](file:///L:\\Personal%202\\Especializaci%C3%B3n\\Proyecto%20integrador\\1%20-%20Requisitos\\Requisitos_REM_Default_Spanish_CRS.html" \l "GLO-0006" \o "La cartera de inversiones o portafolio de inversiones, es el conjunto de activos financieros en los cuales se invierte, para el caso de nuestro sistema, estos activos pueden ser acciones o capital.) de inversiones, es el conjunto de activos financieros en los cuales se invierte, para el caso de nuestro sistema, estos activos pueden ser acciones o [capital](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0009).

**Repo**: Es una operación con pacto de recompra, es decir, una entidad financiera vende un activo con un pacto de recompra por un precio determinado dentro de un tiempo determinado. Esta operación también se conoce como reporto, Repurchase Agreement o Sale and Repurchase Agreement.

**Retiro**: Es una operación en la que un [inversionista](file:///L:\\Personal%202\\Especializaci%C3%B3n\\Proyecto%20integrador\\1%20-%20Requisitos\\Requisitos_REM_Default_Spanish_CRS.html" \l "GLO-0007" \o "Es aquella persona natural (persona física) o jurídica que es propietaria de acciones de los distintos tipos de sociedades anónimas o comanditarias que pueden existir en el marco jurídico de cada país. En el caso de nuestro sistema, es el usuario que compra o vende acciones.) le hace un [retiro](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0011) de efectivo a un [portafolio](file:///L:\Personal%202\Especializaci%C3%B3n\Proyecto%20integrador\1%20-%20Requisitos\Requisitos_REM_Default_Spanish_CRS.html#GLO-0006).

**Trader**: Persona que realiza la compra, venta de acciones en la Bolsa de Valores en nombre del [inversionista](file:///L:\\Personal%202\\Especializaci%C3%B3n\\Proyecto%20integrador\\1%20-%20Requisitos\\Requisitos_REM_Default_Spanish_CRS.html" \l "GLO-0007" \o "Es aquella persona natural (persona física) o jurídica que es propietaria de acciones de los distintos tipos de sociedades anónimas o comanditarias que pueden existir en el marco jurídico de cada país. En el caso de nuestro sistema, es el usuario que compra o vende acciones.) que pone la orden.

# Panorama del Proyecto

## Contexto del negocio

El propósito de BVC Móvil es posicionarse en los Marketplace (App Store, Google Play) como la mejor aplicación para el control de la Bolsa de Valores colombiana. Para esto se realizará un emprendimiento con 2 firmas comisionistas de bolsa que puedan estar interesadas en adquirir el software y entregarlo como valor agregado a sus clientes inversionistas. El software se sustentará económicamente con el uso de anuncios publicitarios.

## Contexto del sistema

El proyecto BVC Móvil tiene como misión crear una aplicación para el proceso de administración de portafolios de inversión en la Bolsa de Valores de Colombia dando respuesta a la necesidad de los inversionistas colombianos de contar con una herramienta en sus dispositivos móviles que sirva para tal fin.

La aplicación al ser de uso personal incluirá las siguientes funciones:

* Administración de cuentas.
* Órdenes de compra y venta.
* Depósitos y retiros de capital.
* Cálculo de dividendos.
* Operaciones con repos.
* Balance del portafolio en tiempo real.

La aplicación no realizará operaciones reales en la BVC, ya que no es su objetivo; su objetivo es el de permitir a los inversionistas en valores administrar el inventario de sus portafolios. La aplicación no se conectará inicialmente con otros mercados financieros diferentes a la BVC para la actualización de los valores de mercado.

# Drivers de la Arquitectura

## Restricciones de negocio

|  |  |
| --- | --- |
| **Nombre del Driver de Negocio: Lanzamiento de la aplicación** | |
| **Declaración:** | La aplicación debe ser lanzada al público la primera semana del mes de noviembre de 2013. |
| **Justificación:** | La aplicación móvil debe ser publicada en los Marketplace (App Store y Google Play) en la primera semana de noviembre con todas sus características disponibles, con el fin de conseguir una mayor cantidad de usuarios. |
| **Implicaciones:** | Ajustar cronograma y realizar análisis de riesgos. |

|  |  |
| --- | --- |
| **Nombre del Driver de Negocio: Portabilidad de la aplicación móvil** | |
| **Declaración:** | La aplicación debe poderse instalar en cualquier dispositivo móvil de gama media-alta. |
| **Justificación:** | Debe crearse una aplicación móvil multiplataforma que soporte cualquier tecnología que se lance al mercado, permitiendo escalabilidad y mantenimiento, con costos y riegos mínimos. |
| **Implicaciones:** | Utilizar tecnologías a la vanguardia Open Source con alta coexistencia con otros Frameworks o librerías. |

|  |  |
| --- | --- |
| **Nombre del Driver de Negocio: Disponibilidad de la BVC** | |
| **Declaración:** | Obtener información actualizada de la Bolsa de Valores colombiana. |
| **Justificación:** | Ninguna aplicación (Google Finance, Yahoo Finance) reporta información sobre el mercado bursátil colombiano, por lo que sería la primera aplicación que permita llevar un control sobre las acciones del mercado local. |
| **Implicaciones:** | Sincronización con servidores externos y/o páginas web, que permitan mantener la información del sistema actualizada. |

|  |  |
| --- | --- |
| **Nombre del Driver de Negocio: Facilidad de uso en las UI** | |
| **Declaración:** | Las aplicaciones amigables generan al usuario una mayor confiabilidad y disminuyen el tiempo de aprendizaje en el uso de la misma. |
| **Justificación:** | La facilidad en el manejo de aplicaciones da al usuario una mejor experiencia en el uso, un menor tiempo en el aprendizaje y un mayor agrado al trabajar, por lo que su actitud frente al cambio y usos puede mejorar su productividad. |
| **Implicaciones:** | Utilizar tecnologías Open Source con una amplia gama de componentes enriquecidos. |

|  |  |
| --- | --- |
| **Nombre del Driver de Negocio: Publicidad** | |
| **Declaración:** | La aplicación será sustentada a partir del uso de anuncios publicitarios. |
| **Justificación:** | El uso de anuncios publicitarios será una de las estrategias de sustento económico de la aplicación. |
| **Implicaciones:** | En la configuración de la aplicación se debe conservar el espacio para la publicidad. |

## Restricciones técnicas

|  |  |
| --- | --- |
| **Nombre del Driver Técnico: Elección de Middlewares de desarrollo** | |
| **Declaración:** | El sistema debe estar implementado en un lenguaje de programación gratuito y potente. |
| **Justificación:** | Deben usarse middleware con gran escalabilidad y gratuitos, que nos permitan tener un sistema potente y con excelentes características en cuanto a tiempo de respuestas y performance en los servidores. |
| **Implicaciones:** | Se usará JEE como base de desarrollo, Postgres como SGBD, HTML5 y CSS3 para la creación de interfaces de usuario. |

|  |  |
| --- | --- |
| **Nombre del Driver Técnico: Presentación desacoplada** | |
| **Declaración:** | Las interfaces de usuario deben estar desacopladas de la lógica del negocio. |
| **Justificación:** | Al generar interfaces separadas es más factible realizar escalabilidad y extensibilidad al producto sin necesidad de interactuar con la lógica del negocio, por lo que su costo y riego sería menor a comparación de las capas acopladas. |
| **Implicaciones:** | Trabajar con un estilo arquitectural en mensajería nos ayuda a desacoplar la presentación de la lógica. |

|  |  |
| --- | --- |
| **Nombre del Driver Técnico: Seguridad en el aplicativo** | |
| **Declaración:** | El login de usuarios en la aplicación debe tener control de acceso mediante autentificación. |
| **Justificación:** | La administración de los maestros y acceso a los datos del aplicativo debe estar controlada mediante un login de usuario, este debe estar cifrado en MD5 el cual consiste en un algoritmo de una sola vía. |
| **Implicaciones:** | Se debe generar un sistema de cifrado MD5 para almacenar las claves de los usuarios. |

|  |  |
| --- | --- |
| **Nombre del Driver Técnico: Actualización valores BVC** | |
| **Declaración:** | La actualización de valores debe ser automática, esta será especificada por un variable de tiempo definida por el usuario. |
| **Justificación:** | La automatización del sistema mediante la actualización de valores le da al software cierta independencia de control humano por lo que resulta ser más fiable. |
| **Implicaciones:** | Se debe realizar un análisis para la automatización de tareas, puede implementarse mediante un Timer (EJB), o un Servlet (crontab o bat). |

|  |  |
| --- | --- |
| **Nombre del Driver Técnico: Creación de Logs de auditoria** | |
| **Declaración:** | Debe implementarse un log en archivo plano para el control de auditoría en el sistema. |
| **Justificación:** | El registro de logs nos ayuda a mantener un control de la información, así como un monitoreo sobre las actividades del sistema donde podemos detectar fallas o posibles errores y corregirlos generando una aplicación con mayor calidad. |
| **Implicaciones:** | Se debe buscar, analizar e implementar una librería gratuita que permita el control de logs basado en archivos planos. |

# Estilos Arquitectónicos Usados

|  |  |
| --- | --- |
| **Nombre del Estilo: SOA** | |
| **Declaración:** | Se define como la utilización de un conjunto de servicios mediante mensajería que se encarga de dar soporte a los requisitos de negocio. |
| **Justificación:** | El objetivo principal de implementar SOA es establecer servicios autónomos, distribuidos, con bajo acoplamiento e interoperables que oculten la complejidad del negocio especificando un contrato específico para cada consumidor. La reutilización de servicios comunes incrementa las oportunidades de escalabilidad y mantenimiento, reduciendo costos futuros. |
| **Implicaciones:** | La metodología de desarrollo debe estar alineada con una clara orientación a servicios, los desarrolladores deben adaptarse a un conjunto de prácticas y políticas para garantizar coherencia en el desarrollo. |

|  |  |
| --- | --- |
| **Nombre del Estilo: 3-Tiers / N-Tiers** | |
| **Declaración:** | El sistema debe estar definido en múltiples capas físicas: Presentación (móvil y aplicación web), Dominio del Negocio (servidor de aplicaciones) y Datos (nube o servidor con un SGBD). |
| **Justificación:** | La arquitectura N-Capas se caracteriza por la descomposición en capas físicas de aplicaciones, componentes, y despliegues distribuidos, ofreciendo una mayor escalabilidad, mantenibilidad, flexibilidad y disponibilidad. Cada capa es completamente independiente de las demás, lo que nos puede brindar una mayor interoperabilidad, la capa n es capaz de manejar las peticiones de la capa n+1 y trasladarlas a la capa n-1 (si la hay). La comunicación entre las capas suele ser asincrónica, brindando así una mayor escalabilidad en el sistema. |
| **Implicaciones:** | Mayor costo en recursos, implementación de un patrón o estilo arquitectónico de comunicación entre capas y especificación de mensajería para el mismo. |

|  |  |
| --- | --- |
| **Nombre del Estilo: Domain Driven Design** | |
| **Declaración:** | DDD se encarga del análisis del dominio del negocio, sus relaciones y elementos. Este puede verse como un marco de soluciones que pueden ser racionalizadas. |
| **Justificación:** | Es un estilo orientado a objetos que se debe considerar si se tiene un dominio de negocio complejo, esto con el fin de realizar el modelado y la definición de los objetos haciendo el modelo del dominio el motor principal del software; siendo este extensible, verificable y flexible, lo que genera una fácil actualización y extensión aun cuando los requisitos cambien, actuando como un conjunto de prácticas y terminologías para tomar decisiones de diseño que enfoquen y aceleren el manejo de dominios complejos en los proyectos de software. |
| **Implicaciones:** | Implementación de arquitectura Layered distinguiendo las diferentes capas y sub-capas internas en la aplicación, delimitando la situación de los diferentes componentes por su tipología. |

|  |  |
| --- | --- |
| **Nombre del Estilo: Layered** | |
| **Declaración:** | La división en capas lógicas se enfoca en mantener agrupadas las funcionalidades parecidas en diferentes niveles a los que se le agregan responsabilidades. |
| **Justificación:** | La arquitectura en capas se encarga de realizar una abstracción de las vistas proporcionando suficientes detalles para entender la individualidad de las capas y la relación entre ellas, estas deben tener una alta cohesión, deben ser reusables con bajo acoplamiento y encapsulamiento de los objetos, lo que puede permitir componentes altamente escalables, con un buen nivel de concurrencia y una buena mantenibilidad sin dañar su funcionamiento. |
| **Implicaciones:** | Análisis de patrones de diseño orientados a la agrupación de funcionalidades y posible reutilización dentro de los componentes. |

# Vista Conceptual

En esta vista se ilustra la visión general que tienen los inversionistas de la aplicación **BVC-MÓVIL**:



**Paquete Portafolios**: Módulo principal de la aplicación desde donde se gestionan los portafolios del inversionista. Contiene los siguientes casos de uso:

* Gestionar portafolios: Consultar la lista de portafolios del inversionista, agregar o editar la información de un portafolio.
* Gestionar transacciones: Gestionar las transacciones de un portafolio en particular para alterar su estado financiero.
* Generar balance: Generar el estado financiero de un portafolio de acuerdo a las entradas y salidas de capital.



**Paquete Depósitos**: Permite gestionar los depósitos de capital sobre los portafolios. Contiene los siguientes casos de uso:

* Consultar depósitos: Consultar el histórico de depósitos que se han realizado sobre un portafolio.
* Depositar: Agregar capital al portafolio del inversionista para poder realizar operaciones de compra.

****

**Paquete Retiros**: Permite gestionar los retiros de capital sobre los portafolios. Contiene los siguientes casos de uso:

* Consultar retiros: Consultar el histórico de retiros que se han realizado sobre un portafolio.
* Retirar: Retirar capital del portafolio de un inversionista de acuerdo a sus rendimientos financieros.



**Paquete Compras**: Permite gestionar las órdenes de compra de acciones por parte del inversionista. Contiene los siguientes casos de uso:

* Consultar compras: Consultar el histórico de compras que se han realizado sobre un portafolio.
* Colocar orden de compra: Colocar una orden de compra de acciones, indicando el intervalo de tiempo, la cantidad y el valor unitario de la acción.
* Cancelar compra: Cancelar una orden de compra de acciones siempre y cuando esta no haya sido procesada por la firma comisionista.

****

**Paquete Ventas**: Permite gestionar las órdenes de venta de acciones por parte del inversionista. Contiene los siguientes casos de uso:

* Consultar ventas: Consultar el histórico de ventas que se han realizado sobre un portafolio.
* Colocar orden de venta: Colocar una orden de venta de acciones, indicando el intervalo de tiempo, la cantidad y el valor unitario de la acción.
* Cancelar venta: Cancelar una orden de venta de acciones siempre y cuando esta no haya sido procesada por la firma comisionista.



**Paquete Repos**: Permite gestionar las operaciones Repo (Prestamos con respaldo de acciones). Contiene los siguientes casos de uso:

* Consultar repos: Consultar el histórico de repos que se han realizado sobre un portafolio.
* Enrepar: Solicitar un préstamo de capital a la firma comisionista respaldado con acciones contenidas en el portafolio.
* Desenrepar: Pagar un préstamo hecho a la firma comisionista liberando las acciones que respaldaron el préstamo.



**Paquete Acciones**: Permite consultar las acciones de la BVC.

* Consultar acciones: Consultar acciones por un criterio: Emisor, mnemotécnico.
* Detallar acción: Ver los detalles de una acción en particular: Cantidad en circulación, valor mínimo, máximo, volumen de negociación, histórico.



**Paquete Seguridad**: Permite gestionar la seguridad del sistema a través de usuarios y credenciales.

* Gestionar usuarios: Operaciones CRUD sobre los usuarios del sistema.
* Autenticar usuario: Permitir acceso a un usuario del sistema a través de sus credenciales.



# Vista Lógica

En esta vista se ilustra la visión que tiene el desarrollador de la aplicación **BVC-MÓVIL**:



Nuestra arquitectura de referencia se basa en el modelo DDD (Domain Driven Design) propuesto a una arquitectura de N-Capas, sin embargo se hizo una unión con patrones arquitecturales SOA (Service Oriented Architecture), por lo que varió la estructura añadiendo nuevos elementos a la misma.

**Infrastructure Layer:** Esta capa se encarga de interactuar con los componentes del sistema y la base de datos. Los patrones DAO y Utility Abstraction serán implementados en este componente, centralizando todas las peticiones.

**Utility Layer:** Esta capa provee servicios de utilidad reutilizables entre capas, por ejemplo conversiones a XML, detección de codificación, envíos de correo electrónico, entre otros.

**Domain Model Layer:** Esta capa contiene todos los artefactos de manejo del dominio de la aplicación, siendo los mismos desacoplados, permitiendo escalabilidad, y distribuyendo la responsabilidad de acuerdo a la funcionalidad de cada uno.

**Application Layer:** Esta capa se va a encarga de publicar los servicios y operaciones disponibles que va a realizar el sistema, así como la interacción entre los artefactos de negocio.

**Web App:** Esta capa comprende el componente de presentación Web donde el administrador realiza la configuración o parametrización del sistema.

**Mobile App:** Este es el componente final de usuario, donde los inversionistas podrán administrar sus portafolios de inversión en la BVC, para realizar las diferentes transacciones como depósitos, retiros, compra y venta de acciones entre otras operaciones.

# Vista Física



## Descripción de la plataforma tecnológica

**Java EE**

Es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process, Java EE es también considerada informalmente como un estándar debido a que los proveedores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE.

Es una plataforma poderosa con bastante usabilidad que nos permite diseñar aplicaciones robustas con mucha facilidad, además de contar con innumerables Apis, Frameworks y librerías que nos serán de mucha ayuda.

**HTML5**

Es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: un «clásico» HTML (text/html), la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5 que deberá ser servida como XML (XHTML) (application/xhtml+xml).

Las principales ventajas en el uso de la misma son:

* Es nativo, y por tanto independiente de plugins de terceros. Es decir, no pertenece a nadie, es opensource.
* Es más semántico, con etiquetas que permiten clasificar y ordenar en distintos niveles y estructuras el contenido. Además, incorpora metadatos de manera más formal, favoreciendo el posicionamiento SEO y la accesibilidad.
* El código es más simple lo que permite hacer páginas más ligeras que se cargan más rápidamente favoreciendo la usabilidad y la indexación en buscadores.
* Ofrece una compatibilidad mayor con los navegadores de dispositivos móviles.
* Incluye la etiqueta de dibujo canvas, que ofrece más efectos visuales.
* Permite realizar diseños adaptables a distintos dispositivos (web, tablets, móviles…)

**Postgres**

Es un Sistema de Gestión de Bases de Datos (SGBD) relacional orientado a objetos y libre, publicado bajo la licencia BSD.

Algunas de sus principales características son:

* Alta concurrencia
* Amplia variedad de tipos nativos
* Licenciamiento Libre
* Integridad transaccional.

## Consideraciones de implementación de los componentes a construir

|  |  |
| --- | --- |
| **Nombre del Componente: BVCMovil.apk** | |
| **Tipo de componente (aplicación, negocio, infraestructura): aplicación** | |
| **Plataforma en la que va a ser implementado** | HTML5  CSS3  JavaScript |
| **Atributos de calidad asociados al componente** | Eficiencia.  Portabilidad.  Facilidad de Aprendizaje.  Facilidad de Comprensión.  Seguridad. |
| **Recomendaciones para su implementación** | Utilizar un framework MVC para estandarizar el código, algunas propuestas son: The M-Project, BackBone.js, AngularJS, etc… |
| **Consideraciones para su despliegue** | Al no ser reconocido en viejas versiones de navegadores por sus nuevas etiquetas, se le recomienda al usuario actualizar a la versión más nueva, para poder disfrutar de todo el potencial que provee HTML5. |

|  |  |
| --- | --- |
| **Nombre del Componente: BVCAdmin** | |
| **Tipo de componente (aplicación, negocio, infraestructura): aplicación** | |
| **Plataforma en la que va a ser implementado** | JAVA (JEE)  JSF 2.1 (PrimeFaces) |
| **Atributos de calidad asociados al componente** | Eficiencia.  Portabilidad.  Facilidad de Aprendizaje.  Facilidad de Comprensión.  Operatividad.  Seguridad. |
| **Recomendaciones para su implementación** | Al utilizar el Framework Primefaces tenemos mayor posibilidad de crear UI (interfaces de usuario) más amigables y de mayor comprensión, además realizando consultas mediante objetos XML tendremos una mayor seguridad en la información. |
| **Consideraciones para su despliegue** | Tener instalado en el servidor el JDK 1.6 o superior, y el servidor de despliegue Glassfish 3.1. |

|  |  |
| --- | --- |
| **Nombre del Componente: BVC** | |
| **Tipo de componente (aplicación, negocio, infraestructura): infraestructura** | |
| **Plataforma en la que va a ser implementado** | Postgres  T-SQL |
| **Atributos de calidad asociados al componente** | Utilización de Recursos.  Seguridad.  Compatibilidad.  Interoperabilidad.  Facilidad de Prueba. |
| **Recomendaciones para su implementación** | Generar un SQL nativo para la creación de las estructuras con el fin de poder realizar una migración a cualquier otro motor sin necesidad de realizar cambios de profundidad en el sistema. |
| **Consideraciones para su despliegue** | Tener instalado en el servidor Postgres 9.19 o superior, además de los controladores JDBC4 9.1 o superior. |

|  |  |
| --- | --- |
| **Nombre del Componente: EntityAbstractionMgt** | |
| **Tipo de componente (aplicación, negocio, infraestructura): infraestructura** | |
| **Plataforma en la que va a ser implementado** | JAVA (JEE)  EJB 3.1 |
| **Atributos de calidad asociados al componente** | Eficiencia.  Cambiabilidad.  Facilidad de Prueba.  Interoperabilidad. |
| **Recomendaciones para su implementación** | EJB nos permite tener interfaces remotas y locales para manipular las operaciones e instancias manteniendo el código ordenado, para el control de los entities de bases de datos manejaremos Querys nativos de SQL utilizando patrones DAO. |
| **Consideraciones para su despliegue** | Tener instalado en el servidor el JDK 1.6 o superior, y el servidor de despliegue Glassfish 3.1. |

|  |  |
| --- | --- |
| **Nombre del Componente: UtilityAbstractionMgt** | |
| **Tipo de componente (aplicación, negocio, infraestructura): negocio** | |
| **Plataforma en la que va a ser implementado** | JAVA (JEE)  EJB 3.1 |
| **Atributos de calidad asociados al componente** | Eficiencia.  Cambiabilidad.  Facilidad de Prueba.  Interoperabilidad. |
| **Recomendaciones para su implementación** | EJB nos permite tener interfaces remotas y locales para manipular las operaciones e instancias manteniendo el código ordenado, para el control de objetos XML se usaran las librerías JDOM y el envió de correos con las librerías mail de java. |
| **Consideraciones para su despliegue** | Tener instalado en el servidor el JDK 1.6 o superior, el servidor de despliegue Glassfish 3.1, verificar que estén incluidas las librerías de utilidad en el jar de despliegue o en el contexto del servidor de despliegue. |

|  |  |
| --- | --- |
| **Nombre del Componente: BvcWS** | |
| **Tipo de componente (aplicación, negocio, infraestructura): negocio** | |
| **Plataforma en la que va a ser implementado** | JAVA (JEE)  Web Services especificaciones 2.0 SOAP |
| **Atributos de calidad asociados al componente** | Eficiencia.  Cambiabilidad.  Facilidad de Prueba.  Interoperabilidad.  Recuperabilidad.  Seguridad. |
| **Recomendaciones para su implementación** | Debe implementarse en la especificación 2.0 de WebServices las consultas se recibirán como objetos y los retornos de las peticiones hacia los clientes serán de tipo XML. |
| **Consideraciones para su despliegue** | Tener instalado en el servidor el JDK 1.6 o superior, el servidor de despliegue Glassfish 3.1, verificación de la librería JDOM para el uso de XML. |

|  |  |
| --- | --- |
| **Nombre del Componente: PortafolioMgt, AccionesMgt, EmpresasMgt, UsuariosMgt, DIvisasMgt, LugaresMgt** | |
| **Tipo de componente (aplicación, negocio, infraestructura): negocio** | |
| **Plataforma en la que va a ser implementado** | JAVA (JEE)  EJB 3.1 |
| **Atributos de calidad asociados al componente** | Eficiencia.  Cambiabilidad.  Facilidad de Prueba.  Interoperabilidad.  Madurez.  Recuperabilidad.  Compatibilidad. |
| **Recomendaciones para su implementación** | EJB nos permite tener interfaces remotas y locales para manipular las operaciones e instancias manteniendo el código ordenado. |
| **Consideraciones para su despliegue** | Tener instalado en el servidor el JDK 1.6 o superior, y el servidor de despliegue Glassfish 3.1. |

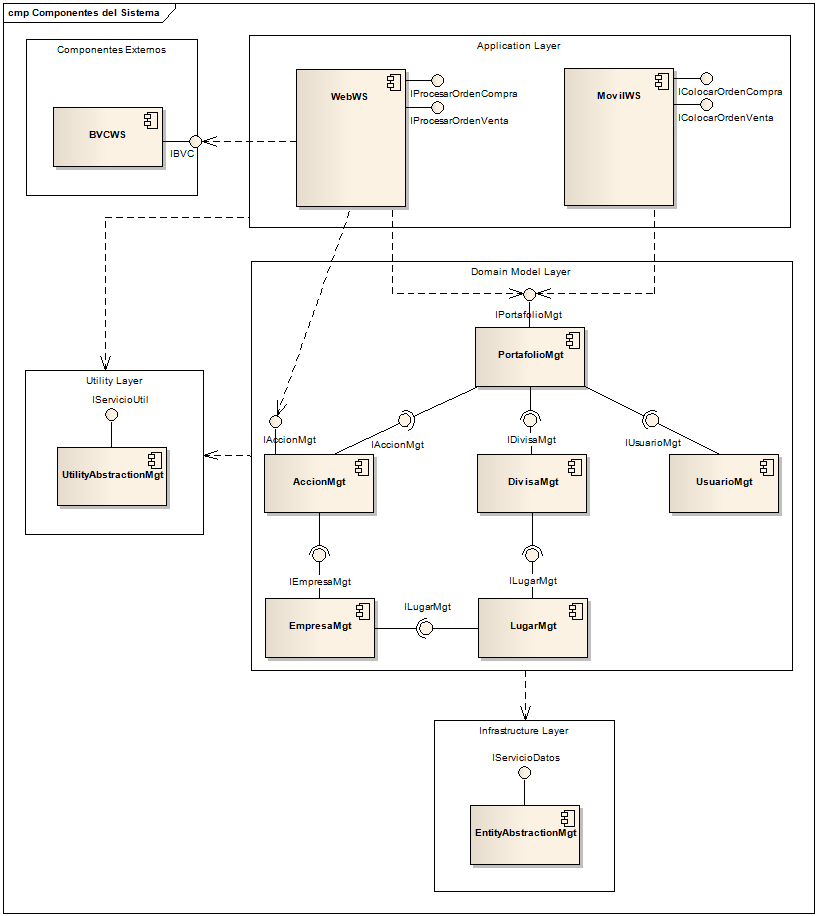
## Consideraciones de componentes a reutilizar

|  |  |
| --- | --- |
| **Nombre del Componente: EntityAbstractionMgt** | |
| **Tipo de componente (aplicación, negocio, infraestructura): infraestructura** | |
| **Justificación de compra o reúso** | El componente se encarga de realizar toda la gestión transaccional a la base de datos por lo que será invocado por todos los componentes de negocio reutilizando las conexiones y Querys eliminando así redundancia en el código. |
| **Recomendaciones para su adquisición** | El desarrollo debe ser nativo y sus características de alto nivel se irán desarrollando de acuerdo a las necesidades de las entidades versus los componentes de invocación. |
| **Sitio / Origen del componente** | Desarrollo a la medida. |

|  |  |
| --- | --- |
| **Nombre del Componente: UtilityAbstractionMgt** | |
| **Tipo de componente (aplicación, negocio, infraestructura): negocio** | |
| **Justificación de compra o reúso** | EL componente se encargara de realizar la conversión de objetos a XML y viceversa, el envió de correos y demás funciones de utilidad que puedan a llegar a ser comunes en los demás componentes de negocio. |
| **Recomendaciones para su adquisición** | El desarrollo se hará basado de acuerdo a la necesidad de los componentes, y el grado de reutilización de la funcionalidad. |
| **Sitio / Origen del componente** | Desarrollo a la medida. |

# Vista de Implementación

En esta vista se ilustra cómo se implementan los componentes que requieren especial cuido en las interfaces, enfatizando en la interacción entre ellos. Primero presentaremos el diagrama general de componentes, con la interacción entre estos y luego presentaremos la estructura interna de nuestros componentes de negocio:

















## Interfaces y Operaciones de la capa Application Layer

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: WebWS** | |
| **Nombre de la Interface Provista: IProcesarOrdenCompra** | |
| **Nombre de la Operación** | actualizarEstado() |
| **Parámetros entrada** | detalle: DetalleCompra (Data transfer object) |
| **Parámetros salida** | resultado: int (0-> fracaso, 1-> éxito) |
| **Precondiciones** | La orden de compra no puede estar en estado “Cancelada” |
| **Postcondiciones** | Al final, habrá cambiado el estado actual de la orden de compra. |
| **Invariantes** | El valor total de la orden de compra seguirá siendo el mismo antes y después del cambio de estado. |
| **Resultados Observables** | Cuando el inversionista consulta sus órdenes de compra, puede observar que la orden fue procesada por un trader. |
| **Consideraciones de implementación** | Utilizar Proxy. La implementación y llamado RCP va a ser mediante nombre de EJB en un archivo .properties configurable. |

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: WebWS** | |
| **Nombre de la Interface Provista: IProcesarOrdenCompra** | |
| **Nombre de la Operación** | obtenerComprasPendientes() |
| **Parámetros entrada** | idTrader: String |
| **Parámetros salida** | resultado: DetalleCompra[] (Data transfer object collection) |
| **Precondiciones** | El id del trader proporcionado como argumento existe en la base de datos. |
| **Postcondiciones** | Al final, se retorna la lista de compras pendientes asignadas al trader cuyo id se recibe como argumento. |
| **Invariantes** | La información de la base de datos no será modificada con esta operación. |
| **Resultados Observables** | Cuando el trader está consultado su lista de transacciones pendientes por procesar, podrá observar las compras devueltas por este método. |
| **Consideraciones de implementación** | Utilizar Proxy. La implementación y llamado RCP va a ser mediante nombre de EJB en un archivo .properties configurable. |

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: WebWS** | |
| **Nombre de la Interface Provista: IProcesarOrdenVenta** | |
| **Nombre de la Operación** | actualizarEstado() |
| **Parámetros entrada** | detalle: DetalleVenta (Data transfer object) |
| **Parámetros salida** | resultado: int (0-> fracaso, 1-> éxito) |
| **Precondiciones** | La orden de venta no puede estar en estado “Cancelada” |
| **Postcondiciones** | Al final, habrá cambiado el estado actual de la orden de venta. |
| **Invariantes** | El valor total de la orden de venta seguirá siendo el mismo antes y después del cambio de estado. |
| **Resultados Observables** | Cuando el inversionista consulta sus órdenes de venta, puede observar que la orden fue procesada por un trader. |
| **Consideraciones de implementación** | Utilizar Proxy. La implementación y llamado RCP va a ser mediante nombre de EJB en un archivo .properties configurable. |

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: WebWS** | |
| **Nombre de la Interface Provista: IProcesarOrdenVenta** | |
| **Nombre de la Operación** | obtenerVentasPendientes() |
| **Parámetros entrada** | idTrader: String |
| **Parámetros salida** | resultado: DetalleVenta[] (Data transfer object collection) |
| **Precondiciones** | El id del trader proporcionado como argumento existe en la base de datos. |
| **Postcondiciones** | Al final, se retorna la lista de ventas pendientes asignadas al trader cuyo id se recibe como argumento. |
| **Invariantes** | La información de la base de datos no será modificada con esta operación. |
| **Resultados Observables** | Cuando el trader está consultado su lista de transacciones pendientes por procesar, podrá observar las ventas devueltas por este método. |
| **Consideraciones de implementación** | Utilizar Proxy. La implementación y llamado RCP va a ser mediante nombre de EJB en un archivo .properties configurable. |

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: MovilWS** | |
| **Nombre de la Interface Provista: IColocarOrdenCompra** | |
| **Nombre de la Operación** | crearOrdenCompra() |
| **Parámetros entrada** | i: idInversionista  p: DetallePortafolio,  a: DetalleAccion,  c: DetalleCompra |
| **Parámetros salida** | Resultado:  0: Éxito de la operación  1: Error: El inversionista es inválido  2: Error: El portafolio es inválido  3: Error: La acción es inválida  4: Error: Saldo insuficiente  5: Error: Fechas inválidas  6: Error: Cantidad de acciones inválidas  7: Error: El valor unitario difiere del rango permitido  8: Error: No existe un trader al cual asignarle la orden |
| **Precondiciones** | idInversionista es una identificación válida de inversionista  El saldo del portafolio es mayor o igual al valor total de la orden de compra  La fecha inicial de la orden de compra es mayor o igual a la fecha actual  La fecha final de la orden de compra es mayor o igual a la fecha inicial  La cantidad de acciones a ordenar es mayor a 0 y menor o igual a la cantidad en circulación  El valor unitario de la orden de compra no difiere del valor actual de la acción en más o menos de 5%  La firma comisionista debe tener por lo menos un trader para asignar la orden |
| **Postcondiciones** | Al final se retorna un código indicando el estado con el que terminó la operación. |
| **Invariantes** | El saldo del portafolio del inversionista sigue siendo el mismo. |
| **Resultados Observables** | Cuando el inversionista guarda una orden de compra, la puede volver a recuperar tras cerrar la aplicación. |
| **Consideraciones de implementación** | Utilizar patrón Transaction para garantizar la atomicidad de la operación y recuperación del sistema. |

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: MovilWS** | |
| **Nombre de la Interface Provista: IColocarOrdenVenta** | |
| **Nombre de la Operación** | crearOrdenVenta() |
| **Parámetros entrada** | i: idInversionista  p: DetallePortafolio,  a: DetalleAccion,  c: DetalleVenta |
| **Parámetros salida** | Resultado:  0: Éxito de la operación  1: Error: El inversionista es inválido  2: Error: El portafolio es inválido  3: Error: La acción es inválida  4: Error: Fechas inválidas  5: Error: Cantidad de acciones inválidas  6: Error: El valor unitario difiere del rango permitido  7: Error: No existe un trader al cual asignarle la orden |
| **Precondiciones** | idInversionista es una identificación válida de inversionista  La fecha inicial de la orden de venta es mayor o igual a la fecha actual  La fecha final de la orden de venta es mayor o igual a la fecha inicial  La cantidad de acciones a vender es mayor a 0 y menor o igual a la cantidad actual en el portafolio  El valor unitario de la orden de venta no difiere del valor actual de la acción en más o menos de 5%  La firma comisionista debe tener por lo menos un trader para asignar la orden |
| **Postcondiciones** | Al final se retorna un código indicando el estado con el que terminó la operación. |
| **Invariantes** | El saldo del portafolio del inversionista aumenta. |
| **Resultados Observables** | Cuando el inversionista guarda una orden de venta, la puede volver a recuperar tras cerrar la aplicación. |
| **Consideraciones de implementación** | Utilizar patrón Transaction para garantizar la atomicidad de la operación y recuperación del sistema. |

## Interfaces y Operaciones de la capa Domain Model Layer

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: AccionMgt** | |
| **Nombre de la Interface Provista: IAccionMgt** | |
| **Nombre de la Operación** | actualizarValor() |
| **Parámetros entrada** | detalle: DetalleAccion (Data transfer object) |
| **Parámetros salida** | resultado: int (0-> fracaso, 1-> éxito) |
| **Precondiciones** | El valor a actualizar es positivo, la acción existe en la base de datos. |
| **Postcondiciones** | Al final, la acción recibida como argumento estará actualizada con su valor en la BVC. |
| **Invariantes** | El resto de información de la acción no cambia, únicamente cambiará su valor. |
| **Resultados Observables** | Cuando se va a hacer una nueva compra o venta de la acción, se debe mostrar el valor actualizado. |
| **Consideraciones de implementación** | Utilizar procedimientos almacenados y patrón DAO. |

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: AccionMgt** | |
| **Nombre de la Interface Provista: IAccionMgt** | |
| **Nombre de la Operación** | obtenerAcciones() |
| **Parámetros entrada** | NINGUNO |
| **Parámetros salida** | resultado: DetalleAccion[] (Data transfer object collection) |
| **Precondiciones** | Previamente se ha matriculado la lista de acciones en los maestros del sistema. |
| **Postcondiciones** | Al final, se retorna la lista de acciones para todos los emisores de la BVC. |
| **Invariantes** | La información de la base de datos no será modificada con esta operación. |
| **Resultados Observables** | Cuando se va a realizar una nueva transacción con acciones, se debe desplegar la lista de acciones devueltas con este método. |
| **Consideraciones de implementación** | Utilizar procedimientos almacenados y patrón DAO. |

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: PortafolioMgt** | |
| **Nombre de la Interface Provista: IPortafolioMgt** | |
| **Nombre de la Operación** | obtenerPortafolios() |
| **Parámetros entrada** | idInversionista: String |
| **Parámetros salida** | resultado: DetallePortafolio[] (Data transfer object collection) |
| **Precondiciones** | El id del inversionista es una identificación válida. |
| **Postcondiciones** | Al final se retorna un array con todos los portafolios pertenecientes al inversionista cuyo id se recibe como argumento. |
| **Invariantes** | La información de la base de datos no será modificada con esta operación. |
| **Resultados Observables** | Cuando el inversionista ingresa a la aplicación móvil, podrá ver en la pantalla inicial la lista de portafolios devueltos por este método. |
| **Consideraciones de implementación** | Utilizar procedimientos almacenados y patrón DAO. |

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: DivisaMgt** | |
| **Nombre de la Interface Provista: IDivisaMgt** | |
| **Nombre de la Operación** | convertirDivisas() |
| **Parámetros entrada** | divisaOrigen: String  divisaDestino: String  valor: double |
| **Parámetros salida** | resultado: double |
| **Precondiciones** | La divisa origen y destino son códigos de divisas válidos. |
| **Postcondiciones** | Al final se retorna la conversión del valor de la divisa origen hacia la divisa destino. |
| **Invariantes** | La información de la base de datos no será modificada con esta operación. |
| **Resultados Observables** | El inversionista podrá ver el balance de sus portafolios en la divisa seleccionada utilizando este método. |
| **Consideraciones de implementación** | Se debe tener cuidado con las operaciones de punto flotante para no tener pérdida de precisión. |

## Interfaces y Operaciones de la capa Infraestructure Layer

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: EntityAbstractionMgt** | |
| **Nombre de la Interface Requerida: IServicioDatos** | |
| **Descripción de la Operación** | realizarConsulta() |
| **Parámetros entrada** | sb: StringBuffer  parametros: List<String> |
| **Parámetros salida** | Resultado: List<T> |
| **Precondiciones** | Los Querys estarán predefinidos en unas variables estáticas de tipo String, los componentes harán el llamado al método enviando junto con la consulta la lista de parámetros necesarios para la ejecución. |
| **Postcondiciones** | Se realizará una consulta retornando ya sea un estado de inserción o actualización o una lista de objetos genérica de tipo <T> la cual será casteada al tipo de objeto que requerimos de acuerdo al componente. |
| **Invariantes** | N/A. |
| **Resultados Observables** | Al ser de nivel de datos no se verá ninguna información en el momento. |
| **Consideraciones de implementación** | Controlar los tipos de objetos para definir las interfaces de los genéricos, predefinir los Querys y estandarizar el diccionario de datos. |

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: EntityAbstractionMgt** | |
| **Nombre de la Interface Requerida: IServicioDatos** | |
| **Descripción de la Operación** | consultarQuerys() |
| **Parámetros entrada** | nombreQuery: String |
| **Parámetros salida** | resultado: StringBuffer |
| **Precondiciones** | Los Querys estarán predefinidos en unas variables estáticas de tipo String. |
| **Postcondiciones** | Se realizara una consulta retornando en un StringBuffer el query de consulta. |
| **Invariantes** | N/A. |
| **Resultados Observables** | Al ser de nivel de datos no se verá ninguna información en el momento. |
| **Consideraciones de implementación** | Ingresar los Querys con parametrizacion tipo (?), para evitar fallos de seguridad y SQLInjection. |

## Interfaces y Operaciones de la capa Utility Layer

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: UtilityAbstractionMgt** | |
| **Nombre de la Interface Requerida: IServicioUtil** | |
| **Descripción de la Operación** | convertirAXML(List<T>) |
| **Parámetros entrada** | parametros: List<T> |
| **Parámetros salida** | resultado: String XML |
| **Precondiciones** | Los objetos deben definir los objetos obligatorios según el XSD para no generar una excepción en el encode. |
| **Postcondiciones** | Se retornara un String con el XML generado. |
| **Invariantes** | N/A. |
| **Resultados Observables** | Al ser de nivel de datos no se verá ninguna información en el momento. |
| **Consideraciones de implementación** | Utilizar librerías JDOM para la implementación y verificar contra los XSD que la trama sea correcta. |

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: UtilityAbstractionMgt** | |
| **Nombre de la Interface Requerida: IServicioUtil** | |
| **Descripción de la Operación** | convertirDeXML() |
| **Parámetros entrada** | parametros: String XML |
| **Parámetros salida** | resultado: List<T> |
| **Precondiciones** | El String de entrada debe ser el mismo generado en el retorno del método ConvertirAXML para no generar excepción alguna |
| **Postcondiciones** | Se retornara una Lista de Objetos generitos <T>. |
| **Invariantes** | N/A. |
| **Resultados Observables** | Al ser de nivel de datos no se verá ninguna información en el momento. |
| **Consideraciones de implementación** | Utilizar librerías JDOM para la implementación y verificar contra los XSD que la trama sea correcta. |

|  |  |
| --- | --- |
| **Nombre del Componente o Parte: UtilityAbstractionMgt** | |
| **Nombre de la Interface Requerida: IServicioUtil** | |
| **Descripción de la Operación** | envioCorreo() |
| **Parámetros entrada** | destinatario : String  asunto : String  cuerpo : String  plantilla : String |
| **Parámetros salida** | resultado: int (0-> fracaso, 1-> éxito) |
| **Precondiciones** | Generar el cuerpo del correo que necesitan y seleccionar la plantilla y demás datos que se soliciten. |
| **Postcondiciones** | Generar una firma digital MD5 para validar la autenticidad del correo, y enviar el mismo al destinatario junto con la firma. |
| **Invariantes** | N/A. |
| **Resultados Observables** | Enviara el correo al destinatario. |
| **Consideraciones de implementación** | Utilizar librerías mail de JAVA, y para la generación del MD5 el paquete javax.security del api. |

# Tamaño y Rendimiento

A continuación se describe las consideraciones de espacio en disco, tiempo de respuesta por transacción y el rendimiento esperado en condiciones normales del sistema.

**Tiempo de respuesta en el acceso a la Base de Datos:** Las tablas de maestros tienen un tiempo máximo de transacción de 3 segundos

**Tiempo de respuesta en el acceso a Local Storage:** El Local Storage o almacenamiento en el dispositivo móvil no superará un tiempo máximo de 2 segundos por transacción.

**Tiempo de respuesta de transacciones de servicios:** El tiempo máximo de transacción de consulta en los servicios será de 8 segundos.

**Tiempo de respuesta de transacciones móvil:** El tiempo máximo de transacción en el dispositivo móvil será de 5 segundos

**Tiempo de respuesta de transacciones BVC:** El tiempo máximo de actualización de los valores del mercado será de 30 segundos en intervalos configurables por el administrador del sistema.

**Espacio en disco para la aplicación móvil:** El instalador de la aplicación para los dispositivos móviles tendrá un peso aproximado de entre 2.8 M y 3.5 M dependiendo del tipo de sistema operativo (Android, iOS, Blackberry, Windows), el mismo deberá tener un espacio aproximado de 5 M para el almacenamiento local de la información del cliente.

**Espacio en disco para la aplicación web:** El war de despliegue de la aplicación web junto con sus librerías y dependencias tendrá un peso aproximado de 8 M, esto sin contar el log de auditoría que se creará a diario con un peso máximo de 2 M.

**Espacio en disco para componentes y contenedor de servicios:** El espacio en disco requerido para los componentes, sus respectivas dependencias y el acceso a datos es de 70 M, esto sin contar con los log’s de auditoría que se crearán a diario con un peso máximo de 2 M por componente.

**Espacio en disco para el servidor de Base de Datos:** Aun no se tiene un aproximado delpeso de las tablas ni la configuración final de las mismas, por lo que se está contando con un espacio inicial de 500 M el cual se irá incrementando a medida que se requiera.

# Calidad

Los atributos de calidad mas importantes que debe cumplir el aplicativo para poder satisfacer las necesidades del cliente y la futura escalabilidad del mismo son:

**Portabilidad**: El aplicativo móvil debe ser multiplataforma y correr en cualquier dispositivo (Android, iOS, Blackberry , Windows), el aplicativo web y la lógica del negocio debe soportar el despliegue en diferentes sistemas operativos(Linux, Windows) y la base de datos podrá gestionarse desde cualquier SGBD(postgres, mySQL, Db2, Oracle, SQL Serer, etc).

**Usabilidad:** La aplicación movil debe ser intuitiva, las funcionalidades deben ser faciles de aprender y las interfaces deben entregar mensajes legibles al usuario.

**Interoperabilidad:** El desarrollo de una arquitectura SOA crea independencia en los componentes por lo que el sistema puede desacoplarse y asi crear diferentes interfaces de presentacion(JAVA, .NET, HTML, etc) o consultar desde sistemas externos a la lógica del dominio de negocio.

**Confidencialidad:** Solo el cliente debe poder obtener acceso a la informacion de su mercado y transacciones en la bolsa.

**Modificabilidad:** El sistema y sus respectivos componentes deben ser construidos con base en estandares y enfoques de desarrollo que faciliten el mantenimiento y los cambios futuros asi como la disminución de los riesgos.

**Disponibilidad:** Los servicios web deben tener una disponibilidad las 24 horas del dia durante 7 dias a la semana . Con una tiempo fuera de servicio inferior al 2% mensual lo que equivale a un maximo de 14 horas.