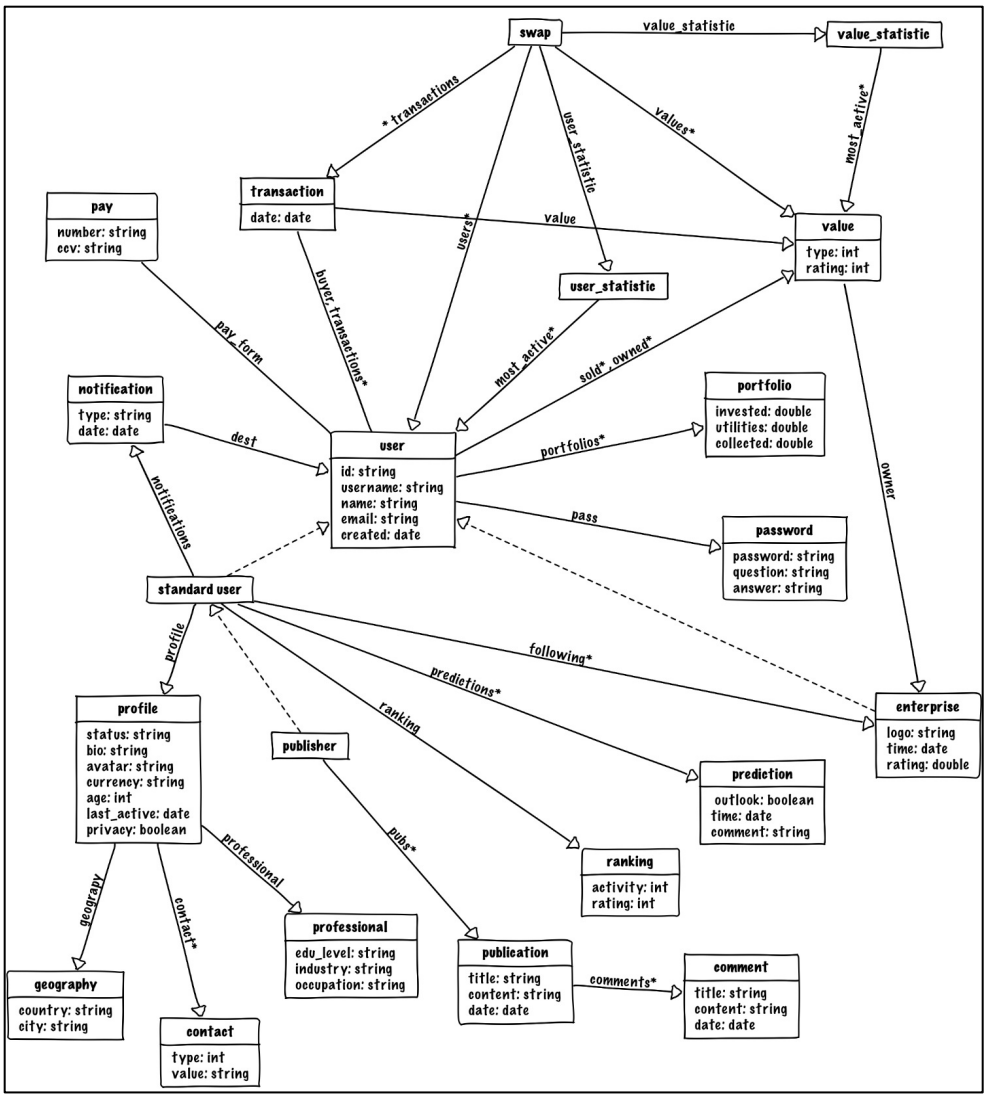
Sebastián Valencia Calderón
Juan Camilo Bages Prada

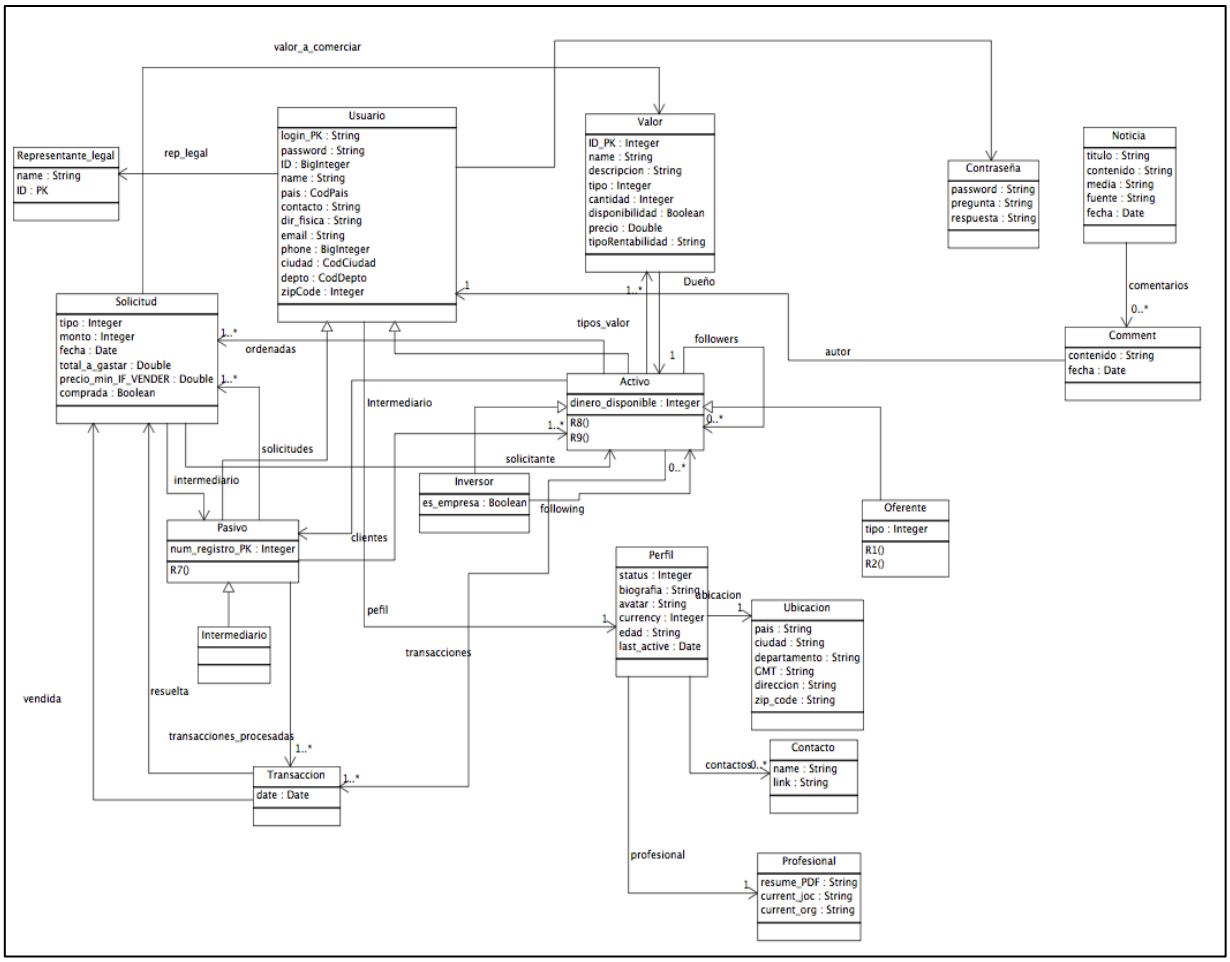
Análisis.

Para el desarrollo de esta iteración, fue necesario agregar ciertas características al modelo de la aplicación, es decir, la disposición de nuevos requerimientos hizo necesaria la inclusión o modificación de nuevas clases, asociaciones o atributos en el modelo propuesto originalmente. A continuación, se muestran los dos modelos, el anterior y el nuevo, es pertinente decir que el nuevo incluye los modelos auxiliares de desarrollo, así como los contemplados para cubrir el enunciado actual de la aplicación.

1. Modelo anterior: (UML-former.png)



2. Modelo actual: (UML-current.png)



Como puede observarse, algunas clases fueron agregadas, otras fueron modificadas y pocas fueron eliminadas, la comparación, se encuentra a continuación:

Comparación de las clases y modelos propuestos en ambas iteraciones.		
Clase	Modelo anterior	Modelo actual
Swap	En el modelo anterior, se traía el modelo conceptual de diseño adoptado por los cursos anteriores, donde era necesaria la presencia de una clase maestra que manejara los recursos y servicios principales de una aplicación.	Gracias a las nuevas metodologías de diseño adoptadas por la nueva ola de desarrollo de software, así como los patrones de diseño, el modelo enseñado por los cursos anteriores carece de sentido ante los

		requerimientos de una aplicación robusta en la modernidad.
Usuario	La clase de usuario fue fortalecida a la luz de los nuevos requerimientos. Algunas asociaciones únicas de esta clase, fueron encapsuladas en una sola clase de usuario. Esto favorece el diseño, sin embargo, la planeación posee problemas para la implementación o normalización de las tablas.	Se agregó la información adecuada para el manejo de monedas, básicamente, los modelos de perfil y contraseña, colapsaron en la clase usuario. Asimismo, se enfatizó la presencia de el usuario como un todo, del cual heredan los objetos reactivos de la aplicación. Esto facilitó el diseño normal de las tablas.
Representante legal		La clase fue agregada para favorecer el proceso de normalización.
Valor	La clase y el diseño de las asociaciones, no contemplaba la pluralidad de opciones que ofrecía o debía ofrecer este modelo (modelo como en MVC).	La tabla de valor, contempla las opciones de compra y otros requerimientos asociados a esta. Asimismo, se tiene en cuenta el proceso de adquisición por parte de un intermediario.
Solicitud	El concepto no estaba del todo integrado en la implementación del prototipo, anteriormente, todo el procesos se realizaba a través de transacciones. Asimismo, se fortaleció el concepto de transacción.	Las solicitudes poseen facilidades para ofertar, y manejar transacciones (en el sentid bancario) fácilmente. Pensando en lo anterior, poseen conocimiento del intermediario, oferente y fecha de realización.
Transacción		El concepto, así como la implementación, son resultado de des solicitudes.
Clases misceláneas		Aún no implementadas

Diseño.

a) A partir del diseño existente, analice el impacto que representa la introducción de los nuevos requerimientos y restricciones a nivel del modelo conceptual. Realice los cambios necesarios en su modelo relacional para respetar las reglas de negocio y asegurar la calidad del mismo. Tenga en cuenta los comentarios recibidos en la sustentación del taller 2. Documente el diseño y las decisiones tomadas para crear los elementos de la base de datos que da el respaldo de persistencia a la aplicación, a partir del modelo conceptual. Incluya un listado con las tablas generadas en la base de datos, utilizando los estándares establecidos, disponibles en la wiki del curso (sección tutoriales). Este listado debe incluir el nombre de la tabla, el nombre y el tipo de dato de sus campos así como los nombres de restricciones de llaves primarias, llaves foráneas y de chequeo. (Sea claro en mencionar explícitamente los cambios relevantes entre su diseño entregado con el taller anterior y este.)

   Dada la similitud estructural de ambos modelos, el cambio o migración manual de los modelos existentes, no representa un cambio significativo en costos de tiempo o instrucción de alguna tecnología o metodología. Esto se debe, a que el modelo inicial incluyó exhaustivamente los requerimientos propuestos en el primer enunciado. A continuación, se incluye el esquema de tabla de cada modelo propuesto, posteriormente, los cambios relevantes en el modelo con respecto a la anterior iteración.

USERS						
LOGIN	**USER_ID**	**FIRST_NAME**	**LAST_NAME**	**EMAIL**	**PHONE**	**USER_PASS**
varchar(20)	varchar(20)	varchar(20)	varchar(20)	varchar(20)	varchar(20)	varchar(20)
PK	NN	NN	NN	NN	NN	NN

   La tabla USERS contiene información relevante sobre los datos de acceso del usuario, asimismo, como se explicó, es la entidad fundamental de todo usuario de Swap, es decir, cualquier agente en Swap es un usuario, entiéndase agente como entidad manejada por un humano o un sistema de inteligencia artificial capaz de simular las acciones relativas a un humano. Se decidió incluir información de contacto dada la relación única de esta información con cada usuario. Sin embargo, se maneja flexibilidad para la normalización, nótese que ni EMAIL ni PHONE son PK.

ACTIVES	
USER_LOGIN	**AVAILABLE_MONEY**
varchar(20)	float
PK, FK(USERS.LOGIN)	NN,

La tabla ACTIVES contiene información relevante sobre los usuarios que son capaces de comprar acciones en el sistema, es decir, los usuarios activos. Pudo haberse incluido información sobre el tipo en la tabla USERS pero esto generaba redundancia no conveniente para la normalización del modelo. Por lo tanto, se maneja la tabla ACTIVES con información del login del usuario, es decir el PK en USERS, y el PK de ésta tabla. AVAILABLE_MONEY, representa la cantidad de dinero del usuario para comprar valores, o los que ha ganado en el proceso cotidiano de Swap.

PASSIVES	
PASSIVE_LOGIN	**PASSIVE_REGISTER**
varchar(20)	varchar(20)
PK, FK(USERS.LOGIN)	NN, UNIQUE

De manera análoga con la tabla ACTIVES, la tabla PASSIVES contiene información asociada a los usuarios pasivos del sistema. Éstos usuarios son los intermediarios, es decir, los usuarios capaces de interactuar por un usuario activo ante el mercado. Actúa como empleado de Swap, como el enunciado lo pedía, este tiene un número de registro único, que no es PK para favorecer la organización de índices dentro de la base de datos.

ACTIVESPASSIVES	
ACTIVE_LOGIN	**PASSIVE_LOGIN**
varchar(20)	varchar(20)
PK, FK(ACTIVES.USER_LOGIN)	PK, FK(PASSIVES.PASSIVE_LOGIN), UNIQUE

Esta relación, evidencia la conexión entre los intermediarios y los inversionistas, es decir, la asignación entre los intermediarios y sus clientes, además, tiene en cuenta que puede haber un intermediarios atendiendo solicitudes de varios clientes, sin embargo, no existe un inversionista con varios intermediarios asociados.

INVESTORS	
USER_LOGIN	**IS_ENTERPRISE**
varchar(20)	varchar(1)
PK, FK(ACTIVES.USER_LOGIN)	NN

Evidencia explícitamente la relación entre usuarios activos e inversionistas, donde cada uno puede ser empresa o no. Esta información, puede manejarse

desde la relación anterior, sin embargo, dados los privilegios que ser empresa corresponde, existe la pertinencia de manejar estos datos de esta manera.

LEGALS		
LEGAL_ID	**LEGAL_NAME**	**USER_LOGIN**
varchar(20)	varchar(20)	varchar(20)
PK	NN, UNIQUE	NN, FK(USERS.LOGIN)

La relación LEGALS, representa los representantes legales dado el usuario, se sabe que cada usuario debe tener un representante legal, y que un representante legal trabaja únicamente para un usuario activo.

OFFERANTS	
USER_LOGIN	**OFFERANT_TYPE**
varchar(20)	varchar(1)
PK, FK(ACTIVES.USER_LOGIN)	NN

La relación OFFERENATS, ayuda a entender la pertinencia de la tabla INVESTORS, ya que existen varios tipos de usuario no mutuamente excluyentes, la información sobre el tipo de usuario puede determinarse a través del login, pero es más claro separar a los usuarios en tablas según su tipo, es claro que esta decisión de diseño o afecta la forma normal de la tabla dada.

PORTFOLIOS		
PK_ID	**USER_LOGIN**	**RISK**
number(38, 0)	varchar(20)	varchar(1)
PK	FK(USER.LOGIN)	

La relación contiene información sobre el portafolio de cada usuario, es decir, la colección de valores que éste posee, para ello, posee información del riesgo, y el usuario correspondiente.

PORTFOLIOS_VALS		
PK_ID	**PK_PORTFOLIO**	**PK_VAL**
number(38, 0)	number(38, 0)	number(38, 0)
PK	FK(PORTFOLIO.PK_ID)	FK(VALS.PK_ID)

La relación PORTFOLIOS_VALS evidencia la colección de un portafolio con unos valores, es decir, los valores contenidos en un portafolio, lo cual corresponde al portafolio en su totalidad. Esta tabla es pertinente dada la multiplicidad de la cantidad de valores de un portafolio.

RENTS						
PK_ID	**RENT_NAME**	**DESCRIPTION**	**RENT_FN**	**RENT_LEN**	**RENT_TYPE**	**OFFERANT_LOGIN**
number(38)	varchar(20)	varchar(140)	varchar(1)	varchar(1)	varchar(1)	varchar(20)
PK	NN	NN	NN	NN	NN	FK(OFFERNATS) IN USER_LOGIN

La relación RENTS, representa un tipo de renta de un valor en Swap, este posee información sobre el nombre, descripción, función, tipo, y login del oferente.

VALS						
PK_ID	**VAL_NAME**	**DESCRIPTION**	**VAL_TYPE**	**AMOUNT**	**PRICE**	**RENT_ID**
number(38)	varchar(20)	varchar(140)	varchar(1)	number(38)	float	number(38)
PK	NN	NN	NN	NN	NN	FK(RENTS.PK_ID)

La relación VALS, representa un valor, posee información básica así como el tipo de renta (es decir un apuntador al registro real que representa la renta de este valor).

SOLICITUDES					
PK_ID	**REQUEST_TYPE**	**AMOUNT**	**AMOUNT_UNIT**	**TIMESTAMP**	**ACTIVE_LOGIN**
number(38)	varchar(20)	float	varchar(1)	timestamp	varchar(20)
PK	NN	NN	NN	NN	FK(ACTIVES.LOGIN)

La relación de solicitudes, contiene únicamente información acerca de la solicitud, incluyendo el usuario quién la realizó. Por último, SWAPTRANSACTIONS, posee información sobre las dos solicitudes y la fecha de creación.

SWAPTRANSACTIONS			
PK_ID	**CREATED_AT**	**PK_SOLICITUDE_1**	**PK_SOLICITUDE_2**
number(38)	timestamp	number(38)	number(38)
PK	NN	FK(SOLICITUDES.PK_ID)	FK(SOLICITUDES.PK_ID)

Con respecto al impacto de introducción de los nuevos requerimientos, a nivel de modelo y restricciones, puede verse que solo fue necesaria la inclusión de nuevas columnas a las tablas en la base de datos, para los parámetros de eliminación o adición de tuplas, se manejó las facilidades que brinda Oracle, o SQL en su versión estándar (DELETE ON CASCADE). Se manejó el archivo de creación sobre el ya existente. La diferencia se observa en los archivos de respaldo del esquema SQL (current.sql, para la versión actual, former.sql, para la versión anterior).

Si se compara el modelo anterior con el modelo actual, puede verse que no hubo grandes cambios, o más bien, los cambios que hubo no pedían harto esfuerzo.

b) Valide que su modelo se encuentra en BCNF y que no presenta anomalías de inserción, borrado o actualización con respecto a las reglas de negocio.

La justificación anterior, muestra que se encuentra en BCNF (Boyce-Codd normal form), sin embargo, se muestra la deducción formal del argumento. Para la validación práctica se realizan las pruebas automatizadas sobre los modelos relacionales.

Nótese que las dependencias funcionales, están determinadas por la llave seleccionada en cada caso. Se puede abstraer el concepto de dependencias, y se ve que cada asociación necesaria para esta iteración, es plasmada en una relación.

c) Documente la lógica de los nuevos requerimientos a desarrollar, descritos en la sección de caso de estudio de este documento. En este punto se requiere definir los mecanismos que utiliza para garantizar las propiedades ACID del requerimiento.

- Recomponer portafolio inversionista: Si el inversionista decide cambiar su inversión, se accede a los componentes atomizados de sus valores y

se recompone antes de una transacción, es decir, se cancelan esto, en caso de que otra transacción esté en curso con los mismos valores, esta, la del inversionista debe tener prioridad sobre la otra. Para esto, el inversionista la debe coger para actualizarla, (cada porción necesaria y pertinente del recurso). Asimismo, debe correr bajo la calidad de transacción.

- Retirar intermediario: Si el intermediario no desea trabajar mas con Swap, es necesario retirar las solicitudes correspondientes a dicho intermediario, para esto, es necesario ir a las solicitudes en estado actual y retirarlas. Es necesario que queden congeladas en su posibilidad para transacción, por lo que es necesario, además de retirarlas avisar a los usuarios involucrados en la transacción. Esto es a los oferentes, a los inversionistas, y retirar a los inversionistas del sistema.
- Los requerimientos de visualización requieren que la visualización de cada objeto por entidad sea mostrado al usuario final de la aplicación.

Implementación.

a) Ajuste las tablas creadas en Oracle de acuerdo a las decisiones del punto anterior.

   Ver SQL plus, o SQLDeveloper. Se incluyen las tablas que necesita el servicio web (web service, o web framework).

b) Pueble las tablas con información suficiente para poder realizar pruebas.

   - Diseñe los datos que le permitan verificar adecuadamente las reglas de negocio. Note que es más importante generar adecuadamente los datos, que obtener un número muy grande de ellos.

   - Puede escribir un programa de generación automática de datos acorde al diseño establecido para los mismos.

   - Para la población de las tablas utilice herramientas de carga masiva como SQLLoader o las disponibles en SQLDeveloper. Consulte el tutorial disponible en la wiki del curso sobre SQLLoader.

c) Ver código.
d) Verifique el comportamiento transaccional de los requerimientos que implican actualización, inserción o borrado de información.

   Interactuar

e) Diseñe un esquema de respaldo de datos. En máximo una página describa los procesos y la infraestructura que usted propone para que el negocio al cual corresponde la aplicación desarrollada pueda tener un esquema confiable de disponibilidad y recuperación de datos.

- Dado que la base de datos está presente en la asignada a uno de los dos integrantes, es necesario realizar un script de proceso de BACKUP a las dos bases de datos, es necesario sacar información del METADATA de Oracle acerca de los CONSTRAINTS de las tablas involucradas en la base de datos , una vez hecho esto, se puede realizar un diccionario que relacione el nombre de las tablas con cada CONSTRAINT. Luego, se extraen los datos y se realiza un script o programa que inserta los datos en la base de datos del otro integrante. El proceso se ilustra a continuación. Además, se propone un BACKUP automatizado haciendo uso de herramientas de Oracle a PostgreSQL. El BACKUP se debe hacer periódicamente.

```
Get METADATA from Oracle using current DB
Hash<String, Complex_Tuple_Object> <- Hash()
for each table in current DB do
    Hash.add(table, info)
    if Hash.len % 200 == 0 then
        save info to drive
        clean MEMORY PAR sleep(9000)

for each table in current DB do
    for each schema on table do
        write into file ("CREATE TABLE ....")
        using constraints on disks

for each table in current DB do
    execute("SELECT * " + table.name + ....)
    for each register sleeping and cleaning data
    periodically, while loading registers on disk

for each table in current DB do
    retrieve register by table in drive
    for each register in registers do
        execute("INSERT INTO " + table.name + ....)
        for each register sleeping and cleaning data
        periodically, while loading registers on disk
```

f) Implemente los escenarios de prueba para que le permiten asegurar el correcto funcionamiento de la aplicación y la corrección y calidad de los datos en la base de datos.

- En un archivo Excel documente claramente, para cada caso de uso, cuáles son los datos que le permiten realizar las pruebas, tanto para los casos de terminación exitosa como los fallidos. Indique cuáles son las respuestas esperadas que corresponden a los datos de prueba.

  Ver test automatizados. Los datos se muestran anteriormente.

Bono.

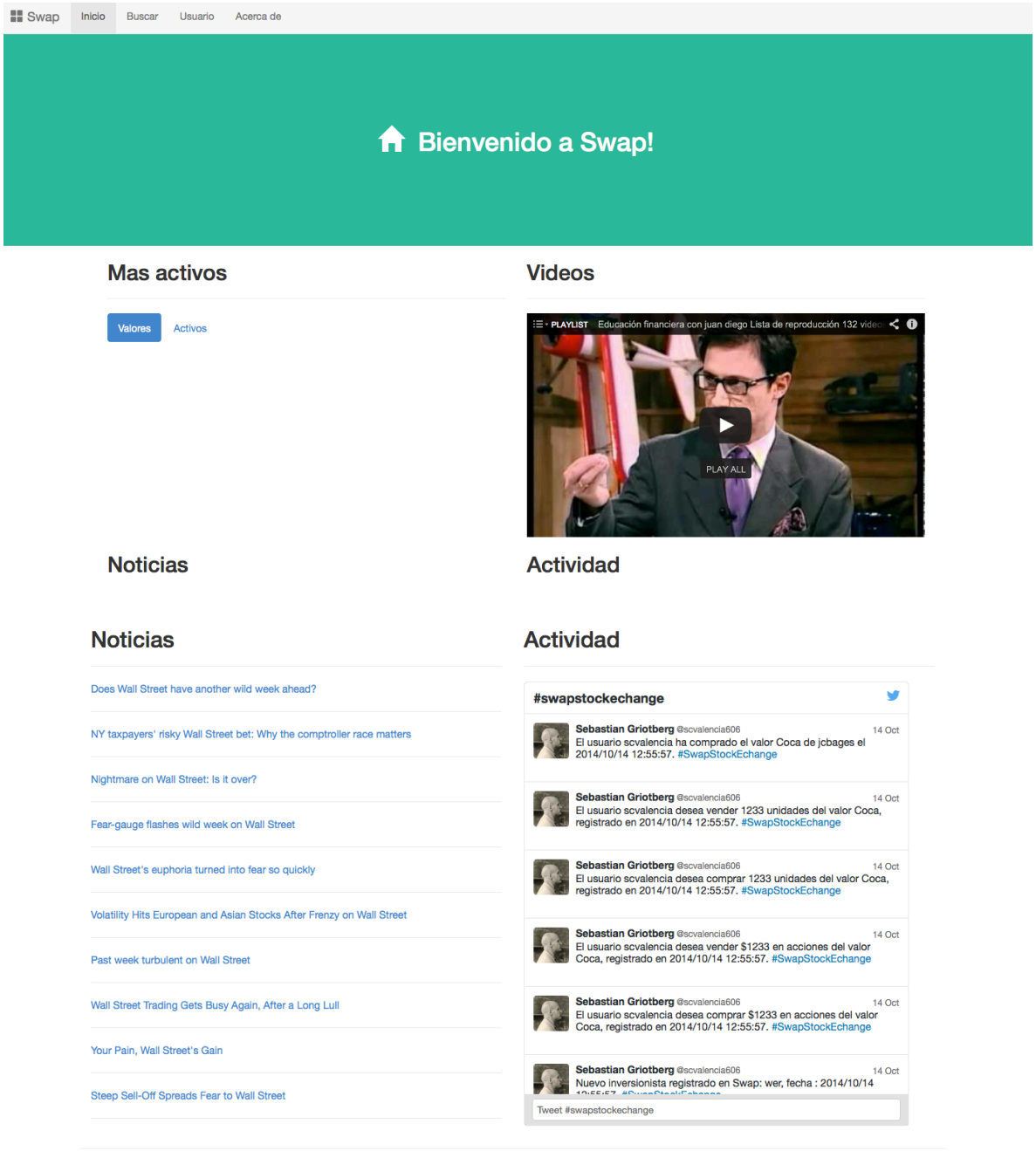
a) ¿Qué diferencias hay en el manejo transaccional por parte de un contenedor de aplicaciones con respecto al manejo dado por parte del programador de la aplicación?. Explique claramente las ventajas y desventajas de cada uno de ellos.

   Dado que el SW de bajo nivel juega directamente con el hardware (de naturaleza concurrente), es más fácil manejar la transaccionalidad de la aplicación por debajo de cuerda, o más bien jugando con la navegación de puertos, de procesos concurrentes que dependen de la disponibilidad del servidor. Asimismo, es más ligero conectarse por la disponibilidad de recursos.

   La mayoría de aplicaciones de este tipo son abiertas, por lo que están bajo constante revisión por parte del público general, lo que garantiza mayor revisión y curaduría del código.

   La concurrencia en el primer caso, es heredada de las capacidades del servidor, y garantiza mayor ligereza en la conexión oportuna con la base de datos, aislando por completo la ejecución de cada sesión más no los datos. Mientras el programador está expuesto a errores que maculen la transaccionalidad.

b)


Swap
Inicio
Buscar
Usuario
Acerca de
Bienvenido a Swap!
Mas activos
Valores
Activos
Videos
PLAYLIST
PLAY ALL
Noticias
Actividad
Noticias
Does Wall Street have another wild week ahead?
NY taxpayers' risky Wall Street bet: Why the comptroller race matters
Nightmare on Wall Street: Is it over?
Fear-gauge flashes wild week on Wall Street
Wall Street's euphoria turned into fear so quickly
Volatility Hits European and Asian Stocks After Frenzy on Wall Street
Past week turbulent on Wall Street
Wall Street Trading Gets Busy Again, After a Long Lull
Your Pain, Wall Street's Gain
Steep Sell-Off Spreads Fear to Wall Street
Actividad
#swapstockechange
Sebastian Griotberg @scvalencia606 14 Oct
El usuario scvalencia ha comprado el valor Coca de jcbages el 2014/10/14 12:55:57. #SwapStockEchange
Sebastian Griotberg @scvalencia606 14 Oct
El usuario scvalencia desea vender 1233 unidades del valor Coca, registrado en 2014/10/14 12:55:57. #SwapStockEchange
Sebastian Griotberg @scvalencia606 14 Oct
El usuario scvalencia desea comprar 1233 unidades del valor Coca, registrado en 2014/10/14 12:55:57. #SwapStockEchange
Sebastian Griotberg @scvalencia606 14 Oct
El usuario scvalencia desea vender $1233 en acciones del valor Coca, registrado en 2014/10/14 12:55:57. #SwapStockEchange
Sebastian Griotberg @scvalencia606 14 Oct
El usuario scvalencia desea comprar $1233 en acciones del valor Coca, registrado en 2014/10/14 12:55:57. #SwapStockEchange
Sebastian Griotberg @scvalencia606 14 Oct
Nuevo inversionista registrado en Swap: wer, fecha : 2014/10/14
Tweet #swapstockechange