PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

IIC2113 - Diseño Detallado de Software (2020-2)

Proyecto Semestral - Rúbrica Entrega 3

Objetivos

- Integrarse a un proyecto legado afrontando las diferencias en concepción sobre calidad de código.
- Aplicar los principios fundamentales, SOLID, CLEAN y otros temas vistos en la parte teórica del curso.

Plazo de entrega (revisión último commit): Jueves 3 de Diciembre a las 23:59.

Descripción

A tu grupo se le hará integrará a un repositorio con código de otros compañeros del curso. Esta vez trabajarás sobre el tema distinto al que eligieron inicialmente.

Entregable:

- Código de la aplicación en repositorio legado de Github asignado al grupo.
- En la primera línea del Readme.md el link a la aplicación levantada en el servidor elegido.
- Documentación en Readme.md del repositorio con información sobre cómo montar ejecutar la aplicación en local y sobre cómo montar o actualizar la aplicación en el servidor.
- Presentación con aprendizajes y dificultades encontradas.

Rúbrica:

Descripción	**Niveles de Desempeño**
Se integra correctamente con un API	30 puntos por correcta conexión -5 puntos por rutas hardcoded -10 puntos por no manejar errores por parte de la API 0 puntos por no integrarse
Lógica de automatización funcionando correctamente	30 puntos por lógica funcionando correctamente - 15 puntos si la lógica no funciona adecuadamente (-5 puntos por error o bug pequeño) 0 puntos por no automatizar
Uso de buenas prácticas de desarrollo	30 puntos - 5 puntos por cada error notorio acerca de SOLID, CLEAN y principios fundamentales. - 1 punto por cada code smell que afecta a la calidad de forma notoria.
Presentación con aprendizajes obtenidos	30 puntos
Total	120 puntos

Rúbrica entrega 3.

Bonus: Los bonus son acumulables entre entregas y se pueden usar de forma retroactiva. Los puntos son sobre la nota final, vale decir, en caso de obtener un 5.0 como nota de la entrega y lograr 1.5 puntos su nota de entrega será 6.5.

Bonus 1 [+1 punto]: Conectarse con su propia API. A forma de facilitar la integración se dispone de conexión con mockapi.
Bonus 2 [+0.5 puntos]: Se bonificará la entrega con 0.5 puntos sobre la nota obtenida si se hace uso adecuado de algún patrón de diseño. Para optar por el bonus deberá adjuntar a su repositorio un

archivo (pdf o md) que describa el patrón utilizado, señale el archivo y las líneas correspondientes, y que describa el valor que le entrega a su aplicación.

Para la presentación final:

La presentación debe durar entre 5 a 7 minutos, deben repartirse como:

- 2 minutos exponiendo los elementos más relevantes de su trabajo (en criterio del grupo). Estos pueden ser: alguna implementación que crean es destacable por su calidad, algún flujo de la aplicación levantada a modo de demo (**no** incluir login o funcionalidades evidentes/básicas para toda aplicación), algún diagrama que represente la estructura general, o cualquier elemento que demuestre dominio de lo estudiado a lo largo del curso.
- 3 minutos aterrizando problemas encontrados y ligandolos a materia del curso.

Luego de la presentación el equipo docente realizará una ronda de preguntas.

Criterio de evaluación	Puntaje máximo
Expone un aspecto relevante al trabajo y que se relaciona a la materia del curso: - Algún extracto de código o comparaciones entre códigos - Algún diagrama y su implementación - Algún patrón implementado - Alguna pull-request que demuestre lo aprendido - Algún flujo simple	10 puntos
Demuestra dominio de lo aprendido en el curso y presentado. Esto es, relaciona lo anterior a cualquiera de los siguientes temas: - Principios fundamentales - Buenas prácticas, code smells, refactoring, testing - Diagramación UML e implementación - Patrones de diseño - DDD, Clean, Paradigmas de programación, Ingeniería inversa	15 puntos
Aspectos prácticos de la presentación: - Uso del tiempo correctamente - Asertividad para comunicar - Preocupación en cómo mostrar la información	5 puntos

Bonus Presentación [+0.5 puntos]: Buen uso de recursos visuales. Esto es que el código o diagramas

presentados se entiendan bien durante la presentación. Casos en los que no se obtendría este bonus por ejemplo son código tamaño pequeño o diagramas en baja resolución.

Funcionalidades esperadas

Si en la primera y segunda entrega con mi grupo trabajamos en un **exchange**, para la tercera entrega trabajaremos sobre un **banco** y se espera:

- Cada 60 minutos el fondo del banco no debe crecer ni disminuir aleatoriamente. En su lugar, debe comprar o vender BTF en un exchange asignado.
    - El banco debe tener una cuenta en el exchange.
    - Si desde la api del exchange existe la posibilidad de arbitrar, el banco debe generar una orden de compra o venta en el exchange.
    - Por simplicidad el banco podrá tener CLP negativo en su cuenta de exchange para poder comprar o vender BTF (es una cuenta partner en el exchange).
- Para comprar debe usar el dinero que se encuentre en su fondo de inversión riesgoso.

Si en la primera y segunda entrega con mi grupo trabajamos en un **banco**, para la tercera entrega trabajaremos sobre un **exchange** y se espera:

- Los usuarios parten con 0 CLP.
- Para cargar CLP, los usuarios deben hacer una transferencia a una de las cuentas de Mercado Bitfake en los Bancos de Inversiones asignados.
    - Deben conectarse a la api de cada banco y revisar su cuenta empresa para ver si hay nuevas transacciones asociadas al RUT de ese usuario. En caso de existir una transacción (no registrada con anterioridad) debe contabilizar el depósito, aumentando así la cantidad de CLP en su cuenta.
- Cada 3 horas, el precio de venta o compra se debe disparar o caer de forma aleatoria.

ANEXO - TEMAS

Tema 1: Exchange de Bitfake

El Bitfake (BTF[1]) es una *criptomoneda* que ha ganado popularidad en el último tiempo. Debido a su popularidad, todos quieren tener un exchange de BTF-CLP.

Funcionalidades esperadas para Entrega 2:

- Debe permitir el registro y login de usuarios.
- Un usuario debe tener una cuenta.
    - Debe ver cuánto BTF y CLP tiene.
    - Puede ver sus transacciones hechas (compra y venta de BTF).
- Por simplicidad en esta entrega, todos los usuarios parten con un valor aleatorio entre los 10.000 CLP y los 100.000 CLP.
- Por simplicidad el valor inicial del BTF (o cualquier moneda) y la cantidad será definida por el ayudante al inicio del proyecto.
- Los usuarios podrán transar BTF, esto es:
    - Vender BTF si tienen en su cuenta. Al ejecutar una orden de venta de BTF, el usuario ingresa “cuánto BTF quiere vender” y el precio viene dado por el “precio de venta” del mercado.
    - Comprar BTF si es que tienen CLP. El ejecutar una orden de compra de BTF, el usuario ingresa “cuánto BTF quiere comprar” y el precio viene dado por el “precio de compra” del mercado.
    - Los precios de compra y venta serán rangos aleatorios cuyos mínimos y máximos iniciales serán distintos para cada grupo. Luego de cada transacción, el valor de compra y venta se debe actualizar con la siguiente lógica:
        - Si la transacción es venta, el precio del BTF de compra disminuye en 0.03 sobre el porcentaje de su valor.
        - Si la transacción es compra, el precio del BTF de venta aumenta en 0.03 sobre el porcentaje de su valor.
    - Por simplicidad, las transacciones no tienen delay.
    - Por cada transacción se debe enviar un correo al usuario.
    - Por simplicidad la cantidad de BTF es constante (ver bonus).

[1] No confundir con Bitcoin Faith.

- Debe permitir el tipo de cuenta “partner”. Este tipo de cuenta puede vender o comprar BTF sin límite (tener saldos negativos) ya que al ser partner, hacen ajustes de cuentas de forma externa (funcionalidad pensada para simplicidad en la entrega 3).
- Debe exponer una API. Cada usuario puede generar su “token” para usar la api. Los usuarios a través de la API podrán:
    - Obtener los precios actuales de compra y venta en el exchange.
    - Vender BTF (ejecutar una orden de compra).
    - Comprar BTF (ejecutar una orden de venta).
    - Conocer el balance actual de mi cuenta (cuanto BTH y CLP tengo).

Bonus: Registrar una segunda moneda: Badtherium (BTH[2]), que permite compra y venta de BTH-CLP.

Funcionalidades esperadas para Entrega 3:

- Los usuarios parten con 0 CLP.
- Para cargar CLP, los usuarios deben hacer una transferencia a una de las cuentas de Mercado Bitfake en los Bancos de Inversiones asignados.
    - Deben conectarse a la api de cada banco y revisar su cuenta empresa para ver si hay nuevas transacciones asociadas al RUT de ese usuario. En caso de existir una transacción (no registrada con anterioridad) debe contabilizar el depósito, aumentando así la cantidad de CLP en su cuenta.
- Cada 3 horas, el precio de venta o compra se debe disparar o caer de forma aleatoria.

Ejemplo de una api de un exchange: https://api.buda.com/

[2] No confundir con Bithereum.

Tema 2: Banco de Inversiones

Se te pide crear la aplicación de un banco que además posee un fondo de inversión riesgoso.

Funcionalidades esperadas para Entrega 2:

- Debe permitir el registro y login de usuarios.
- Un usuario debe tener una cuenta.
    - Puede ver cuánto CLP tiene en su cuenta y cuánto tiene en Ahorros.
    - Puede ver las transacciones hechas (transferencias, depósitos o ahorros).
- Por simplicidad en esta entrega, todos los usuarios parten con un valor aleatorio entre los 10.000 CLP y los 100.000 CLP.
- Los usuarios podrán transferir dinero entre cuentas del mismo banco. Para transferir dinero debe conocer el número de cuenta del receptor. Así mismo al hacer una transferencia, el otro usuario recibirá un depósito.
    - Para poder efectuar la transferencia, la página debe validar que el usuario ingrese el mismo código que le llegará a su correo al momento de solicitar transferir.
    - Mientras no ingrese el código, la transferencia no se realizará.
- El banco tiene un módulo de inversiones. El módulo de inversiones es un fondo de inversión riesgoso que el banco usa para invertir y así generar ganancias (o pérdidas) proporcionales para todos sus usuarios dependiendo del monto que ha ahorrado.
    - Un usuario puede enviar dinero desde su cuenta al fondo de inversiones. Se registra este valor como ahorro.
    - El banco usa este fondo para invertir. Para efectos de esta entrega, y por simplicidad, este “invertirá 60 minutos”. Esto se traduce en que el fondo crecerá o disminuirá en un factor aleatorio entre [-0.03, +0,03] porcentaje del valor.
    - Si el fondo gana o pierde, se deberá ver reflejado no solo en el fondo total, si no que también en las cuentas de ahorro individuales de cada usuario.
- Debe exponer una API. Cada usuario puede generar su “token” para usar la api. Los usuarios a través de la API podrán:
    - Obtener las transacciones de la cuenta del usuario.
        - Todas.
        - En un período de tiempo.

Bonus: Registrar un segundo fondo de inversión. Los usuarios pueden destinar sus ahorros a un segundo fondo un poco más conservador, cuya lógica de inversión implica invertir solo la mitad del fondo ahorro (la otra mitad se mantiene sin invertir).

Funcionalidades esperadas para Entrega 3:

- Cada 60 minutos el fondo del banco no debe crecer ni disminuir aleatoriamente. En su lugar, debe comprar o vender BTF en un exchange asignado.
    - El banco debe tener una cuenta en el exchange.
    - Si desde la api del exchange existe la posibilidad de arbitrar, el banco debe generar una orden de compra o venta en el exchange.
    - Por simplicidad el banco podrá tener CLP negativo en su cuenta de exchange para poder comprar o vender BTF (es una cuenta partner en el exchange).
- Para comprar debe usar el dinero que se encuentre en su fondo de inversión riesgoso.

Ejemplo de una api bancaría: https://fintoc.com/docs#introduccion