MÓDULOS - MICROSERVICIOS

En un inicio se van a incluir todos los endpoints en un mismo proyecto. Después la idea será separar los endpoints en proyectos diferentes para evitar modificar cosas de otros servicios.

Los endpoints los tendremos que agrupar de forma ordenada y no entremezclada en los siguientes servicios / módulos / secciones que hay también aparecen etiquetados por nivel de criticidad siendo los N2 los servicios críticos.

Fase 1

- user (userinfo and appSetings) (N1)
- subir sonidos (N1)
- charts y valores (N2)
- orders and order executions (N2)
- admin (N2)
- depósitos (dinero demo)

Fase 2

- depósitos y retiradas (dinero real) (N2)

Nota:

Hay endpoints que ya están hechos en el proyecto anterior (marcados con (H)) pero no están en la carpeta API. Están como archivos sueltos en formato .php, se debe hacer la implementación del endpoint en la carpeta API reutilizando el código ya hecho que debe estar funcionando.

Los endpoints que aparecen con la URL local en este archivo están ahora mismo hechos un mocks hardcodeados en el Back, hay que hacer su implementación.

Notas:

En todas las URLS tienen son llamadas mediante '/clusterbeat/API/index.php' y se le pasan por parámetros dos propiedades, a (indica el endpoint que vamos a utilizar) y versión que indica la versión de la API, esta define la cantidad de endpoints

USER

getUser
URL: https://localhost/clusterbeat/API/index.php?a=getUser&version=v2&userId=20
Tipo: GET
Descripción:
Devuelve los datos del usuario asociado al id recibido.
Input parameters:

Atributo	Tipo	Descripcion	Opcional
userId	Long		No

Output parameters:

Atributo	Tipo	Descripcion	Opcional
username	String		
email	String		
pro	String		
premium	String		
firstname	String		
lastname	String		
genero1	String		
genero2	String		
descargas	Int		
registerdate	Date		
lastlogindate	Date		
penultimatelogindate	date		
role	String		
language	String		
packuploaded	String		

fullinitsteps	String		
skin	String		
urlAvatar	String		

Register
URL: https://localhost/clusterbeat/API/index.php?a=signUp&version=v2
Tipo: POST
Descripción:
Sirve para registrar un usuario en la web.
Input parameters:

Atributo	Tipo	Descripción	Opcional
user	String		No
email	String		No
pass	String		No

Login
URL: https://localhost/clusterbeat/API/index.php?a=login&version=v2
Tipo: POST
Descripción:
Sirve para iniciar sesión en la web, retorna id, token jwt y exp que es un timestamp.
Input parameters:

Atributo	Tipo	Descripción	Opcional
userEmail	String		No
pass	String		No
isPin	Boolean	Indica si el atributo pass es la contraseña del usuario o si es el pin	No

Output parameters:

Atributo	Tipo	Descripción	Opcional
userId	Long		
token	String		
exp	Timestamp	Indica cuando vence el token	

roles	Role[]	Array con los roles del usuario	

Role Object

Atributo	Tipo	Descripción	Opcional
name	String	Nombre del role asociado al usuario	

Reset Password
URL: https://localhost/clusterbeat/API/index.php?a=resetpass&version=v2
Tipo: POST
Descripción:
Sirve para resetear la contraseña de un usuario ya registrado.
Input parameters:

Atributo	Tipo	Descripción	Opcional
email	String		No

SUBIR SONIDOS y DISPONIBLES

Available Assets
URL: https://localhost/clusterbeat/API/index.php?a=availableAssets&version=v2
Tipo: GET
Descripción:
Devuelve todas las librerías disponibles
Output parameters:

Atributo	Tipo	Descripción	Opcional
currentPrice	Double		
genre	String		
id	Long		
imgurl	String		
symbol	String		
variationPercent	String		

User Wallet
URL:https://localhost/clusterbeat/API/index.php?a=userWallet&version=v2
Tipo: GET
Descripción:

Devuelve todas las librerías disponibles
Input parameters:

Atributo	Tipo	Descripción	Opcional
userId	Long		No

ORDERS AND ORDER EXECUTIONS

setOrder
URL: https://localhost/clusterbeat/API/index.php?a=setOrder&version=v2
Tipo: POST
Descripción:
Sirve para crear una orden, sea de compra o de venta
Input parameters:

Atributo	Tipo	Descripción	Opcional
type	String	Indica si la orden es de tipo buy o sell. Valores válidos: buy - sell	No
userId	Long		No
amount	Int		No
idLibrary	Long		No
limitOption	Int		No
limitOrder	Int		No
numberOrder	Int		No
dayOrder	Int		No

executePendingOrders
URL: https://localhost/clusterbeat/API/index.php?a=executePendingOrders&version=v2
Tipo:GET
Descripción:
Hace un match de las órdenes guardando las órdenes ejecutadas

getPendingOrders
URL:https://localhost/clusterbeat/API/index.php?a=getPendingOrders&version=v2
Tipo: GET
Descripción:

Retorna una lista de las órdenes pendientes de una librería y usuario en específico, estas pueden ser de tipo venta o compra.
Input parameters:

Atributo	Tipo	Descripcion	Opcional
userId	Long		
libraryId	Long		

Output parameters:

Atributo	Tipo	Descripción	Opcional
orderId	String		
individualPrice	double		
totalPrice	double		
creationDate	Date		
orderOfDay	Int		
userId	Long		
libraryId	Long		
type	String	Indica el tipo de orden. Valores válidos: buy - sell	
numberOfOrders	Long		

getAllPendingOrders
URL:https://localhost/clusterbeat/API/index.php?a=getAllPendingOrders&version=v2
Tipo: GET
Descripción:
Retorna una lista de las órdenes pendientes, estas pueden ser de tipo venta o compra.
Output parameters:

Atributo	Tipo	Descripción	Opcional
orderId	String		
individualPrice	double		

totalPrice	double		
creationDate	Date		
orderOfDay	Int		
userId	Long		
libraryId	Long		
type	String	Indica el tipo de orden. Valores válidos: buy - sell	
numberOfOrders	Long		

getExecutedOrders (HISTORIAL)
URL:https://localhost/clusterbeat/API/index.php?a=getExecutedOrders&version=v2
Tipo:GET
Descripción:
Retorna una lista de órdenes ejecutadas, tanto de compra como de ventas.
Output parameters:

Atributo	Tipo	Descripción	Opcional
orderId	Long		
totalPrice	Double		
creationDate	Date		
executionDate	Date		
orderOfDay	Int		
libraryId	Long		
userId	Long		
type	String	Indica el tipo de orden. Valores válidos: buy - sell	
numberOfOrders	Int		
idUserExecute	Long		

DEPÓSITOS

addDemoDeposit
URL: https://localhost/clusterbeat/API/index.php?a=addDemoDeposit&version=v2

Tipo:POST
Descripción:
Carga dinero demo a la cuenta del usuario
Inputs parameters:

Atributo	Tipo	Descripción	Opcional
userId	Long		No
money	Double		No

CHARTS Y VALORES

getItemChartData
URL:https://localhost/clusterbeat/API/index.php?a=getItemChartData&version=v2&idLibrary=$idLibrary
Tipo:GET
Descripción:
Carga dinero demo a la cuenta del usuario
Inputs parameters:

Atributo	Tipo	Descripción	Opcional
idLibrary	Long		No

Output Parameters:

Atributo	Tipo	Descripción	
id	Long		
library_id	Long		
timestamp	Date		
current_price	Double		
demand_price	Double		
demand_volume	Int		
offer_price	Double		
offer_volumen	Int		
open_price	Double		
close_price	Double		
max_price	Double		
variation_percent	Double		

volume_today	Int		
volume_last_seven_days_average	Int		
volume_last_three_month_average	Int		
min_price	Double		

USERS

/register (P1) (H) R

Funcionando en el API hay que acomodar lo recibido en el front para dar bien el mensaje.

/login (P1) (H) R (Ya funcionado e implementado)

https://localhost/clusterbeat/API/index.php?a=signUp&version=v2

/resetpass (P2) (H?)

https://localhost/clusterbeat/API/index.php?a=appSettings&version=v2

Devuelve atributos relacionados con la configuración de la app. Si hay que añadir atributos consultar.

https://localhost/clusterbeat/API/index.php?a=userWallet&version=v2

Devuelve la cartera del usuario. Los elementos que tiene comprados y las unidades y precio correspondiente. También debe devolver el saldo demo y real que tiene el usuario.

SUBIR SONIDOS y DISPONIBLES

En el proyecto ya existe este endpoint, no es necesario modificarlo.

https://localhost/clusterbeat/API/index.php?a=availableAssets&version=v2

Debe obtener los datos de los sonidos (activos financieros) que están disponibles para su compra / venta (trading)

CHARTS Y VALORES

https://localhost/clusterbeat/API/index.php?a=assetDynamicDetails&version=v2&idPack=1

Debe obtener los datos de un sonido (activo financiero) a lo largo del tiempo para completar el gráfico de precios visible en la aplicación.

Los atributos que debe devolver son los necesarios para hacer el input de datos en el componente del gráfico.

/getItemChartData/:id

No sé si está duplicado con el anterior.

/recalculateAssetPrice

Este endpoint se ejecutará mediante un cron para cada sonido (activo financiero) cada cierto tiempo y debe hacer el cálculo del nuevo precio para el timestamp actual en tiempo UTC.

Guardará en una tabla los registros de precios calculados,tendrá en cuenta el libro de órdenes, las compraventas y otros factores.

NO HACER - A DEFINIR EL ALGORITMO

ORDERS AND ORDER EXECUTIONS

/setOrder

Crea una nueva orden de compra o de venta a precio de mercado.

Cuando se crea una nueva orden se agrega a una base de datos de órdenes a realizar (libro de órdenes) y se quedan pendientes hasta que se ejecute el match compra - venta.

/executePendingOrders

Este endpoint contendrá un algoritmo para encontrar compradores a los vendedores.

Fase 1 - Reparto inicial

Cuando se dé de alta un nuevo sonido (activo financiero) se permitirá comprar a todos los interesados. Hasta que se haga un reparto de unidades compradas a definir.

A partir de que se alcance ese número de reparto de compras este endpoint que se ejecutará mediante un cron cada cierto tiempo tendrá que matchear órdenes de compra venta para encontrar compradores a los vendedores.

Tanto las órdenes de compra como de venta podrán ejecutarse por tramos según se vayan haciendo match.

Una ejecución por tramos es cuando hay una orden de venta de 100 unidades pero de compra hay de otras cantidades por ejemplo 4 órdenes de compra de 25 unidades.

La orden de venta de 100 unidades se ejecutará en 4 tramos y posiblemente a diferentes precios.

Las órdenes de compra y de venta de mayor volumen tendrán prioridad respecto a órdenes de menor volumen.

DÉPOSITOS - Dinero Demo - Dinero real

/addDemoDeposit

Agrega la cantidad de dinero demo recibida por parámetro a un usuario

/addRealDeposit (no hacer)

/realWithdraw (no hacer)

ADMIN - NO HACER

En la fase 2 con pagos reales se implementará una pantalla para confirmar los depósitos de dinero.

/getDepositRequests

Obtendrá un listado de peticiones de depósito

/confirmDepositRequest

Confirmará que un déposito ha sido realizado. sumará en una unidad de confirmación para ese depósito.

/denegateDepositRequest

Denegará un depósito.

/evaluateConfirmationRequests

Evaluará el número de confirmaciones hechas y superado un número preventivo a definir, la cuenta del usuario tendrá que ver reflejado el ingreso. Previo guardado del dato en la BBDD.

Cuando se tengan todas las tablas creadas se autogenerará un panel de administración con el framework Yii que crea un CRUD completo para las tablas de forma automática.

/recalculateAssetPrice
Dependiendo de la oferta y la demanda de las participaciones va a variar el precio. También va a afectar el precio de las participaciones externas, por ejemplo en spotify, youtube, listas de popularidad, etc. Para hacer esto habría que hacer web scraping.

CASOS DE USO DE TESTS

Todos los test deben comprobarse tanto en web, como en una app instalada en un hardware.

Proceso:

Al iniciar un testing antes de cerrar una versión se debe copiar la siguiente plantilla en un nuevo documento de word.

Se deben ejecutar todos los casos de test descritos a continuación y sobre el nuevo documento word se realizará un informe coloreando en verde los test OK tanto en web

como en app. En amarillo los que no funcionen en alguna de las dos plataformas acompañado de un comentario con el problema. Al detectarse el problema se debe abrir un ticket en trello en la columna “bugs” para que se pueda corregir. Los puntos que no funcionen de ninguna manera se marcarán en rojo y se abrirá también el ticket detallando un comentario en ambos.

0 - General

0.1 - Los iconos de instagram están ocultos.

0.2 - Se comprueba que todas las páginas son responsive:

Comprobar que cada página se ve bien en:

a) web
b) tablet
c) móvil

0.3- Notificaciones push

Se comprueba que se pueden enviar notificaciones push correctamente con el texto, descripción y redirección marcada y que lleguen de forma correcta.

1 - user (userinfo and appSetings)

1.1 - Como usuario puedo registrar una nueva cuenta.

Y

a) solo se puede registrar el mismo email una sola vez.

b) al completar el formulario de registro hay otro formulario que pide el número de teléfono

c) el formulario al completarse con errores tiene validación de formato email y de contraseñas equivalentes y muestra los errores correspondientes al completarse con combinaciones incorrectas.

d) al completar el registro los datos están guardados en la tabla de usuarios y por defecto en el atributo de “pro” de la tabla es 0 que indica que es una cuenta demo. (todos los usuarios deben iniciarse con el 0)

1.2- Como usuario puedo hacer log out de una cuenta y al volver atrás en las páginas o cargar páginas por url no se puede ver la información del usuario anterior o directamente no se puede entrar en las páginas sin estar logado.

1.3- Como usuario puedo hacer log in con los credenciales de una cuenta creada previamente.

1.4- Como usuario, cuando intento hacer log in con unos credenciales inválidos obtengo los errores correctos.

1.5- Como usuario, cuando intento registrar un correo ya registrado anteriormente obtengo el mensaje de error correspondiente.

1.6- Como usuario puedo resetear mi password. Al intentar resetear el password se introduce el correo y debe de llegar un mail de confirmación (cuando el back esté desplegado)

1.7- Como usuario, cuando entro en la aplicación veo en la interfaz los datos asociados al endpoint userinfo de forma correcta y equivalente a lo que hay en bbdd para ese mismo usuario.

1.8 - Se recibe la información de appSettings de forma correcta equivalente a lo guardado en BBDD.

1.9 - Como usuario cuando intento cambiar de idioma veo que se han cambiado de idioma los textos (que no vienen de backend) de todas las páginas de la aplicación. títulos, textos, botones..

1.10 - Como usuario cuando hago click en el link de la navbar de “¿Cómo funciona?” veo que la aplicación navega a una página donde veo un resumen de la funcionalidad de la aplicación. (por definir los textos por ahora poner un texto de ejemplo lorem ipsum)

1.11- Como usuario cuando hago click en el link de “Terminos y condiciones” localizado tanto en el formulario de registro como en el footer de la aplicación veo que contiene toda la información relacionada al uso de la aplicación con todas sus secciones y apartados.

Siguiendo este documento: (por adjuntar) o equivalente a los términos y condiciones de la versión anterior.

2 - subir sonidos

2.1- Como usuario no administrador si intento acceder al panel de administración para subir sonidos no es posible acceder.

2.2- Como usuario administrador si intento acceder al panel de administración para subir sonidos es posible acceder.

2.3 - Como usuario administrador, cuando completo el formulario para subir nuevos sonidos y los subo estos se suben sin problema y después se ven reflejados en la lista de disponibles (POR MEJORAR FLUJO)

3 - charts y valores

3.1 - Como usuario cuando entro en la página de un sonido puedo visualizar la información correcta correspondiente a ese sonido/librería/track y debe verse:

- la información correspondiente de ese sonido en particular.
- El vídeo embebido de youtube de ese sonido en particular y que se reproduzca.
- Los datos del gráfico y con los datos correspondientes en diferentes puntos de tiempo que se ajustan a los guardados en la base de datos para ese sonido en específico y no otros.
- El precio marcado es el precio actual.
- Los datos de información de producto es la correcta
- La imagen de la carátula es la correcta y tiene el fondo con la carátula correspondiente con el degradado.
- Todos los elementos son visibles en la página.
- Cuando los textos de la información bajo la fotografía son muy largos se ven correctamente y no se superponen o descolocan en ningún tipo de tamaño de pantalla.

4 - orders and order executions

Definición de “compras no asociadas a vendedor” - es una variable que puede ser diferente para cada librería y es un número entero con el que se indica la cantidad máxima de compras que pueden ser ejecutadas para un track sin que haya un vendedor al que el comprador compra las participaciones.

Cuando el número de participaciones compradas supera ese número todas las compra-ventas se ejecutarán siempre que haya una contraparte interesada, en específico un orden de compra o de venta que se complementen.

Definición de “Supply Total” - Es el número máximo de participaciones que podrá haber en circulación.

Definición de “Supply en circulación” - Es el número de participaciones que hay en circulación en un momento dado.

- **4.1 Órdenes de compra**
- 4.1.1 Como usuario, cuando intento crear un órden de compra y cuando la cantidad de compras realizadas no supera el número asignado de “compras no asociadas a vendedor” tanto el frontend como el backend por separado comprueban que:

a) El usuario dispone en cuenta del dinero para efectuar la órden.

   En caso de tener disponibilidad se debe comprobar que la orden se ha guardado correctamente en la BBDD.

   Al modificar el input de la cantidad de participaciones se debe mostrar el número de dinero que multiplicado por el precio actual corresponde y en caso de que supere el saldo en cuenta se mostrará el input del número de participaciones como erróneo con el borde en rojo. Además la cifra con el total calculado también aparecerá en rojo y también aparecerá un pequeño mensaje en rojo informando con el texto “Esta cifra supera su saldo en cuenta”

   Si no lo cumple se debe mostrar un error y no guardar la orden en BBDD.

b) Si el número de “compras no asociadas a vendedor” ya se ha superado. La órden se guardará pero no se ejecutará hasta encontrar un vendedor.

Se notifica al usuario mediante un toast con éxito (verde) o error (rojo) si la petición de órden de compra se ha ejecutado con éxito o no.

- **4.2 Órdenes de venta**
- 4.2.1 Como usuario, cuando intento crear una órden de venta tanto el frontend como el backend por separado comprueban que:
  a) El usuario dispone de la cantidad de participaciones que quiere vender.

     Si dispone de la cantidad se guardará la orden en BBDD y se mostrará el toast de notificación verde de éxito.

     Si NO dispone de la cantidad habrán dos comprobaciones:

     a) el frontend no debe dejar escribir en el formulario un número mayor al de la disponibilidad o debe colorearse en rojo el borde del campo al introducir un número incorrecto.
     b) En el caso de enviar al backend un número incorrecto éste devolverá un error detallado y el front notificará del error al usuario con el toast rojo de error.

- **4.3 Resumen de órdenes pendientes de ejecución**

  4.3.1 Cuando accedo a la página de “mi cartera” aparte de ver un resumen de la cartera del momento actual puedo ver un resumen con las órdenes pendientes. Y puedo comprobar que:

a) cuando creo una orden de compra aparece en el listado del resumen de órdenes pendientes de ejecución y además los datos coinciden. El track, y el número de participaciones y el tipo de la órden.
b) cuando creo una orden de venta aparece en el listado del resumen de ordenes pendientes de ejecución. El track, y el número de participaciones y el tipo de la órden.
c) cuando una orden de compra se ejecuta desaparece del listado de resumen de órdenes pendientes de ejecución.
d) cuando una orden de venta se ejecuta desaparece del listado de resumen de órdenes pendientes de ejecución.

- **4.5 Ejecución de órdenes pendientes**
  a) Se comprueba que del listado completo de órdenes se ejecutan antes las órdenes de COMPRA de mayor tamaño de participaciones.
  b) De la misma forma se comprueba que del listado completo de órdenes se ejecutan antes las órdenes de VENTA de mayor tamaño de participaciones.
  c) Se comprueba la ejecución de las órdenes por tramos:

     Si hay una orden de COMPRA de 10 participaciones pero solo hay órdenes de VENTA de 5 participaciones. La orden de compra se ejecutara en 2 tramos haciendo match con dos vendedores diferentes. Las órdenes de los vendedores serán ejecutadas en un solo tramo.

     Lo mismo se debe cumplir al contrario con órdenes de venta.

     Si hay una orden de VENTA de 10 participaciones pero solo hay órdenes de COMPRA de 5 participaciones. La orden de venta se ejecutara en 2 tramos haciendo match con dos compradores diferentes. Las órdenes de los compradores serán ejecutadas en un solo tramo.

- **4.6 Historial de transacciones**

  El usuario puede acceder a su historial de transacciones ejecutadas en los últimos 3 meses a través de un botón de tipo texto situado cerca del listado de órdenes pendientes. Es un listado en que que se indica: tipo de órden (compra o venta), nombre librería o símbolo, fecha, y cantidad ingresada (ejemplo +34€) en color verde o cantidad descontada (ejemplo -34€) en color rojo. Cuando una órden se ejecuta por tramos aparecerán tantas líneas de transacción como tramos de ejecución.

5 - admin

POR DEFINIR

- **6 - depósitos (dinero demo)**

6.1 - Como usuario, cuando registró un nuevo usuario, veo la pantalla para iniciar la cuenta demo y comprar uno de los cuatro paquetes de saldo.

6.1.1 Y además veo el botón de skip para saltar el paso. cuando hago click en el botón se redirige a la página de trading donde se muestran todos los disponibles.

6.2. Como usuario, cuando selecciono cualquiera de los paquetes de saldo y continuo me llevan a la pasarela de pago para hacer el pago. (comprobar con todos los paquetes)

6.3 - Como usuario cuando termino de realizar el pago al volver a la plataforma tengo el nuevo ingreso de saldo demo en la cuenta correspondiente al paquete de saldo que he pagado.

6.4- Como usuario cuando hago click en el botón deposit, si tengo una cuenta demo (campo de la tabla usuarios “pro” como 0) me debe redireccionar a la misma pantalla de compra de los paquetes de saldo demo. Y deben ejecutarse bien los puntos anteriores del apartado 6.

6.5- El dinero demo de cada usuario debe estar indicado en un atributo dedicado en la tabla de usuarios “demoMoney” acompañado de un campo “demoMoneyCurrency” (por defecto eur)

Fase 2

- depósitos y retiradas (dinero real) (N2)

Test de Login