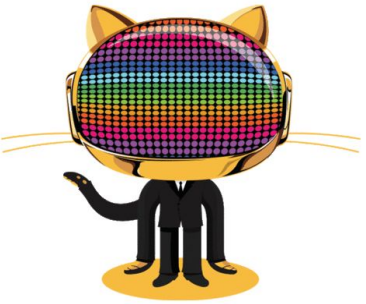
Hitbug

Share your music, share your history, share your emotions…



“La idea era que se llamara Hitbag, ya que tenés toda tu información musical en Bags, pero al registrar el dominio hubo un bug se ve…”, nos cuenta entre risas el creador de esta novedosa red social musical que busca revolucionar la vida de las personas dejando rastro de su actividad musical.

Dominio

Según nos cuenta su creador, la idea de esta nueva plataforma es muy simple: Poder hacer listas de reproducción (playlists) colaborativas, e ir almacenando su historia. A cada una de estas listas, se las ha dado en llamar Bags.

Un Bag puede contener diversos tipos de contenido: Imágenes, video, texto y música. Los videos, imágenes y canciones tienen asociados un nombre o título, una descripción (puede estar vacía) y una URL al contenido multimedia. Los textos, simplemente tienen un título y su contenido.

Asimismo, un Bag también puede incluir todo el contenido de otro Bag, referenciándolo: si el contenido del Bag referenciado cambia, estos cambios deben reflejarse en el Bag que lo referencia de forma transparente.

Todos los Bags pertenecen a cierto usuario, que puede editar su contenido, y tienen a su vez una lista de colaboradores que también pueden hacerlo. Cada cambio realizado sobre un Bag es atómico y se lo denomina “Hit” (de forma similar a los sistemas de versionado). Editar el contenido involucra agregar o remover contenido, o editar un específico contenido. Por ejemplo, un Hit puede contener el agregado de una canción y el cambio del título de una imagen.

Dado que es necesario poder conocer el estado de un Bag en el pasado, debe ser posible navegar el historial de Hits realizados y situarse en uno en particular, viendo al Bag como era en ese entonces.

Finalmente, los colaboradores también pueden hacer “Hit Requests”: un Hit Request es un Hit sugerido que no se aplica efectivamente hasta tanto el dueño no lo apruebe, es decir que puede estar pendiente, rechazado o aprobado.

Requerimientos

Los requerimientos concretos son:

1) Un Bag debe poder:
    a) almacenar cualquier contenido multimedia
    b) referenciar el contenido de otros Bags

2) Un usuario debe poder:
    a) crear un Bag y ver el listado de su contenido (tanto propio como referenciado)
    b) añadir y remover colaboradores de sus Bags

c) realizar un Hit con cambios en un Bag que posee
d) realizar Hits en un Bag del que es colaborador
e) realizar Hit Requests en un Bag del que es colaborador
f) aprobar o descartar Hit Requests en un bag que posee
g) ver el estado de un Bag en el pasado, parado en un cierto Hit. Por ejemplo, ver cómo estaba el Bag "Favoritos" cuando se hizo la modificación "Agrego la película de Bob Esponja"
h) deshacer los últimos N Hits o aplicar un Hit Request rechazado

- Se pide diseñar una solución que resuelva los requerimientos planteados y comunicarla utilizando prosa, diagramas, pseudocódigo,etc.
- Se pide, asimismo para el último requerimiento (2-h), diseñar una solución alternativa y compararla con la primera según su ***simplicidad*** y otra cualidad de diseño elegida.
- Bajo la solución que propongas, si tenemos un Bag "Rock Nacional" y otro "Canciones Bizarras" el cual contiene el tema "Solo le pido a Dios":
  - ¿puede un mismo Hit *mover* "Solo le pido a Dios" a "Rock Nacional"?
  - Si es posible, contar por qué. Si no, explicar a alto nivel qué cambios sería necesario hacer a la solución para soportarlo

Code Smells

Una consultora de la India implementó el siguiente código:

```
class Cancion
  attr video, album, artista, titulo

  def initialize(video, album, artista, titulo)
    album.addCancion(self)

  def vistaPrevia
    if(album.artistas.contains(artista))
      return Preview.new(self)
    else
      throw Exception.new("El artista ingresado es incorrecto")
    end
end
```

```
class Album
  attr canciones, artistas, titulo,tapaDeAlbum, listaDeCanciones,contraTapa

  def serEscuchado
    artistas.map{ artista =>
      if(artista.albumesCreados.contains(self))
       try
         artista.discografica.pagar(100)
       catch NullPointerException
         //entonces es un artista independiente
         artista.pagarArtistaIndependiente(100)
       end
       TopLists.instance.recientementeEscuchados.push(artista)
      else if(artista.albumesParticipados.contains(self.titulo))
        TopLists.instance.recientementeEscuchados.push(artista)
    }
  end
```

```
  def vistaPrevia
    //Para que no tire la excepcion
    return canciones.filter(artistas.contains(cancion.artista)).map(vistaPrevia)
  end
end
```

```
class Artista
  attr albumesCreados, albumesParticipados

  def crearAlbum(titulo, colaboradores, tapaDeAlbum, listaDeCanciones,contraTapa)
    if(!albumesCreados.any(a => a.titulo == titulo))
      albumesCreados.add(
        Album.new(titulo,self,colaboradores, tapaDeAlbum, listaDeCanciones, contraTapa))
      colaboradores.foreach{ col =>
        if(!col.albumesParticipados.any(title => title == titulo))
          colaborador.albumesParticipados.add(titulo)
        else
          throw new AlbumExistenteException
      }
    else
      throw new AlbumExistenteException
    administrador.notficarAlbumCreado()
  end
end
```

- Se pide identificar al menos 5 Code Smells distintos en el código y proponer a alto nivel refactors para solucionar cada uno de ellos o justificar por qué no representan un problema de diseño, si fuera el caso. No es necesario marcar todas las diferentes apariciones de un mismo Code Smell.