Informe Bandu

Integrantes: Vicente Salazar y Sebastian Sandoval

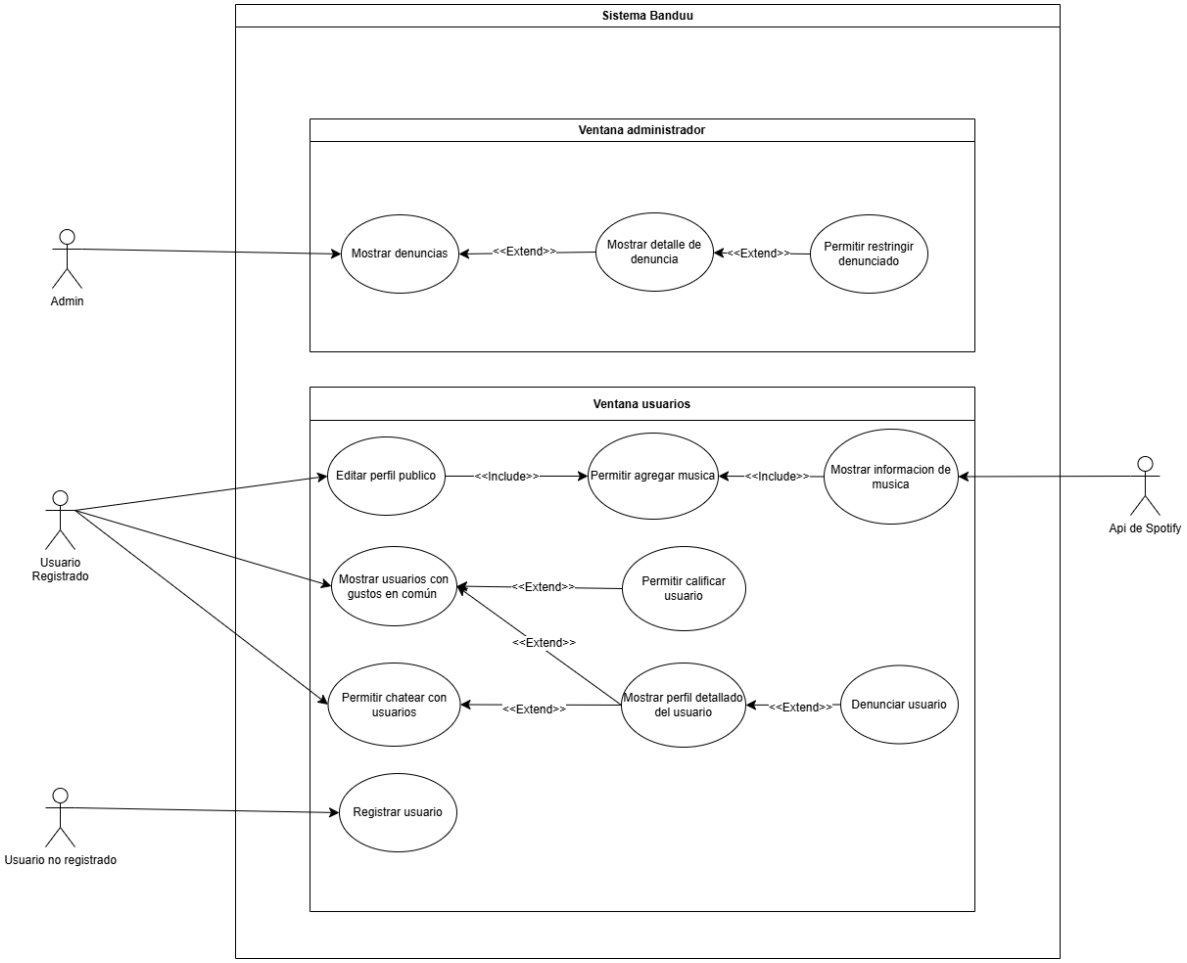
Profesor: Oscar Aguayo

Fecha: 22/04/202

Introducción

Nuestro grupo se propuso diseñar una aplicación con un estilo similar a Tinder, con un listado dinámico que compare los gustos musicales de quienes la usen para encontrar a alguien con quien tengan algún gusto en común, para de esta forma sea más fácil iniciar una conversación. El propósito es que los estudiantes tengan mayor facilidad de conocer a sus compañeros y así tener un punto de partida para poder iniciar conversaciones.

Diagrama de caso de uso



Especificación de casos de uso

Nombre	Registrar usuario
Actor	Usuario no registrado
Descripcion	Permite al usuario crear una cuenta en el sistema
Casos de uso relacionados	Este caso de uso no tiene relacion
Condiciones previas	No debe existir una cuenta creada con sus credenciales
Escenario basico	1. El usuario accede al sistema de registro 2. El usuario ingresa sus datos 3 El sistema valida la informacion 4. El sistema crea la cuenta y redirige al inicio
Alternativa	3.1 El correo ya esta registrado 3.1.1 El sistema informa sobre el error
Fallo #1	2.1 No se puede completar el registro
Fallo #2	3.1 Fallo de validacion de datos
Condiciones de fallo #1	Problemas de conexion
Condiciones de fallo #2	Datos incompletos o erroneos
Post-condiciones	Se registra al usuario

Nombre	Editar perfil publico
Actor	Usuario registrado
Descripcion	El usuario puede modificar la informacion visible de su perfil
Casos de uso relacionados	"Agregar musica favorita"
Condiciones previas	El usuario debe haber iniciado sesion
Escenario basico	1. El usuario accede a la edicion de perfil 2. El sistema carga los datos actuales 3. El usuario modifica los datos 4. El sistema guarda los cambios
Alternativa	1.1. El usuario cancela la edicion 1.2. El sistema descarta los cambios
Fallo #1	1.1 Sesion expirada
Fallo #2	4.1 No se puede guardar el perfil
Condiciones de fallo #1	Sesion caducada durante la edicion
Condiciones de fallo #2	Error de conexion o de validacion de datos
Post-condiciones	Se actualizan los datos del perfil

Nombre	Agregar musica favorita
Actor	Usuario registrado
Descripcion	Permite al usuario añadir albums,canciones o artistas favoritas a su perfil
Casos de uso relacionados	"Editar perfil publico"
Condiciones previas	El usuario debe editar el perfil
Escenario basico	1. El usuario selecciona la opcion de agregar musica 2. El sistema consulta a la API de Spotify 3. El usuario busca y selecciona canciones, albums o artistas. 4. El sistema guarda las selecciones
Alternativa	2.1 No hay conexion con Spotify 2.1.1 Se muestra un mensaje de error
Fallo #1	2.1 Error de busqueda
Fallo #2	5.1 No se pueden guardar los datos
Condiciones de fallo #1	Spotify no responde
Condiciones de fallo #2	Problemas de red
Post-condiciones	Se agregaran las elecciones al perfil

Nombre	Mostrar usuarios con gustos en común
Actor	Usuario registrado
Descripcion	El usuario puede visualizar una lista de otros usuarios que comparten gustos musicales similares, basandose en sus álbums o canciones favoritas
Casos de uso relacionados	"Calificar usuario", "Ver perfil detallado del usuario", "Chatear con usuarios"
Condiciones previas	El usuario debe iniciar sesion y creado su perfil
Escenario basico	1. El usuario accede a la vista de perfiles 2. El sistema consulta por los perfiles 3. El sistema compara los gustos entre perfiles 4. El sistema le muestra una lista de usuarios que cumplan los parametros 5. El usuario selecciona para ver el perfil o iniciar una interaccion
Alternativa	2.1 No se encuentran usuarios que coincidan con los gustos 2.1.1 El sistema envia un mensaje indicando que no existen coincidencias por el momento 2.1.2 El caso de uso finaliza
Fallo #1	2.1 Error de conexion con la base de datos de perfiles
Condiciones de fallo #1	No existen datos de los perfiles
Post-condiciones	Se muestran perfiles

Nombre	Calificar usuario
Actor	Usuario registrado
Descripcion	Permite dar "me gusta" o "rechazar" un usuario sugerido
Casos de uso relacionados	En este caso de uso se incluye "Mostrar usuarios con gustos en común"
Condiciones previas	Debe mostrarse un usuario
Escenario basico	1. El sistema muestra un perfil sugerido 2. El usuario elige la calificacion 3. El sistema guarda la calificacion
Alternativa	1.1 El usuario no califica el perfil 1.1.1 El sistema devuelve el perfil a la lista de sugeridos
Fallo #1	2.1 No se guarda la calificacion
Condiciones de fallo #1	Error de base de datos
Post-condiciones	La calificacion queda registrada

Nombre	Mostrar perfil detallado del usuario
Actor	Usuario registrado
Descripcion	Muestra más detalles de un perfil
Casos de uso relacionados	"Calificar usuario", "Chatear con usuarios", "Denunciar usuario"
Condiciones previas	Debe haberse seleccionado un perfil
Escenario basico	1. El usuario selecciona a otro usuario 2. El sistema carga los detalles del perfil
Alternativa	2.1 El usuario vuelve atras 2.1.1 El sistema vuelve a la lista
Fallo #1	1.1 No se puede mostrar el perfil
Condiciones de fallo #1	Error de consulta de perfil
Post-condiciones	Se visualiza el perfil

Nombre	Denunciar usuario
Actor	Usuario registrado
Descripcion	Permite al usuario enviar una denuncia sobre un usuario por comportamiento inapropiado
Casos de uso relacionados	"Ver perfil detallado del usuario"
Condiciones previas	El usuario debe acceder al perfil de otro usuario
Escenario basico	1. El usuario accede al perfil del usuario 2. El usuario selecciona la opcion de denuncia 3. El usuario especifica el motivo 4. El sistema guarda la denuncia
Alternativa	2.1 El usuario cancela la denuncia 2.1.1 El sistema no guarda nada
Fallo #1	4.1 Error al registrar denuncia
Condiciones de fallo #1	Problemas de conexion con la base de datos
Post-condiciones	Denuncia queda registrada

Nombre	Permitir Chatear con usuarios
Actor	Usuario registrado
Descripcion	Permite al usuario iniciar una conversacion con otro usuario compatible
Casos de uso relacionados	"Mostrar usuarios con gustos en comun", "Ver perfil detallado del usuario"
Condiciones previas	Deben coincidir los "me gusta" de ambos perfiles
Escenario basico	1. El usuario accede al listado de perfiles que coinciden 2. El usuario debe seleccionar un chat 3. El sistema permite enviar mensajes 4. El sistema envia los mensajes
Alternativa	3.1 El otro usuario no responde
Fallo #1	4.1 No se pueden intercambiar los mensajes
Condiciones de fallo #1	Error de conexion
Post-condiciones	Se muestran los mensajes en el chat

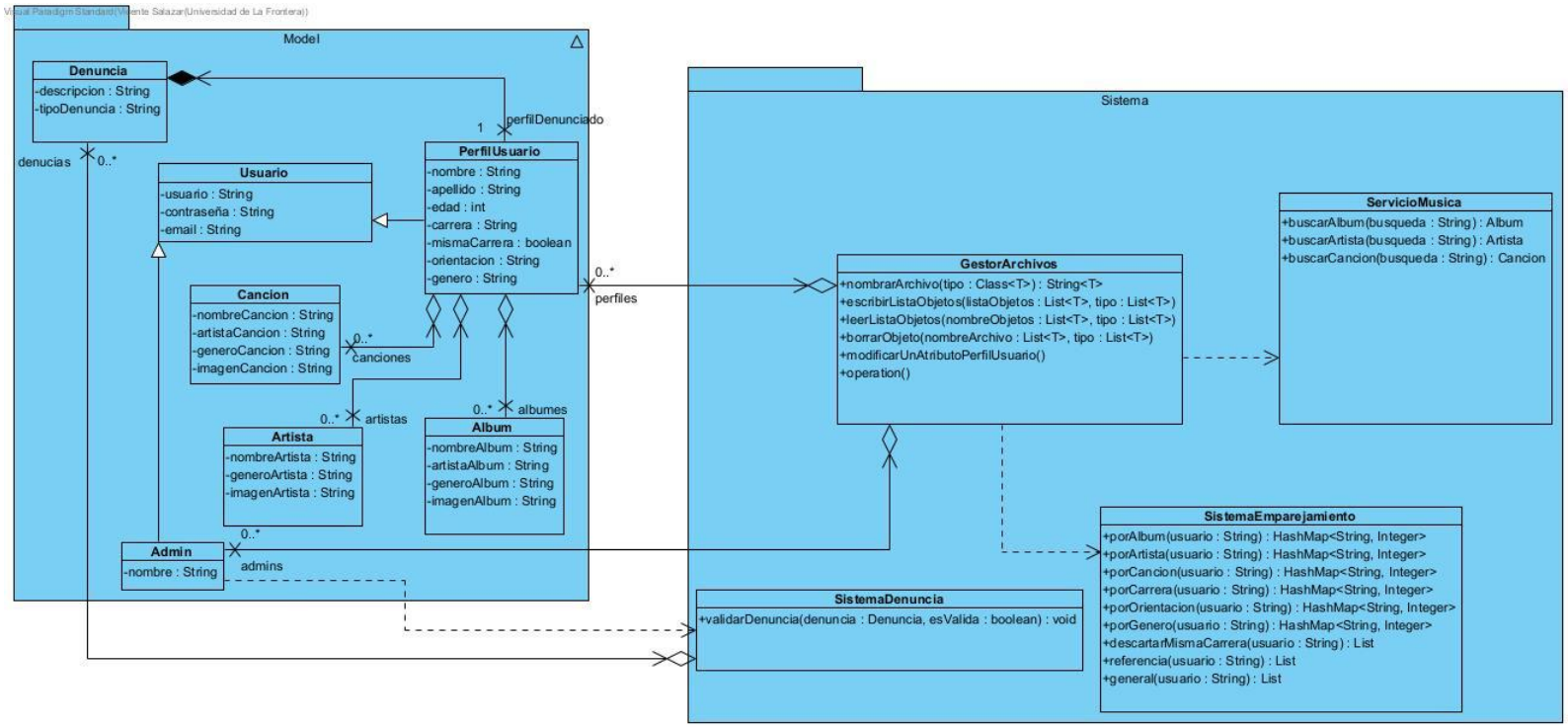
Nombre	Mostrar informacion de musica
Actor	API de Spotify
Descripcion	Permit obtener y mostrar datoss musicales desde la API externa
Casos de uso relacionados	"Permitir agregar musica"
Condiciones previas	Debe haber conexion con la API
Escenario basico	1. El sistema envia una consulta a la API 2. El sistema recibe los datos 3. El sistema muestra los datos
Alternativa	2.1 No se encuentran los resultados 2.1.1 Se muestra un mensaje de error
Fallo #1	1.1 La API no responde
Condiciones de fallo #1	Problemas de conexion con la API
Post-condiciones	Mostrar informacion de la peticion

Nombre	Mostrar Denuncias
Actor	Administrador
Descripcion	Permite al administrador visualizar las denuncias ingresadas por los usuarios
Casos de uso relacionados	"Mostrar detalle de denuncia", "Permitir restringir denunciado"
Condiciones previas	El administrador debe haber iniciado sesion
Escenario basico	1. El administrador accede al panel de denuncias 2. El sistema consulta por las denuncias 3. El sistema muestra la lista de denuncias recibidas
Alternativa	2.1 No hay denuncias 2.1.1 Se muestra mensaje "no hay denuncias"
Fallo #1	2.1 No se cargan las consultas
Condiciones de fallo #1	Problemas de conexion
Post-condiciones	Se muestra el listado de denuncias

Nombre	Mostrar detalle de denuncia
Actor	Administrador
Descripcion	Permite al administrador visualizar informacion especifica sobre una denuncia
Casos de uso relacionados	"Mostrar denuncias", "Restringir denunciado"
Condiciones previas	Debe haberse seleccionado una denuncia
Escenario basico	1. El administrador selecciona una denuncia 2. El sistema muestra todos los detalles
Alternativa	2.1 El administrador vuelve atras 2.1.1 El sistema vuelve a mostrar la lista
Fallo #1	1.1 El sistema no puede cargar las denuncias
Condiciones de fallo #1	Error de conexion
Post-condiciones	Se muestra la informacion detallada de la denuncia

Nombre	Permitir restringir denunciado
Actor	Administrador
Descripcion	Permite aplicar una sancion a un usuario tras
Casos de uso relacionados	"Mostrar denuncias", "Mostrar detalles de denuncia"
Condiciones previas	Debe haberse visualizado una denuncia validad
Escenario basico	1. El administrador analiza la denuncia 2. Selecciona una accion (bloquear) 3. El sistema ejecuta la accion
Alternativa	2.1 El administrador no aplica sancion
Fallo #1	3.1 El usuario ya habia sido sancionado
Condiciones de fallo #1	Estado inconsistente del denunciado
Post-condiciones	El denunciado queda sancionado

Diagrama de clases



Descripción de la Solución Desarrollada

Una gran parte de los métodos son similares cambiando los parámetros, por ello se utilizaran algunos como ejemplo.

Para la información de la música, artistas, álbumes, se implemento la API web de Spotify para poder recolectar información.

```java
public Album buscarAlbum(String busqueda) {
        busqueda = busqueda.replace(" ","+");
        String urlConsulta = urlBusqueda+busqueda+tipo+"album"+parametros;

        //conexion
        HttpClient client = HttpClient.newHttpClient();
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create(urlConsulta))
                .header("Authorization", ACCESS_TOKEN)
                .GET()
                .build();
        HttpResponse<String> response = null;
        try {
            response = client.send(request, HttpResponse.BodyHandlers.ofString());
        } catch (IOException | InterruptedException e) {
            throw new RuntimeException(e);
        }
        //lectura del body
        ObjectMapper mapper = new ObjectMapper();
        JsonNode raiz = null;
        try {
            raiz = mapper.readTree(response.body());
        } catch (JsonProcessingException e) {
            throw new RuntimeException(e);
        }
        JsonNode album = raiz.at("/albums/items/0");
        if (album.isMissingNode()) {
            System.out.println("No se encontró ningún álbum con ese offset.");
            return null;
        }

        //instaciar las variables del body
        String albumName = album.get("name").asText();
        String imageUrl = album.at("/images/0/url").asText();
        JsonNode artist = album.at("/artists/0");
        String artistName = artist.get("name").asText();
        String artistId = artist.get("id").asText();

        // Obtener género del artista
        HttpRequest artistRequest = HttpRequest.newBuilder()
                .uri(URI.create(urlArtista + artistId))
                .header("Authorization", ACCESS_TOKEN)
                .GET()
                .build();

        HttpResponse<String> artistResponse = null;
        try {
            artistResponse = client.send(artistRequest, HttpResponse.BodyHandlers.ofString());
        } catch (IOException | InterruptedException e) {
            throw new RuntimeException(e);
        }
        JsonNode artistRoot = null;
        try {
            artistRoot = mapper.readTree(artistResponse.body());
        } catch (JsonProcessingException e) {
            throw new RuntimeException(e);
        }

        JsonNode genres = artistRoot.get("genres");
        String genre = genres.isArray() && genres.size() > 0 ? genres.get(0).asText() : "No disponible";

        return new Album(albumName,artistName,genre,imageUrl);
    }
```

Para las funcionalidades de la aplicación utilizamos almacenamiento de listas de objetos en archivos json para gestionar los usuarios con sus respectivos parámetros.

En la misma lista se encapsulan dentro de usuario la información del perfil y dentro de la información del perfil se encuentra las preferencias musicales.

```java
public static <T> void escribirListaObjetos(List<T> listaObjetos, Class<T> tipo) {
        String nombreArchivo = nombrarArchivo(tipo);
        try{
            mapper.writeValue(new File(nombreArchivo), listaObjetos);
            logger.info("Archivo guardado de forma satisfactoria");
        }catch (Exception e){
            logger.warning("Error al escribir el archivo: "+ nombreArchivo +": "+ e.getMessage());
        }
    }
```

```java
public static <T> List<T> leerListaObjetos(String nombreArchivo, Class<T> tipo) {
        try{
            return mapper.readValue(new File(nombreArchivo),
                    mapper.getTypeFactory().constructCollectionType(List.class, tipo));
        }catch(Exception e){
            logger.warning("Error al leer el archivo");
            return Collections.emptyList();
        }
    }
```

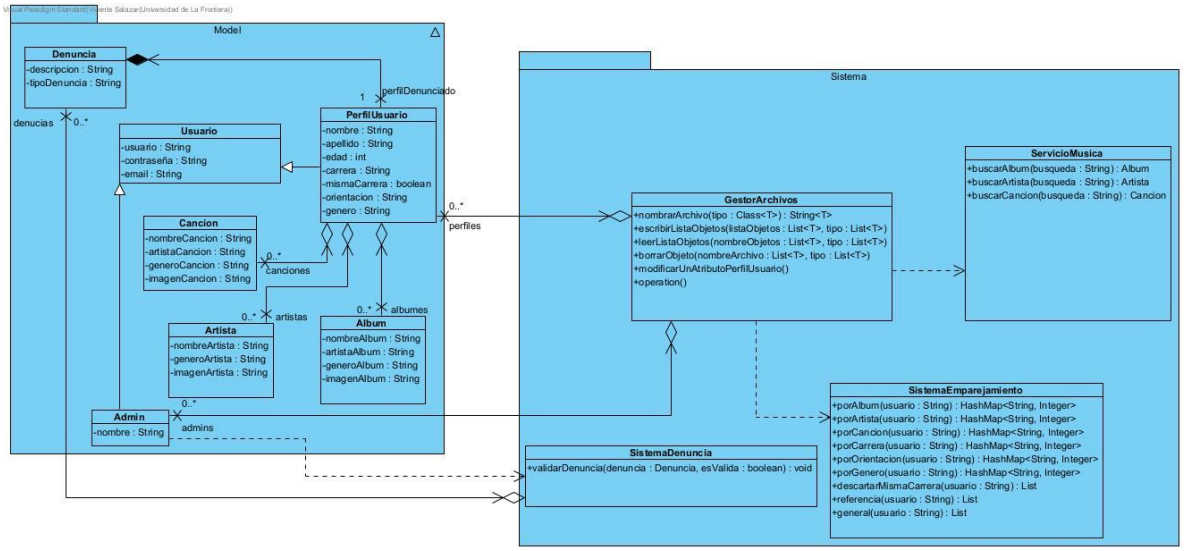
Para el sistema de emparejamiento se usará un sistema de puntos según los gustos musicales del usuario. Para eso se fue revisando los perfiles añadiéndoles puntos a los nombres de usuario, devolviendo así un hashmap.

```java
public HashMap<String, Integer> porAlbum(String usuario) {
        HashMap<String, Integer> puntuacion = new HashMap<>();
        //compara todos los usuarios del sistema con el usuario referencia
        for (Object o : general(usuario)) {
            PerfilUsuario perfilUsuario = (PerfilUsuario) o;
            int contador = 0;
            for (Object o1 : referencia(usuario)) {
                PerfilUsuario perfilUsuario1 = (PerfilUsuario) o1;
                for (Object o2 : perfilUsuario.getAlbums()) {
                    if (perfilUsuario1.getAlbums().contains(o2)) {
                        contador+=2;
                    }
                }
            }
            puntuacion.put(perfilUsuario.getUsuario(), contador);
        }

        return puntuacion;
    }
```

Para hacer una lista negra de usuarios dependiendo de la carrera se usa el método para ver los usuarios de la misma carrera, se separa los usuarios y se los añade a una lista.

```java
public List descartarMismaCarrera(String usuario) {
            return porCarrera(usuario).entrySet().stream()
                    .filter(entry -> entry.getValue() == 1)
                    .map(Map.Entry::getKey)
                    .collect(Collectors.toList());
        }
```



Conclusión

En este primer avance del proyecto se logró conectar con la API de Spotify, lo que permite extraer géneros, artistas, álbumes, canciones de tal forma que se puedan agregar a los perfiles y a partir de esto también se hizo un método emparejamiento el cual se encargara de crear las listas de estudiantes con gustos musicales en común y además de otros parámetros elegidos por ellos. Con esto ya se tiene la base del proyecto funcional.

Repositorio Git

https://github.com/TilinInsano312/PA_Proyecto_Bandu