Escuela Politécnica Nacional

Facultad de Ingenería en Sistemas

Proyecto Grupal Programación II

Manual de Estándares de Codificación y Bases de Datos

Período 2025 A

Índice

Estándares de Codificación y Bases de Datos

1. Estándares de Codificación

Se trabajará con la versión 24 de JDK, tanto el código implementado como la documentación serán escritos en inglés.

a. Nombre de archivos

Todos los nombres de archivos de soporte, diseño y código fuente van en minúsculas. Se debe mantener una relación entre el nombre del archivo y su contenido, evitando en lo posible las abreviaturas y ambigüedades.

Ejemplo: .vscode/settings.json

Los archivos referentes a clases, interfaces, controladores deben ser nombrados con UpperCamelCase, además, evitar nombres en plural.

Ejemplo: Song.java

b. Organización de archivos

Los directorios serán creados en relación al contenido. Todos los archivos correspondientes a la aplicación se encuentran en:

src\...

Dentro existe un directorio para cada uno de los tipos de archivo empleados, de modo que:

- src\controller incluye todos los archivos de control y/o clases funcionales que interactúan con los modelos e interfaz gráfica.
- src\data incluye todos los archivos de datos de la aplicación.
- src\media incluye todos los archivos de medios (mp3).
- src\model incluye todos los modelos (clases) de la aplicación.
- src\ui incluye todos los archivos para la interfaz gráfica de usuario o GUI.

La aplicación también incluye el directorio lib, utilizado para todos los archivos con extensión .jar necesarios para su correcta ejecución.

Además, todo el material necesario para el diseño y planificación se encuentra en el directorio design.

c. Identación

Se utilizará una tabulación al inicio de cada bloque dentro de una clase, método o función para identificar correctamente el código. Es necesario también colocar 1 espacio entre operadores.

Aquí hay un ejemplo

```
public String playSong(Song song) {
```

```
        if(condition1 || (condition2 && condition3)) {
                return “Mensaje1”;
    }
    else {
        return “Mensaje2”;
    }
    }
```

El operador ternario será usado de la siguiente manera

```
beta = (booleanExpression) ? action1 : action2;
```

Además, entre métodos o bloques distintos se debe añadir un salto de línea.

```
method1 {
        //codeBlock
}

method2{
        //codeBlock
}
```

d. Comentarios

Se debe evitar en lo posible el uso de comentarios en la misma línea o llenar el código de comentarios, los únicos comentarios admitidos serán para documentación.

i. Documentación

Para identificar la documentación se utilizará /** … */, que debe incluir todos los parámetros descritos en el ejemplo:

Ejemplo para clases

```
/**
 *Descripción breve de la clase
 *Métodos con los que trabaja
 */
```

Ejemplo para métodos

```
/**
 *@param valores o parámetros que recibe la función
 *@throws control de excepciones
 *@return valores que devuelve o retorna
```

*/

e. Declaraciones

Se debe realizar una declaración por línea, agrupando variables del mismo tipo antes de cambiar de variables. La declaración de variables irá al inicio de cada bloque de código y deberán ser inicializadas, a menos que requieran ser procesadas antes de asignarles un valor.

Ejemplo:

```
int level = 0;
int size = 0;
int height = 0;
boolean empty = true;
boolean full = false;
```

No se deben incluir espacios entre la declaración de un método y el paréntesis “(” que inicia la descripción de sus parámetros.

La llave de inicio de un método o clase “{” estará separada por 1 espacio y la llave de cierre “}” debe ir en una línea por sí sola.

f. Convención de nombres

Identificador	**Convención**	**Ejemplo**
Clases	El nombre de las clases debe ser un sustantivo singular que describa de la mejor manera su función. Se debe evitar el uso de abreviaturas y ambigüedades.	class Artist;
Interfaces	Seguirán la misma convención para clases.	interface MediaPlayer;

Variables	Las variables serán nombradas por lowerCamelCase. Su nombre debe ser corto y descriptivo. Solo serán admitidas las variables de una sola letra para variables temporales.	int i; //variable temporal para un contador
		float myStreamingTime;

Constantes	Su nombre debe ser escrito con mayúsculas y usando “_” para espacios. Por lo general su uso no es recomendado.	int MAX_LENGHT = 3;

g. Buenas Prácticas

No se debe colocar todos los modificadores de acceso como públicos a menos que sea estrictamente necesario para el programa. Caso contrario todo debe manejarse como privado o protegido. De igual modo se debe acceder a una clase a través de getters / setters, más no de forma directa.

2. Estándares de Bases de Datos

Se usará una base de datos relacional con SQLite. Al igual que el código será escrita en inglés.

a. Estructura General

La base de datos está organizada en base a 6 tablas principales:

- song: canciones
- album: Álbumes
- artist: Artistas
- playlist: Listas de reproducción
- album_artist: la relación manejada será 1 a muchos, por el momento solo se aceptará el artista principal del álbum.
- playlist_song: la relación manejada será 1 a 0 o muchos, una playlist puede contener muchas canciones o crearse vacía.

b. Convenciones

Tipo	**Convención**	**Ejemplo**
Tablas	Se usarán nombres en minúsculas y en singular. Para las tablas de relación se usará snake_case, con los nombres de las tablas relacionadas separados por '_'	song playlist_song
Columnas	Se usarán nombres en minúscula que describan adecuadamente el contenido, se usará snake_case de ser necesario.	publish_year name
PRIMARY KEY	Se usará la columna 'id' como PRIMARY KEY. Se debe establecer como INTEGER y AUTOINCREMENT. Será la primera columna de cada tabla.	N/A
FOREIGN KEY	Las claves foráneas se identificarán	artist_id

	usando snake_case, con la tabla de origen e ‘id’ separados por ‘_’	album_id

c. Tipos de datos

Tipo de dato	**Uso**
INTEGER	Empleado para claves primarias (id), años y relaciones
TEXT	Empleado para nombres, títulos y rutas de acceso a archivos
REAL	Empleado en la tabla song, para identificar la duración en minutos.segundos

d. Claves Primarias y Foráneas

Tabla	**PRIMARY KEY**	**FOREIGN KEY**
song	id	album_id → album(id)
album	id	N/A
artist	id	N/A
playlist	id	N/A
album_artist	Compuesta: album_id, artist_id (tabla de relación)	album_id → album(id) artist_id → artist(id)
playlist_song	Compuesta: playlist_id, song_id (tabla de relación)	playlist_id→ playlist(id) song_id → song(id)