Desarrollo de software para identificar canciones más escuchadas en una emisora y realizar sorteos

En el contexto de una emisora de radio, se presenta la necesidad de crear un software capaz de identificar las canciones más populares entre los oyentes y gestionar la realización de sorteos basados en las respuestas de una encuesta. El propósito principal es mejorar la interacción con la audiencia y promover la participación de los oyentes, al tiempo que se obtienen datos valiosos sobre las preferencias musicales.

Detalles del problema

Canciones más escuchadas: La emisora busca identificar cuáles son las canciones más escuchadas por su audiencia. Para ello, planea llevar a cabo una encuesta entre 50 personas seleccionadas aleatoriamente. En esta encuesta, cada persona debe elegir un artista y, a partir de este, seleccionar tres canciones favoritas.

Recompensa por Participación: Como incentivo para que los oyentes participen en la encuesta, la emisora entrega una boleta a cada encuestado, que contiene un número y un color únicos. Esta boleta sirve como entrada para un sorteo posterior.

Sorteo de Premios: Una vez finalizada la encuesta, la emisora realiza un sorteo de premios. Si el número y el color en la boleta de un oyente coinciden con los elegidos en el sorteo, el oyente gana el premio. ~~En caso contrario, se continúa con la rifa hasta que se determine un ganador~~.

Normalización de Registros: Es importante que el software cuente con un sistema de normalización para los registros de artistas y canciones. Esto garantizará que los nombres de artistas y las canciones estén registrados de manera coherente y uniforme en la base de datos.

Objetivos del software

- Recopilar y almacenar las respuestas de la encuesta, que incluyen el nombre del artista y las tres canciones favoritas.
- Normalizar los registros de artistas y canciones para evitar duplicaciones y errores en los datos.
- Generar boletas únicas con números y colores para cada participante de la encuesta.
- Realizar sorteos aleatorios y determinar si un oyente ha ganado un premio.
- Proporcionar informes sobre las canciones más populares y los resultados de los sorteos.

Requerimientos funcionales genéricos

RF1: Registro de canciones y artistas

- El sistema debe normalizar los nombres de artistas y canciones para evitar duplicaciones y errores en los datos.

Catálogo de Artistas

Un artista musical es el encargado de crear las obras radiofónicas emitidas por la emisora de radio.

Para un artista musical nos interesa conocer su `nombre`, independientemente de que sea individual o grupal, y su `biografía`. A su vez un artista tiene asociado un `arreglo de canciones` de las que es propietario, autor, compositor…

Endpoints: `/api/artists` `(ALL VERBS)`

Tabla BBDD: `ARTIST`

Restricciones de negocio:

- El nombre del artista tendrá como máximo 256 caracteres y no estará vacío. De cumplimiento obligatorio.
- La biografía del artista es un campo de como máximo 2048 caracteres. No es obligatorio.
- No se permitirá eliminar artistas si ya tienen canciones en su repertorio asociadas a alguna encuesta.
- Los campos opcionales, pueden dejarse en blanco durante la creación o edición. El resto de los campos no pueden quedarse en blanco.

Auditoría de datos: SI *(timestamp y usuario tanto de creación como de última modificación)*.

Identificador: Simple, artificial *(dato de tipo `Long`)*.

Relaciones con otras entidades:

- Cada artista puede tener un número infinito de canciones (`0:N`).

Catálogo de canciones

Una canción es aquella obra radiofónica susceptible de ser emitida por la emisora de radio y la base de las encuestas que se desea realizar con tal de conocer los gustos de la audiencia.

Para gestionar una canción se deberá primero conocer su autor, por lo que es preferible que se monte el endpoint de gestión de canciones bajo la ruta de cada artista. De ese modo daremos también a la API un poco de semántica.

Para una canción también nos interesa conocer:

- `title`: es el título de la canción. Longitud de 256 caracteres, no vacío, dato obligatorio.
- `releaseYear`: es un entero positivo que identifica el año de lanzamiento de la canción (dato opcional).
- `duration`: es un entero positivo que indica la longitud de la canción en segundos (dato opcional).
- `genre`: es el género musical de la canción. 50 caracteres (dato opcional).

Endpoints: `/api/artists/[artistId]/songs` `(ALL VERBS)`

Tabla BBDD: `SONG`

Restricciones de negocio:

- Los campos opcionales, pueden dejarse en blanco durante la creación o edición. El resto de los campos no pueden quedarse en blanco.
- No se podrá eliminar canciones que ya estén asociadas a alguna respuesta de encuesta.
- No se permitirá cambiar el autor de una canción.

Auditoría de datos: SI *(timestamp y usuario tanto de creación como de última modificación)*.

Identificador: Simple, artificial *(dato de tipo `Long`)*.

Relaciones con otras entidades:

- Cada canción solo puede pertenecer a un artista (`1:N`).
- Cada oyente participa, a modo de respuestas de encuesta, con un número fijo de canciones y, una misma canción puede ser elegida por múltiples participantes (`N:M` mediante la entidad pivote `RespuestaEncuesta`) e indistintamente de la encuesta.

RF2: Registro de Encuestas

- El sistema debe permitir a los encuestados registrar sus respuestas.
- Cada respuesta debe contener el nombre del artista y tres canciones favoritas.
- Cada respuesta debe asociar los datos básicos de quien la realizó.

Catálogo de oyentes

Un oyente es aquella persona que escucha las canciones emitidas por una determinada emisora de radio. Dicha persona es susceptible de participar en las distintas encuestas que haga dicha emisora de radio.

Como datos básicos deseamos capturar:

- `name`: el nombre completo de la persona. 255 caracteres, no vacío, obligatorio.
- `phone`: el teléfono de contacto de la persona. 10-15 caracteres, pattern="`\+\d+`", obligatorio.
- `email`: el email de contacto de la persona. 255 caracteres, pattern, obligatorio, único para todos los participantes.
- `address`: dirección postal del participante. 500 caracteres, (dato opcional).

Estos datos, debido a la susceptibilidad de estar bajo la Ley Orgánica de Protección de Datos (LOPD) serán tratados como tal bajo las directrices y parámetros que suscita dicha Ley, siendo anonimizados durante el proceso de sorteo mediante el uso de boletos. Solo el propio interesado así como la entidad pertinente (poder judicial o empresa adjudicataria del software) serán los encargados de mantener y asegurar la veracidad y seguridad de dicha información.

Endpoints: `/api/v1/radio-listeners` `(ALL VERBS)`

Tabla BBDD: `RADIO_LISTENER`

Restricciones de negocio:

- Los campos opcionales, pueden dejarse en blanco durante la creación o edición. El resto de los campos no pueden quedarse en blanco.
- No se podrá eliminar oyentes que tengan asociadas alguna participación de encuesta.

Auditoría de datos: SI *(timestamp y usuario tanto de creación como de última modificación)*.

Identificador: Simple, artificial *(dato de tipo `Long`)*.

Relaciones con otras entidades:

- Cada oyente participa, respondiendo a encuestas con un número fijo de canciones a modo de respuestas. Un mismo participante puede responder múltiples encuestas y a su vez una encuesta tiene múltiples participantes, (`N:M` mediante la entidad pivote `ParticipacionEncuesta`).

Catálogo de encuestas

Una encuesta es una serie de preguntas que se hace a muchas personas con la intención de reunir datos para detectar la opinión pública de un asunto. En este caso de conocer las mejores canciones.

Como datos básicos se desea conocer:

- `title`: el título de la canción. 255 caracteres, no vacío, obligatorio.
- `description`: la descripción de la canción. 2048 caracteres (dato opcional).
- `status`: campo interno que determina el estado de la encuesta. 50 caracteres, enumerado, obligatorio.
- `start_date`: fecha de comienzo de la encuesta, tipo `LocalDateTime`, obligatorio.
- `end_date`: fecha de finalización de la encuesta, tipo `LocalDateTime`, obligatorio. posterior a `start_date`.
- Configuración de encuesta:
  - `num_max_participants`:número máximo de participantes. Por defecto: 50.
  - `num_survey_responses`: número de respuestas necesarias. Por defecto: 3.

Endpoints: `/api/surveys` `(ALL VERBS)`

`/api/surveys/[surveyId]/close` `(POST)`

Tabla BBDD: `SURVEY`

Restricciones de negocio:

- Los campos opcionales, pueden dejarse en blanco durante la creación o edición. El resto de los campos no pueden quedarse en blanco.
- Si al crear la encuesta, el catálogo de oyentes o de canciones está vacío, se lanzará un error como forma de notificar al usuario.

- No se podrá eliminar encuestas que tengan asociadas algún oyente, sorteo o respuesta.
- Habrá un número máximo de participantes, 50, que serán seleccionados aleatoriamente de entre todos los oyentes de la emisora.
- Cada participante escogerá un artista y 3 canciones del mismo.
- Tanto el número máximo de participantes como el número de canciones no serán obligatorios a la hora de crear el recurso, cogiendo del `application.properties` sus valores de configuración por defecto correspondientes.
- Las encuestas se crean con `status=PENDING`.
- Una vez se seleccionen los participantes, no se podrá editar la configuración de encuesta. Una vez en `status=CLOSED`, no se podrá editar de ninguna manera.
- La entidad pasará a `status=RUNNING` una vez algún oyente envíe sus respuestas (canciones favoritas).
- La entidad pasará a `status=CLOSED` si se sobrepasa la fecha de fin de la encuesta y hay algún boleto generado en ella (proceso manual mediante endpoint), o cuando todos los participantes hayan respondido (proceso automático). En este estado y sólo en éste, se le podrá asociar un sorteo/rifa de premios.

Auditoría de datos: SI *(timestamp y usuario tanto de creación como de última modificación)*.

Identificador: Simple, artificial *(dato de tipo `Long`)*.

Relaciones con otras entidades:

- Cada encuesta puede generar a su finalización una rifa/sorteo (`0:1`).
- Un mismo participante/oyente puede responder múltiples encuestas y a su vez una encuesta tiene múltiples participantes, (`N:M` mediante la entidad `ParticipacionEncuesta`).

Participaciones por encuesta

Se trata de una entidad pivote que relaciona a los oyentes con las múltiples encuestas en las que han sido seleccionados aleatoriamente.

Por lo tanto, para gestionar una participación se deberá primero conocer la encuesta y al oyente participante en la misma; por lo que es preferible que se monte el endpoint de selección de participantes bajo la ruta de cada encuesta. También se podrá consultar las encuestas de un determinado oyente desde su correspondiente lado de la relación. De ese modo daremos también a la API un poco de semántica.

Endpoints: `/api/surveys/[surveyId]/participations` (ALL VERBS)

`/api/radio-listener/[participantId]/surveys` (GET)

Tabla BBDD: `SURVEY_PARTICIPATION`

Restricciones de negocio:

- Se seleccionarán todos los participantes de una tacada. Una vez seleccionados los participantes no se podrán agregar más.
- El número de participantes máximo a seleccionar se determinará a partir de la configuración de la encuesta.
- Si al seleccionar los participantes, el catálogo de oyentes o de canciones está vacío, se lanzará un error como forma de notificar al usuario.
- No entra dentro del alcance del problema modificar ni borrar las participaciones desde fuera del sistema.
- La fecha de participación (`participated_at`), de tipo `LocalDateTime`, se rellena cuando el participante envía sus canciones favoritas.
- El boleto de rifa también es generado cuando el participante envíe sus canciones.

Auditoría de datos: SI *(timestamp y usuario tanto de creación como de última modificación)*.

Identificador: Compuesto, artificial *(ids tipo `Long` provenientes de `Encuesta` y `Oyente`)*.

Relaciones con otras entidades:

- Cada participación generará un boleto de rifa/sorteo (`0:1`) cuando el oyente envíe sus n-canciones favoritas.
- Una participación se relaciona tanto con encuesta como oyente (extremos `N:M`) con una cardinalidad (`1:N`)

Respuestas de participación por encuesta

Se trata de una entidad pivote que relaciona la participación de un oyente en una determinada encuesta con las canciones que ha determinado ser sus favoritas dentro de toda la audiencia.

Para gestionar el arreglo de respuestas (canciones) se deberá primero conocer la encuesta y al oyente participante en la misma. Semánticamente, tiene más sentido enviar dicho arreglo desde el endpoint correspondiente del oyente, ya que es quien debe cubrir la encuesta. No obstante, se abre la posibilidad de listar todas las respuestas de cualquier participante desde el endpoint de encuestas.

Endpoints:

`/api/surveys/[surveyId]/participations/responses` (ALL VERBS)

`/api/radio-listener/[participantId]/surveys/[surveyId]/responses` (GET)

Tabla BBDD: `SURVEY_RESPONSING`

Restricciones de negocio:

- A la hora de crear el recurso, se seleccionarán todas las canciones de una tacada. Si el catálogo de canciones está vacío, se lanzará un error como forma de notificar al usuario.
- No se podrán modificar ni eliminar las canciones registradas en forma de respuestas.
- El número de respuestas a enviar viene determinado por la configuración de la encuesta.
- Preferiblemente las canciones serán todas del mismo autor, pudiendo completar hasta el número de respuestas configurada en la encuesta con las de otros autores musicales.
- Para no repetir el mismo dato en cada registro de respuesta, la fecha de envío de respuestas se traslada a la entidad `Participación`. Se establece la fecha de participación (`participated_at`) a la actual, cuando el participante envía sus canciones favoritas.
- El boleto de rifa también es generado cuando el participante envíe sus canciones y establecido en la entidad `Participación`.

Auditoría de datos: NO.

Identificador: Compuesto, artificial *(ids tipo `Long` provenientes de `Encuesta`, `Oyente` y `Canción`)*.

Relaciones con otras entidades:

- Cada respuesta se relaciona con participación y canciones en la forma cardinal (`1:N`).

RF3: Generación de Boletas

- El sistema debe generar boletas únicas para cada encuestado.
- Cada boleta debe tener un número y un color únicos.

Catálogo de colores

La entidad color es una parte importante en la generación de la parte boletos, ya que se trata de uno de los componentes que identifica unívocamente a las mismas junto con el número de ticket.

El único dato básico a capturar será el código de color (`code`), ya sea en formato hexadecimal (`#F2F2F2`) o el alias (`rebeccapurple`). Será una cadena de caracteres de 25 caracteres, no vacía y obligatoria.

Endpoints: `/api/colors` `(ALL VERBS)`

Tabla BBDD: `COLOR`

Restricciones de negocio:

- Los códigos de colores serán únicos en todo el sistema.
- No se pueden editar los códigos de colores.
- No se podrá eliminar un color que ya esté asociado a un ticket de sorteo.

Auditoría de datos: SI *(timestamp y usuario tanto de creación como de última modificación)*.

Identificador: Simple, artificial *(dato de tipo `Long`)*.

Relaciones con otras entidades:

- Un color puede estar presente en múltiples boletos (`1:N`).

Boleto de Sorteos

Es el cartón o papel que se reparte entre los oyentes que realizan la encuesta para el futuro sorteo.

Como datos básicos deseamos conocer:

- `number`: tendrá como máximo 50 caracteres, no vacío, obligatorio.
- `color`: de tipo Color, de tipo Long referenciando a uno del catálogo de colores.

Endpoints: NO

Tabla BBDD: `RAFFLE_TICKET`

Restricciones de negocio:

- Una vez realizada la encuesta (enviadas las respuestas) se entrega una boleta a cada participante. Será el sistema, que internamente, el que la genere. No hay posibilidad alguna de que esta tarea la pueda realizar el usuario.
- Las boletas serán únicas para todos los participantes. Cada boleta tendrá un número y un color único en todo el sistema.
- No hay posibilidad de eliminar ni modificar los boletos.
- Si el catálogo de colores está vacío, se lanzará un error como forma de notificar al usuario.

Auditoría de datos: NO

Identificador: Simple, artificial *(dato de tipo `Long`)*.

Relaciones con otras entidades:

- Un tiquet de sorteo posee un `Color` y un color es empleado en múltiples tickets, por lo que su relación tiene cardinalidad (`1:N`).
- Un ticket pertenece a una y solo una participación de encuesta por lo que la relación de cardinalidad con `Survey_Participation` es de (`0:1`).

RF4: Realización de Sorteos

- El sistema debe realizar sorteos aleatorios basados en las boletas generadas.
- Debe determinar si un oyente ha ganado un premio en el sorteo.

Catálogo de premios

Los premios son el reconocimiento material dado a la persona que participa en las distintas encuestas lanzadas por la emisora de radio.

Para la entidad `Premio` nos interesa conocer:

- `title`: El título del premio del sorteo, no puede tener más de 255 caracteres, obligatorio.
- `description`: Es la descripción del premio del sorteo, opcional. de longitud máxima 2048 caracteres.
- `monetary_value`: Es el valor monetario del premio del sorteo. Un número decimal positivo.

Endpoints: `/api/prizes` `(ALL VERBS)`

Tabla BBDD: `PRIZE`

Restricciones de negocio:

- Los campos opcionales, pueden dejarse en blanco durante la creación o edición. El resto de los campos no pueden quedarse en blanco.
- No se podrá eliminar premios que estén asociados a algún sorteo.
- No se podrá editar premios que estén asociados a algún boleto ganador.

Auditoría de datos: SI *(timestamp y usuario tanto de creación como de última modificación)*.

Identificador: Simple, artificial *(dato de tipo `Long`)*.

Relaciones con otras entidades:

- Una misma definición de premio puede ser sorteado en múltiples sorteos y a su vez una rifa sortea múltiples premios, (cardinalidad `N:M` descrita mediante la entidad `PremioSorteo`).

Sorteos de encuesta

El sorteo de premios dentro de una encuesta es la forma que tendrá la emisora de radio de promover la participación de los oyentes en las encuestas.

Como datos básicos se desea conocer:

- `status`: El estado interno del sorteo. 50 caracteres, no vacío, obligatorio, enumerado.
- `resolution_date`: La fecha de resolución es de tipo `LocalDateTime`, (dato opcional).

Endpoints: `/api/raffles` (ALL VERBS)

`/api/raffles/[surveyId]/prizes` (ALL VERBS)

`/api/raffles/[surveyId]/run` (POST)

Tabla BBDD: `RAFFLE`

Restricciones de negocio:

- Solo se podrán crear sorteos de encuestas en estado `CLOSED`. Siempre se crean con el estado interno `status=PENDING`, estado con el que únicamente se puede configurar sus premios asociados.
- Al correr el sorteo se cambia el estado a `RUNNING`. Solo se podrán correr sorteos que tengan como mínimo un premio.
- Al finalizar el sorteo del premio se establece la fecha de resolución del mismo, el boleto ganador y se cambia el estado del sorteo a `RESOLVED`.
- El rango de premios a asociar a un sorteo está entre 1 y el número de boletos que se hayan entregado para la encuesta subyacente.
- Un boleto solo puede ganar un premio dentro de una misma rifa.
- Si el catálogo de premios está vacío, se lanzará un error como forma de notificar al usuario.

Auditoría de datos: SI *(timestamp y usuario tanto de creación como de última modificación)*.

Identificador: Simple, artificial *(dato de tipo `Long`)*.

Relaciones con otras entidades:

- La entidad `Sorteo` tiene una relación de cardinalidad (`1:1`) con `Encuesta`.
- Una rifa sortea múltiples premios y a su vez una misma definición de premio puede ser sorteado en múltiples sorteos, (cardinalidad `N:M` descrita mediante la entidad `PremioSorteo`).

Premios asociados a un sorteo

`Premio-Sorteo` es una entidad pivote que permite relacionar la definición de un premio con un determinado sorteo a la vez que, al realizar el sorteo de premios permite conocer el boleto ganador del mismo.

Endpoints: `/api/raffles/[raffleId]/prizes` (ALL VERBS)

`/api/raffles/[raffleId]/run` (POST)

Tabla BBDD: `RAFFLE_PRIZE`

Restricciones de negocio:

- El rango de premios a asociar a un sorteo está entre 1 y el número de boletos que se hayan entregado para la encuesta subyacente.
- Solo se podrán gestionar los premios de un sorteo si el sorteo está en `status=PENDING`.
- No se permite asociar un mismo premio varias veces a un sorteo en particular.
- El campo `winner_ticket_id` se establece posteriormente, cuando se realice el sorteo aleatorio, y el valor será uno de los boletos de la encuesta asociada al sorteo.
- Si el catálogo de premios está vacío, se lanzará un error como forma de notificar al usuario.

Auditoría de datos: SI *(timestamp y usuario tanto de creación como de última modificación)*.

Identificador: Compuesto, artificial *(ids tipo `Long` provenientes de `Premio` y `Sorteo`)*.

Relaciones con otras entidades:

- Cada premio asociado a un sorteo será rifado entre todos los boletos que entren en el sorteo, pero uno y sólo uno puede ser el ganador del premio (cardinalidad `0:1`).
- Una asociación de premio se relaciona tanto con premio como con sorteo (extremos `N:M`) con una cardinalidad (`1:N`).

RF5: Informes de Popularidad de Canciones

- El sistema debe generar informes sobre las canciones más populares basados en las respuestas de la encuesta.
- El sistema debe recopilar la información de forma segura y fiable.

Endpoints: `/api/reports/songs-popularity` `(GET)`

Para cada canción sacaremos en el informe (JSON) los campos:

- `song.id`: como identificador de la canción.
- `song.title`: como nombre/título de la canción.
- `song.releaseYear`: como año de publicación de la canción.
- `song.genre`: como género musical de la canción.
- `artist.id`: como identificador del artista musical al que pertenece la canción.
- `artist.name`: como nombre del artista musical al que pertenece la canción.
- `likes`: como contador de grupo que indica la popularidad agregada de dicha canción.

Se podrá filtrar opcionalmente por:

- Un rango sobre la fecha de encuesta o la de respuesta.
- El género musical de la canción.

RF6: Informes de Resultados de Sorteos

- El sistema debe generar informes sobre los resultados de los sorteos, incluyendo los ganadores.
- El informe debe incluir validez sobre las participaciones y con los requisitos legales de que no haya duplicados.

Endpoints: `/api/reports/raffle-results` `(GET)`

Para cada sorteo sacaremos en el informe (JSON) los campos:

- `raffle.id`: como identificador del sorteo.
- `raffle.resolutionDate`: como fecha de resolución del sorteo.
- `raffle.status`: como estado del sorteo.
- `survey.title`: como nombre de la encuesta asociada al sorteo.
- `raffle.prizes[]`: arreglo con las relaciones premios-ganador del sorteo.
  - `prize.id`: como identificador del premio.
  - `prize.title`: como nombre del premio.
  - `prize.monetaryValue`: como valor monetario del premio.
  - `participant.raffleTicket.number`: como número del boleto.
  - `participant.raffleTicket.color`: como color del boleto.
  - `participant.participatedAt`: fecha de envío de respuestas de encuesta.
  - `participant.id`: identificador del participante/oyente ganador.
  - `participant.name`: nombre completo del participante/oyente ganador.
  - `participant.phone`: telefono de contacto del participante/oyente ganador.

Se podrá filtrar opcionalmente por:

- Un rango sobre la fecha de sorteo.
- Número o color de boleto, como forma de saber si un boleto es ganador o no.

Requerimientos no funcionales

RNF1: Eficiencia en la Base de Datos

- La estructura de la base de datos debe ser eficiente para gestionar grandes cantidades de datos.

Con tal de implementar dicho requerimiento y a su vez con el normalizado de datos, cada entidad contará con una tabla propia en la propia base de datos.

Por ejemplo: En la relación Sorteo-Premio, se podrían repetir la misma especificación de Premio para varios Sorteos. Así es necesario introducir una tabla pivote intermedia, Sorteo-Premio, que contendrá los registros de las relaciones, únicas o repetidas. Lo mismo sucede en Encuesta-Oyente y Participación-Canción.