## 1. Esta consulta se usa para obtener los chats directos del usuario.

SELECT chats.\*, users.photo as photo, users.UserName as UserName, users.subscribed as verify

FROM chats

JOIN users ON (

(chats.to = users.id AND chats.from = :userId)

OR (chats.from = users.id AND chats.to = :userId)

)

WHERE chats.status = 1

AND chats.AcceptFrom IS NULL

AND chats.AcceptTo IS NULL

En esta consulta SQL:

* :userId es un parámetro que representa el ID del usuario autenticado.
* La cláusula JOIN se realiza en la tabla users con una condición que verifica si chats.to es igual al ID del usuario autenticado y chats.from es igual al ID del usuario autenticado, o viceversa.
* Se seleccionan las columnas de la tabla chats y algunas columnas de la tabla users.
* Se filtran los resultados para incluir solo aquellos donde chats.status es 1 y tanto chats.AcceptFrom como chats.AcceptTo son NULL.

## 2. Consulta para obtener los chats de transacción del usuario.

SELECT chats.\*, users.photo as photo, users.UserName as UserName, users.subscribed as verify

FROM chats

JOIN users ON (

(chats.to = users.id AND chats.from = :userId)

OR (chats.from = users.id AND chats.to = :userId)

)

WHERE chats.status = 1

AND (chats.AcceptFrom IS NOT NULL OR chats.AcceptTo IS NOT NULL)

En esta consulta SQL:

* :userId es un parámetro que representa el ID del usuario autenticado.
* La cláusula JOIN se realiza en la tabla users con una condición que verifica si chats.to es igual al ID del usuario autenticado y chats.from es igual al ID del usuario autenticado, o viceversa.
* Se seleccionan las columnas de la tabla chats y algunas columnas de la tabla users.
* Se filtran los resultados para incluir solo aquellos donde chats.status es 1 y donde chats.AcceptFrom no es NULL o chats.AcceptTo no es NULL.

## 3. Consulta para obtener las licencias:

```sql

SELECT \* FROM licenses

WHERE status = 1

AND post IS NULL;

```

## 4. Consulta para obtener los métodos de pago:

```sql

SELECT \* FROM paid\_methods

WHERE status = 1;

```

## 5. Consulta para obtener los usuarios:

```sql

SELECT \* FROM users

WHERE status != 0

AND type = 1

AND id != :auth\_user\_id;

```

`:`auth\_user\_id` es un parámetro que representa el ID del usuario autenticado.

## 6. Consulta para obtener las guías:

```sql

SELECT \* FROM guides

WHERE status = 1;

```

## 7. Consulta para obtener las preguntas y respuestas:

```sql

SELECT \* FROM qas

WHERE status = 1;

```

## 8. Consulta para obtener los post publicados por Soulbeat.

SELECT posts.\*

FROM posts

JOIN users ON posts.user = users.id

WHERE users.type = 1

AND posts.status = 1

ORDER BY posts.created\_at DESC;

Esta consulta SQL:

* Realiza un JOIN entre las tablas posts y users donde posts.user es igual a users.id.
* Filtra los resultados para incluir solo aquellos donde users.type es 1 y posts.status es 1.
* Selecciona todas las columnas de la tabla posts.
* Ordena los resultados por posts.created\_at en orden descendente.

## 9. Consulta para obtener los usuarios de Soulbeat con mas seguidores.

SELECT users.\*,

(SELECT COUNT(\*)

FROM users AS followers

WHERE followers.followed LIKE CONCAT('%~', users.id, '~%')) as followers\_count

FROM users

WHERE status = 1

AND id != :auth\_user\_id

ORDER BY followers\_count DESC

LIMIT 5;

En esta consulta SQL:

* Se seleccionan todas las columnas de la tabla users.
* Se incluye una subconsulta que cuenta el número de seguidores (followers\_count) para cada usuario.
* La subconsulta cuenta las filas en la tabla users (alias followers) donde la columna followed contiene el ID del usuario actual (users.id).
* Se filtran los resultados para incluir solo aquellos donde status es 1 y el id no es igual al ID del usuario autenticado (:auth\_user\_id).
* Se ordenan los resultados por followers\_count en orden descendente.
* Se limitan los resultados a los primeros 5 registros.

## 10. Consulta para obtener las playlist de un usuario.

SELECT \*

FROM playlists

WHERE user = :user\_id

AND status = 1;

En esta consulta SQL:

* :user\_id es un parámetro que representa el ID del usuario.
* Se seleccionan todas las columnas de la tabla playlists.
* Se filtran los resultados para incluir solo aquellos donde la columna user es igual al ID del usuario y la columna status es igual a 1.

## 11. Consulta para obtener los posts mas reaccionados.

SELECT posts.\*,

users.photo as UserPhoto,

users.UserName,

users.name as PersonName,

users.FullName as PersonFullName,

users.subscribed,

IF(posts.user = :auth\_user\_id, 'True', 'False') as ThisUser

FROM posts

JOIN users ON posts.user = users.id

WHERE posts.status = 1

AND NOT EXISTS (

SELECT 1

FROM ads

WHERE ads.post\_id = posts.id

)

ORDER BY (reaction1 + reaction2 + reaction3 + reaction4 + reaction5) DESC

LIMIT :limit;

En esta consulta SQL:

* :auth\_user\_id es un parámetro que representa el ID del usuario autenticado.
* :limit es un parámetro que representa el número máximo de registros a devolver.
* Se seleccionan todas las columnas de la tabla posts y algunas columnas de la tabla users.
* Se realiza un JOIN entre las tablas posts y users donde posts.user es igual a users.id.
* Se filtran los resultados para incluir solo aquellos donde posts.status es 1 y no existen registros en la tabla ads que tengan ads.post\_id igual a posts.id.
* Se ordenan los resultados por la suma de reaction1, reaction2, reaction3, reaction4 y reaction5 en orden descendente.
* Se limitan los resultados al número especificado por el parámetro :limit.

## 12. Consulta para obtener los posts anunciados.

SELECT posts.\*,

users.photo as UserPhoto,

users.UserName,

users.name as PersonName,

users.FullName as PersonFullName,

users.subscribed,

IF(posts.user = :auth\_user\_id, 'True', 'False') as ThisUser

FROM posts

JOIN ads ON posts.id = ads.post

JOIN users ON posts.user = users.id;

En esta consulta SQL:

* :auth\_user\_id es un parámetro que representa el ID del usuario autenticado.
* Se seleccionan todas las columnas de la tabla posts y algunas columnas de la tabla users.
* Se realiza un JOIN entre las tablas posts y ads donde posts.id es igual a ads.post.
* Se realiza un JOIN entre las tablas posts y users donde posts.user es igual a users.id.

## 13. Consulta para obtener los posts de un usuario.

SELECT posts.\*,

users.photo as UserPhoto,

users.UserName,

users.name as PersonName,

users.FullName as PersonFullName,

users.subscribed,

IF(posts.user = :auth\_user\_id, 'True', 'False') as ThisUser

FROM posts

JOIN users ON posts.user = users.id

WHERE posts.status = 1

AND posts.user = :user\_id

ORDER BY posts.created\_at DESC

LIMIT 5;

En esta consulta SQL:

* :auth\_user\_id es un parámetro que representa el ID del usuario autenticado.
* :user\_id es un parámetro que representa el ID del usuario cuyo perfil se está viendo.
* Se seleccionan todas las columnas de la tabla posts y algunas columnas de la tabla users.
* Se realiza un JOIN entre las tablas posts y users donde posts.user es igual a users.id.
* Se filtran los resultados para incluir solo aquellos donde posts.status es 1 y posts.user es igual al ID del usuario.
* Se ordenan los resultados por posts.created\_at en orden descendente.
* Se limitan los resultados a los primeros 5 registros.

## 14. Consulta para obtener las reviews hacia un usuario.

SELECT reviews.\*,

users.photo as UserPhoto,

users.name as PersonName,

users.FullName as PersonFullName

FROM reviews

JOIN users ON reviews.to = users.id

WHERE reviews.to = :user\_id

AND reviews.status = 1;

En esta consulta SQL:

* :user\_id es un parámetro que representa el ID del usuario cuyo perfil se está viendo.
* Se seleccionan todas las columnas de la tabla reviews y algunas columnas de la tabla users.
* Se realiza un JOIN entre las tablas reviews y users donde reviews.to es igual a users.id.
* Se filtran los resultados para incluir solo aquellos donde reviews.to es igual al ID del usuario y reviews.status es 1.

## 15. Consulta para obtener los posts anunciados por un usuario.

SELECT ads.\*, posts.\*

FROM ads

JOIN posts ON ads.post = posts.id

WHERE posts.user = :auth\_user\_id

AND ads.status = 1;

En esta consulta SQL:

* :auth\_user\_id es un parámetro que representa el ID del usuario autenticado.
* Se seleccionan todas las columnas de las tablas ads y posts.
* Se realiza un JOIN entre las tablas ads y posts donde ads.post es igual a posts.id.
* Se filtran los resultados para incluir solo aquellos donde posts.user es igual al ID del usuario autenticado y ads.status es 1.

## 16. Consulta para obtener los mensajes de un chat.

SELECT messages.\*,

users.photo as photo,

users.UserName as UserName

FROM messages

JOIN users ON messages.user = users.id

WHERE messages.chat = :chat\_id

AND messages.status != 0;

En esta consulta SQL:

* :chat\_id es un parámetro que representa el ID del chat.
* Se seleccionan todas las columnas de la tabla messages y algunas columnas de la tabla users.
* Se realiza un JOIN entre las tablas messages y users donde messages.user es igual a users.id.
* Se filtran los resultados para incluir solo aquellos donde messages.chat es igual al ID del chat y messages.status es diferente de 0.