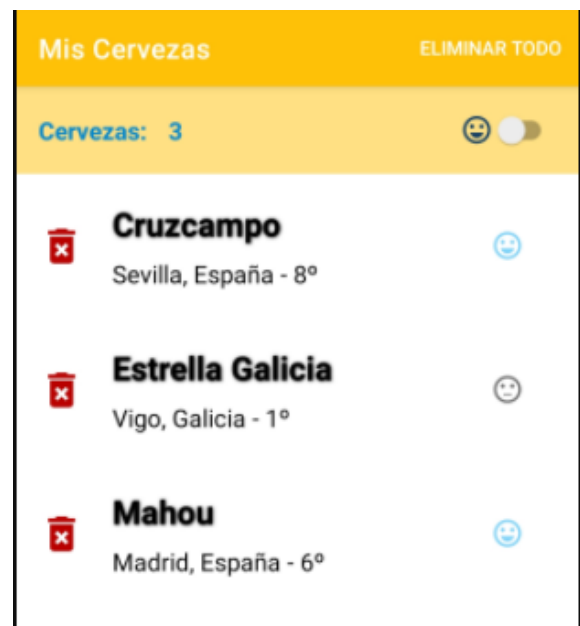


Fase 2: Integración de la ubicación y la toma de imágenes en nuestro proyecto

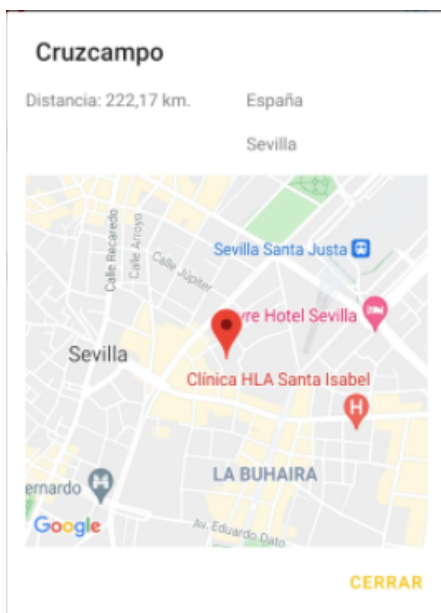
Funcionalidad 1

Por defecto, el listado de ciudad se deberá ordenar alfabéticamente.

Para ello, utiliza la función `sortBy` de `ArrayList` dentro del **observer**, justo antes de pasar la lista al adaptador



Funcionalidad 2



Queremos mostrar para cada ciudad la distancia que hay entre la ciudad y la ubicación actual del usuario. Se mostrará en la pantalla en detalle de cada ciudad.

Para ello, habrá que tener en cuenta varios detalles

1. Al entrar en la actividad de la ciudad, se pedirán permisos para obtener la ubicación actual del usuario.
2. Si se conceden los permisos, se obtendrá la ubicación del usuario en ese momento.
3. Teniendo la ubicación de la ciudad y la del usuario, podemos obtener la distancia entre ellos fácilmente con las funciones **distanceTo** o **distanceBetween** de la clase `Location` (habrá que crear un objeto `Location` con la ubicación de la ciudad)
4. Una vez calculada la distancia, mostrarla en la pantalla.

Funcionalidad 3

Insertar imagen al crear un lugar

- En la tabla de la base de datos debe existir una columna de tipo `String` para almacenar la ruta de la imagen. Si no se incorporó al crear la data class la primera vez, habrá que incluirla ahora.
- En la actividad para crear un `Lugar`, habrá que habilitar un mecanismo para poder añadir una imagen. El usuario podrá elegir una imagen de la galería del dispositivo, o hacer una foto y añadirla a la pantalla.
- Al crear el lugar, habrá que añadir en la tabla, junto al resto de datos, la ruta donde está almacenada la imagen.

También habrá que mostrar esta imagen en la actividad o pantalla donde se muestran todos los detalles de un lugar.

- En la App ya existía una funcionalidad que mostraba toda la información de un sitio al pulsar sobre él en el listado.
- En esta pantalla donde se muestra toda esta información, también habrá que mostrar la imagen. No será complicado, ya que como almacenamos la Uri en la base de datos, únicamente habrá que mostrarla en el ImageView.
- Hay que tener en cuenta que las imágenes, si están en un directorio público, pueden ser eliminadas desde cualquier otra App. Antes de mostrar la imagen, lo ideal es comprobar que existe, para que no ocurra un error.

Funcionalidad extra 1

Obtener la ubicación del usuario cuando se está añadiendo una ciudad y usarla en la latitud y longitud para la ciudad.

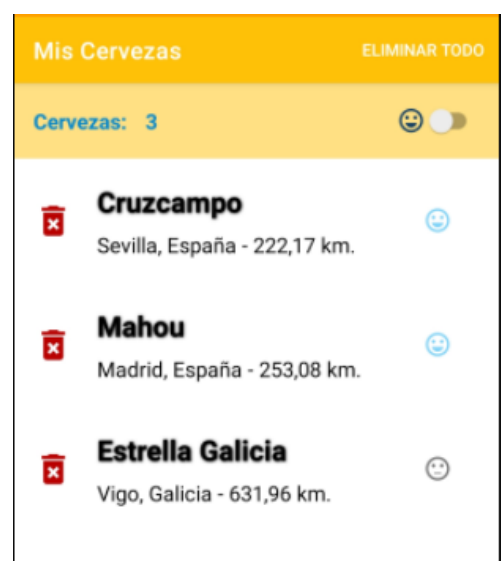
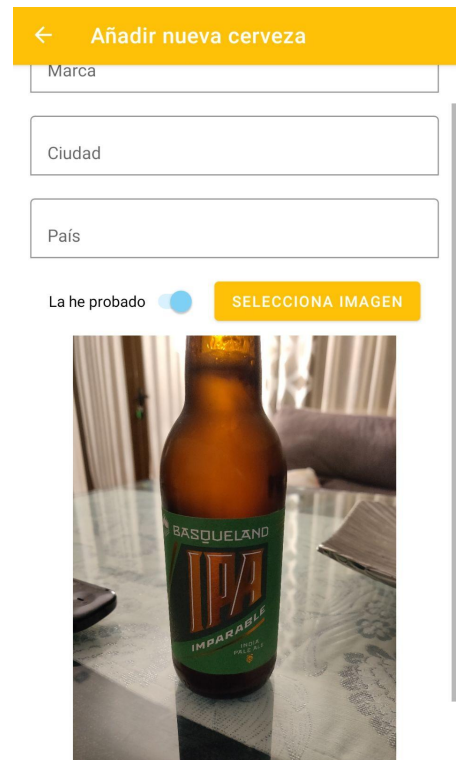
Sería una alternativa al mapa. Cuando el usuario está añadiendo una ciudad nueva, puede seleccionar su ubicación pulsando en el mapa u obteniendo su ubicación actual, ya el usuario se encuentra en la ciudad en el momento de añadirla a la base de datos







Funcionalidad extra 2

Ordenar el listado de ciudades según la distancia que hay entre la ciudad y la ubicación del usuario. Para hacerlo, puedes seguir estos consejos:

1. Crear una nueva columna en la clase Ciudad llamada distancia.que inicialmente valdrá 0.
2. Al entrar en la actividad del listado, se pedirán permisos para obtener la ubicación actual del usuario.
3. Si se conceden los permisos, se obtendrá la ubicación del usuario en ese momento.
4. Crear una función en el ViewModel, a la que se le pasa la ubicación obtenida. Esta función recorre el listado de ciudades, calcula la distancia con la función *distanceTo* de Location, y actualiza el valor de distancia de cada ciudad. Por último, actualizamos la base de datos pasando el listado de ciudades con la distancia calculada.
5. La Query para obtener todas las ciudades puede estar ordenada por distancia. Cuando se actualice el listado, automáticamente el observer obtendrá el listado ordenado.

Funcionalidad extra 3



Mis Cervezas		ELIMINAR TODO
Cervezas: 3		
	Cruzcampo Sevilla, España - 222,17 km.	
	Mahou Madrid, España - 253,08 km.	
	Estrella Galicia Vigo, Galicia - 631,96 km.	

Uno de los problemas de usar el almacenamiento externo es que cualquier App puede acceder a las imágenes.

La funcionalidad opcional será que las imágenes que usa la App tendrán que estar en la carpeta privada de nuestra App.

- Para la cámara, ya lo teníamos así hecho con el FileProvider
- Para la galería, habrá que copiar la imagen seleccionada de la carpeta pública a la privada de la App. Así únicamente nuestra App podrá operar sobre ella^o
 - El mismo FileProvider creado para la cámara servirá para crear URIs a la carpeta privada
 - Al final del documento se adjunta unas funciones que permiten copiar una imagen de la carpeta pública a la privada.
 - Cuando la App obtiene la imagen de la carpeta privada (en el contrato) habrá que llamar a la función `copyImageToPrivateFolder`. A esta función hay que pasarle
 - La Uri de la imagen de la galería
 - Un file temporal creado en la carpeta privada, que será donde se realizará la copia de la imagen de la galería.
 - La App usará directamente la ruta de la nueva imagen en la carpeta privada

```
//dataString es la ruta de la imagen seleccionada de la galería
//fileDestino es un file temporal que hemos creado en la carpeta privada
//esta función convierte la imagen de la galería en un bitmap, y llama a la
función saveImage
//pasándole el bitmap y el file de destino
private fun copyImageToPrivateFolder(dataString: Uri?, fileDestino: File) {
    if (dataString != null) {
        try {
            val inStream: InputStream? =
                contentResolver.openInputStream(dataString)
            val selected_img = BitmapFactory.decodeStream(inStream)
            saveImage(selected_img, fileDestino)
        } catch (e: FileNotFoundException) {
            e.printStackTrace()
            Toast.makeText(this, "An error occurred!", Toast.LENGTH_LONG).show()
        } catch (e: IOException) {
            e.printStackTrace()
        }
    }
}

//esta función copia el bitmap recibido al file destino
@Throws(IOException::class)
private fun saveImage(bitmap: Bitmap, fileDestino: File) {
    val saved: Boolean
    val fos: OutputStream
    fos = FileOutputStream(fileDestino)
    saved = bitmap.compress(Bitmap.CompressFormat.PNG, 100, fos)
    fos.flush()
    fos.close()
    if (saved) Toast.makeText(this, "Todo fue bien", Toast.LENGTH_SHORT).show()
}
```