Proyecto Android Studio
“Foro”

Eduardo Palma

Diseño y análisis de algoritmos

Prof. Dr. Eric Jeltsch F

Estado del arte de la propuesta.

Al ir seleccionando que proyecto realizar, se fueron dando ideas de algo simple pero que tenga relevancia hoy en día, en este caso los foros que rondan por internet, al ser seleccionado esta temática, se toma varias decisiones, empezar desde cero este proyecto ya que en mi opinión si te piden una aplicación del cual necesitas hacerlo de 0 es necesario tener los conocimientos básicos para realizarlo, donde este es el momento para adaptarse a este entorno y empezar una base para el desarrollo.

Dicho este la inspiración parte por foros de internet muy grandes, donde estos de alguna u otra manera ayudan en estos tiempos de pandemia compartir información e incluso socializar con demás usuarios de la plataforma, donde algunos como StackOverflow es una comunidad para responder duda de desarrollo y aprender metodologías nuevas para realizar en proyectos, como estado de la propuesta es estar inspirado en 3 foros.

1. StackOverflow.
2. 4chan
3. Reddit.

Siendo estos la base para el foro y aplicar lo mas simple que se puede aprecias

Propuesta.

Para empezar la idea de crear un foro parte por la forma de compartir opiniones, conocimientos e información referentes a un tema. Esto es inspirado de los Foros mas grandes que hay en internet sea 4Chan, Reddit y el querido por nosotros los desarrolladores StackOverflow, donde cada uno de estos comparten información que les ayudan a otros usuarios otorgándole conocimientos o opiniones dependiendo del tema o el estilo que se creó ese foro.

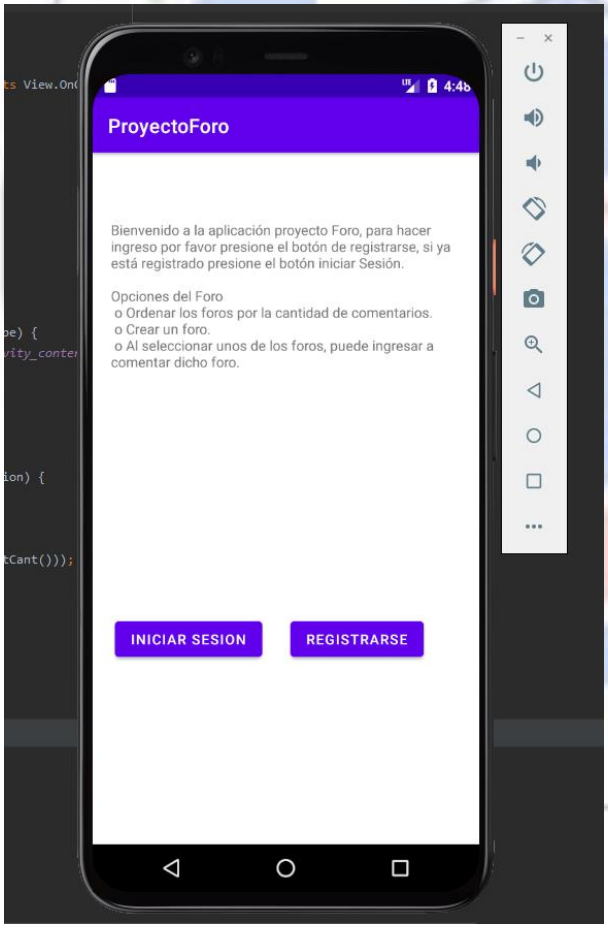
Teniendo la idea principal que es lo que se aborda, la mayoría de estas aplicaciones son de gran envergadura que llevan varios años de desarrollo los puntos mas relevantes es aplicar los conocimientos que se posee en cómo manejar la cantidad enorme de usuarios que pueden ingresar, como se manejan la forma en cual se muestran los foros y como son visualizados, además de agregar los algoritmos que ocupan para estas soluciones.

La problemática en cuestión que me di es crear un foro donde se guardan usuarios, foros y comentarios, donde estos usuarios pueden crear un foro, comentarlo y aplicar acciones que los lleven al foro seleccionado, la idea es que el usuario se registre, acceda al sistema de los foros que existen seleccione alguno o simplemente los crea y comenta.

Solución propuesta

Al ir realizando el proyecto se hizo un esquema de como es la forma en que el usuario debería registrarse para acceder a el foro, también al acceder puede realizar las siguientes acciones.

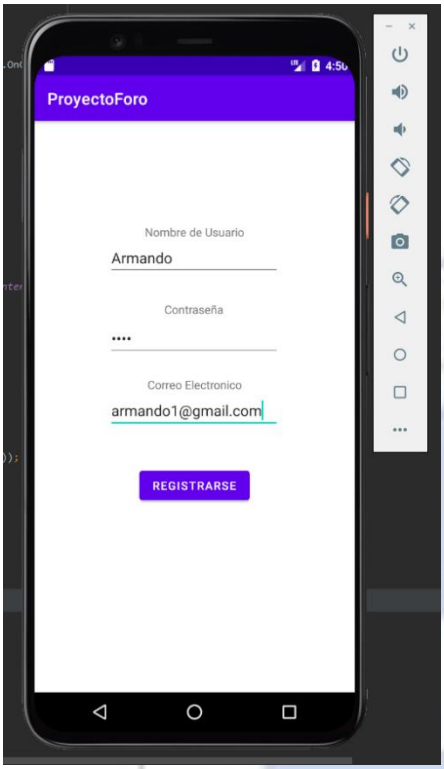
- Iniciar sesión y Registrarse
- En Registrarse el usuario registra su cuenta y al ser agregado se direcciona a la pantalla de inicio de sesión en el cual el usuario tiene que ingresar sus datos para el ingreso.
- Inicio sesión ingresar los datos correspondientes y hacer ingreso a la plataforma.

al 

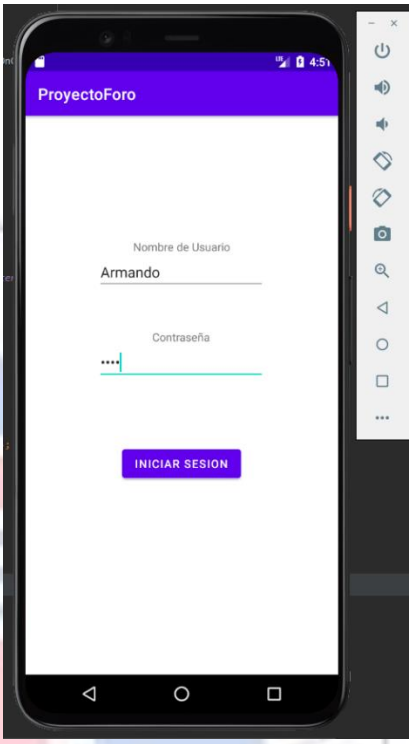
iniciar la aplicación muestra un mensaje de bienvenida y los botones correspondiente dependiendo de la acción sea iniciar sesión y registrarse.

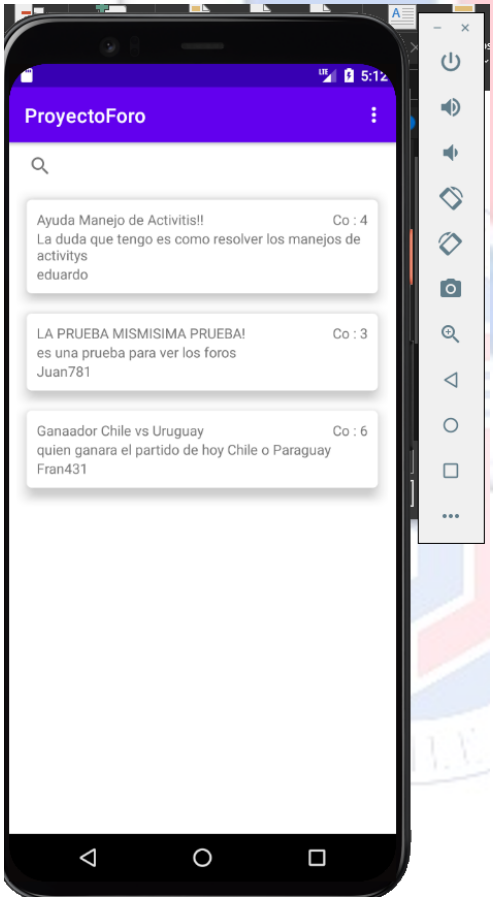
Opciones del Foro

- Ordenar los foros por la cantidad de comentarios.
- Crear un foro.
- Al seleccionar unos de los foros, puede ingresar a comentar dicho foro.



seleccionando el botón registrarse le direcciona a el panel para el ingreso de datos de registro luego de esto le manda al panel de inicio de sesión en el cual se manda los datos para hacer el correspondiente ingreso

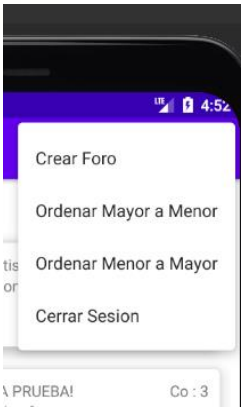




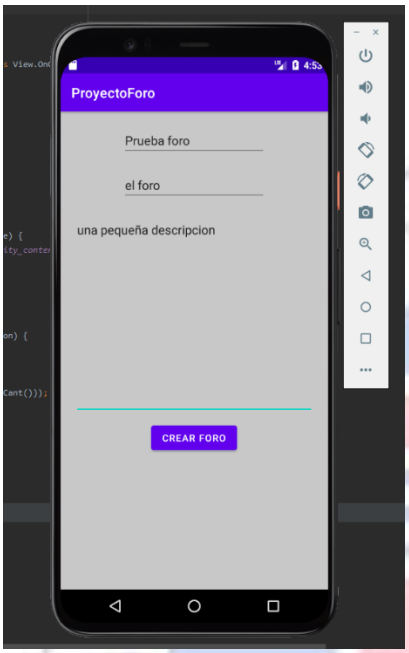
Ya al hacer ingreso al sistema se le muestras los foros que están, donde se pueden seleccionar y ver sus comentarios, estos foros poseen un título la descripción de ese foro y el usuario en el cual lo hizo, también se implementaría una búsqueda para una secuencia de foros pero por tema de tiempo no se alcanzo a desarrollar.

También existe un panel de opciones en la parte superior derecha del cual son.

- Crear Foro
- Ordenar mayor a menor
- Ordenar menor a mayor
- Cerrar sesión.



Opción crear foro

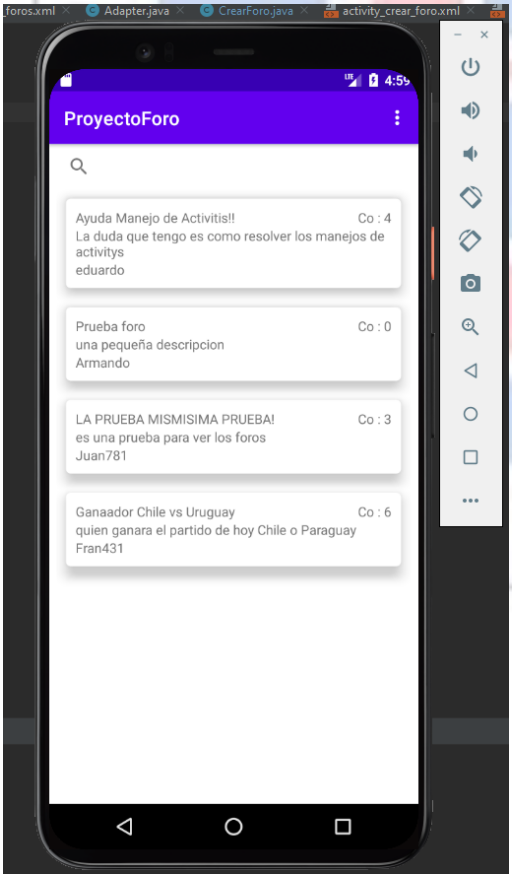


al seleccionarla lo redirige para crear el foro con los datos correspondientes.

Se hace una prueba.

- Título: Prueba Foro
- Tema: el foro
- Descripción: una pequeña descripción.

Al crearlo de agrega a la visualización de los foros.

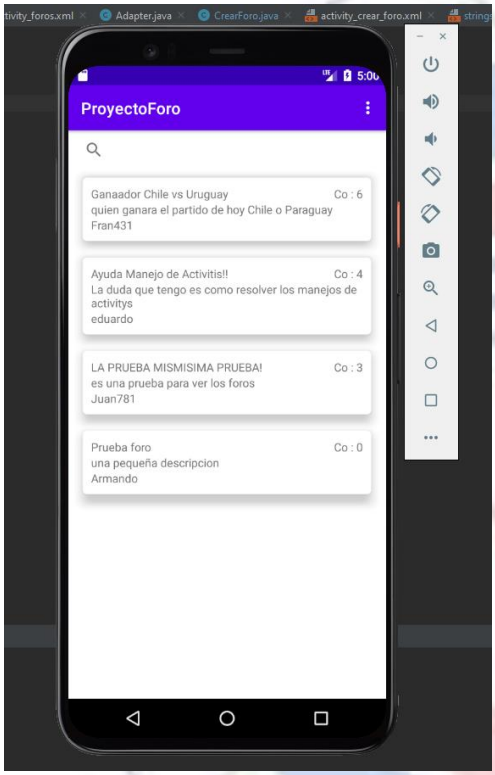


se muestra el foro creado por el usuario en este caso armando y su titulo u descripción mostrando su cantidad de comentarios en este caso 0.

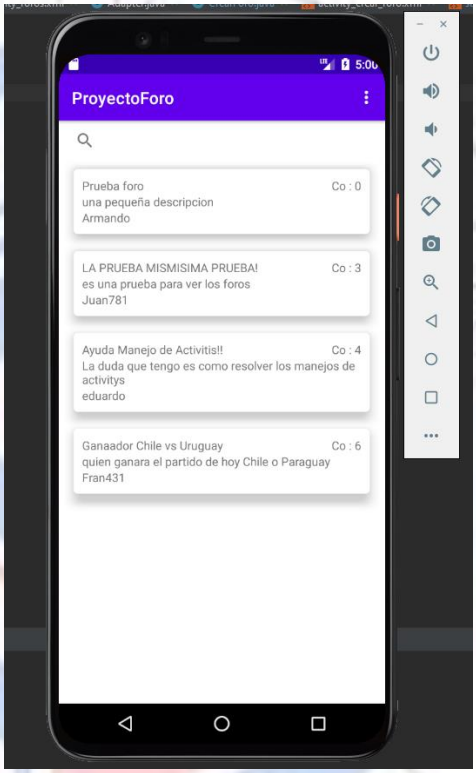
Opción del ordenar mayor menor y viceversa.

al ser selecciona cualquier de los dos los foros pasan a ser ordenados por la cantidad de comentarios dados.

Mayor a menor

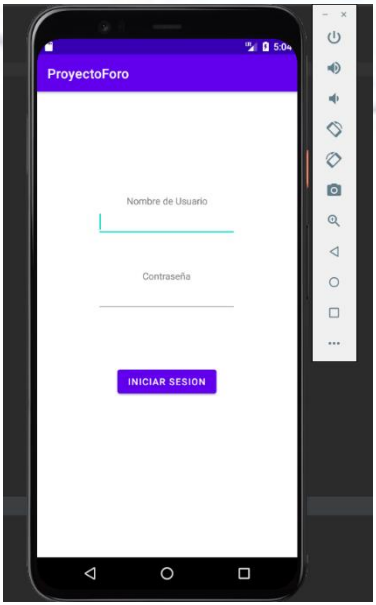


Menor a Mayor



Opción cerrar Sesión.

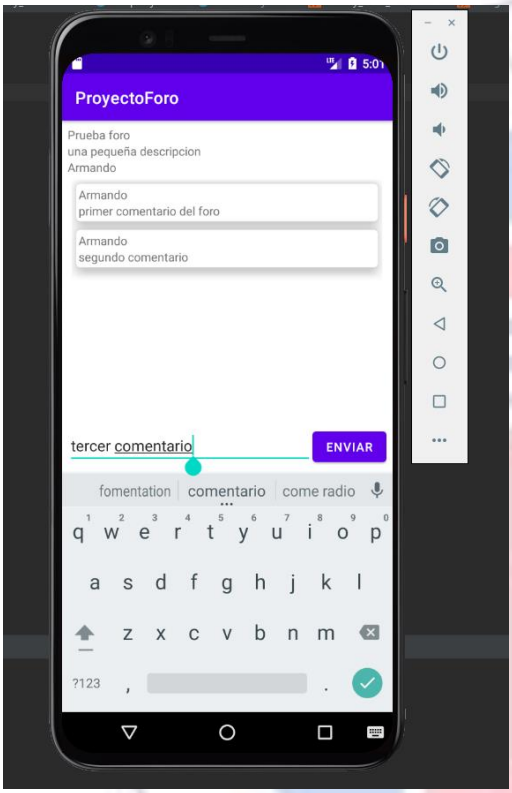
Para este caso solo se cierra la sesión y vuelve a el panel iniciar sesión



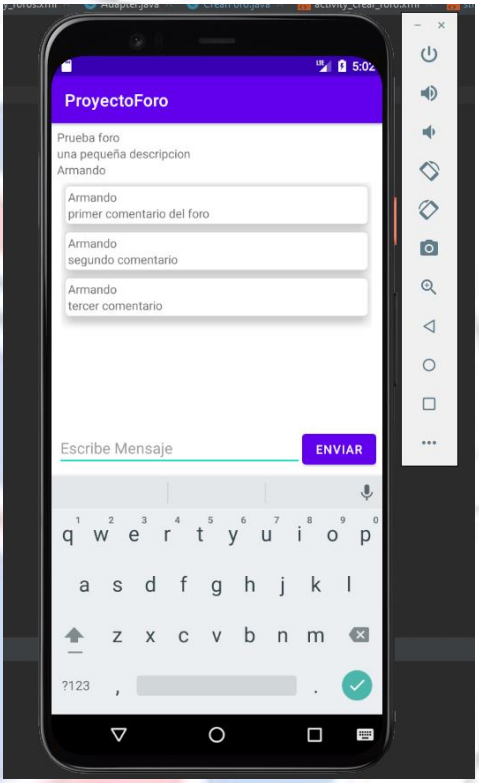
Al seleccionar un foro.

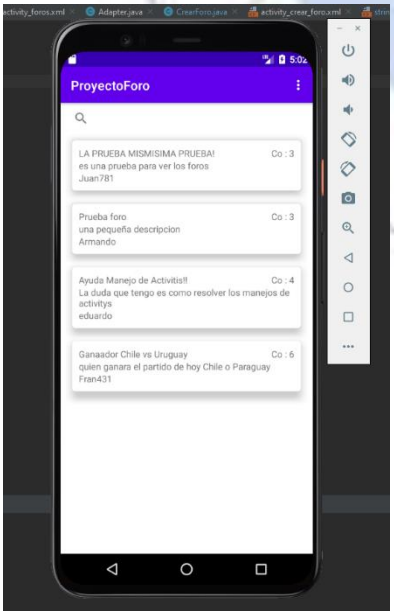
Al seleccionarlo se muestra el foro con sus comentarios que posee al hacer un comentario se ingresa y se va actualizando la cantidad de comentarios que posee

Comentario a ingresar

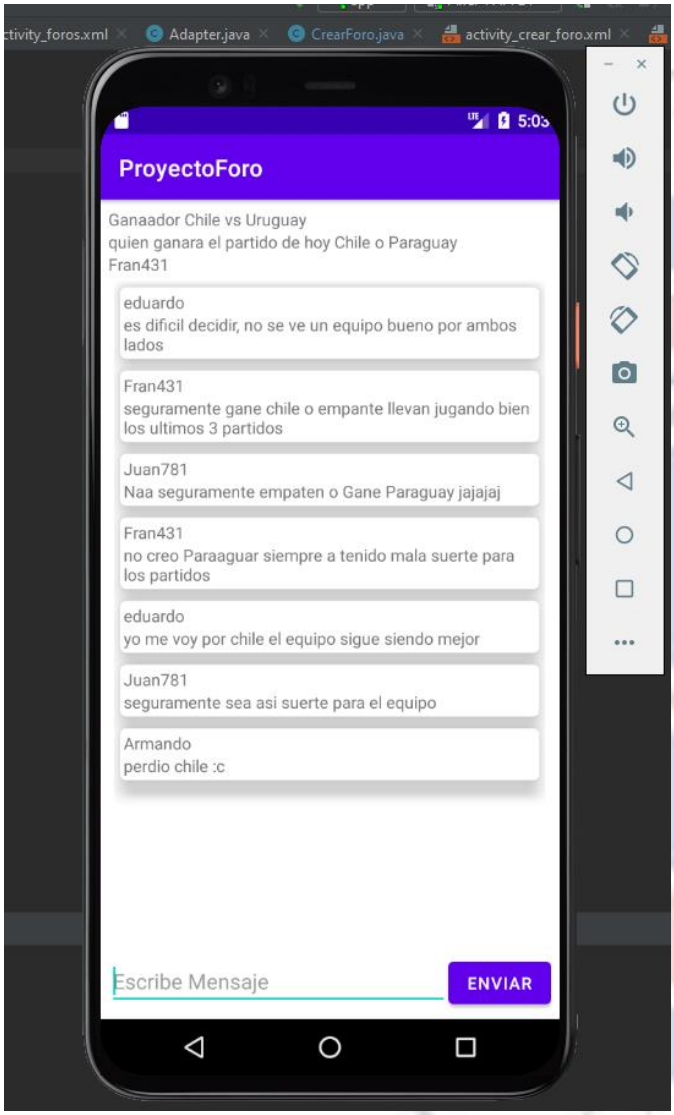


Comentario ya ingresado

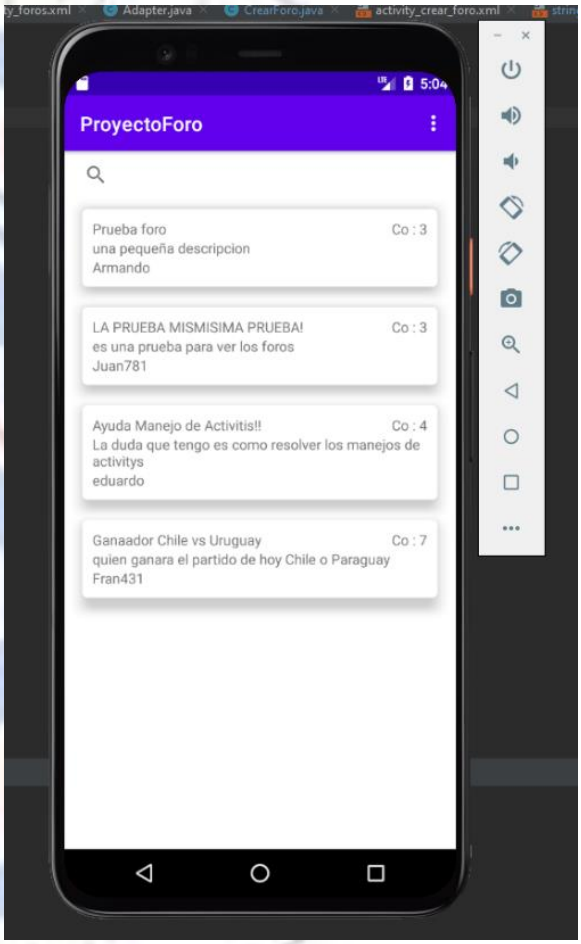




Al volver del foro y haber ingresado esos comentarios. Se muestran los foros actualizados con el comentario ya ingresado.



Otro ejemplo de ingresar a un foro y comentarlo, al haber ingresado perdió chile el usuario Armando se actualiza el foro.



Algoritmos y Estructuras de datos

Para la propuesta de aplicación se desarrollaron 3 estructuras para el desarrollo de estas.

1. **Estructura de datos tabla Hash**: se incorpora la estructura de datos tablas hash para el guardado eh ingreso de Usuarios a la aplicación, donde como la cantidad de usuarios puede ser muy grande, el ingreso de usuario y el buscar usuario es mas veloz, siento esta estructura para el mejor manejo de estos, implementando una función hash que funcionara de manera que el nombre de usuario sea el identificador para esto, también se trabaja con colisiones mediante encadenamiento haciendo una lista para el manejo de la colisión.

```java
public class TablaHash implements Serializable {
    private ListaUsuario arregloUsuarios[];
    private int tam;

    public TablaHash(int tam){
        this.tam = tam;
        this.arregloUsuarios = new ListaUsuario[tam];
        for(int i=0;i<tam;i++){
            arregloUsuarios[i] = new ListaUsuario();
        }
    }

    public int hash(String nombreUsuario){
        int valor = 0;
        for(int i=0;i<nombreUsuario.length();i++){
            char c = nombreUsuario.charAt(i);
            int ascii = (int) c;
            valor = valor + ascii;
        }

        return valor%this.tam;
    }

    public boolean ingresar(Usuario usuario){
        int index = hash(usuario.getNombreUsuario());
        this.arregloUsuarios[index].ingresarUsuario(usuario);
        return true;
    }

    public Usuario buscar(String nombreUsuario){
        int index = hash(nombreUsuario);
        Usuario u = arregloUsuarios[index].buscarUsuario(nombreUsuario);
        if(u == null) return null;
        else return u;
    }

    public boolean eliminar(String nombreUsuario){
        int index = hash(nombreUsuario);
        if(this.arregloUsuarios[index].getPrimero() == null){
            return false;
        }else{
            if(this.arregloUsuarios[index].eliminar(nombreUsuario)) return true;
            else return false;
        }
    }
```

2. **Estructura de datos lista:** para este caso se ocupa en mayor medida para guardar los comentarios ingresados en el foro donde estos como se tiene que ir ingresando de manera desde el más antiguo al más nuevo, es el más óptimo para el desarrollo de estas.

```
public class ListaComentario implements Serializable {
    private NodoLista primero;
    private NodoLista ultimo;
    private int cant;

    public ListaComentario(){
        this.primero = null;
        this.ultimo = null;
        this.cant = 0;
    }

    public boolean ingresarComentario(Comentarios comentario){
        NodoLista nuevo = new NodoLista(comentario);
        if(this.primero == null){
            this.primero = nuevo;
            this.ultimo = nuevo;
            this.cant = 1;
            return true;
        }else{
            nuevo.setPrev(this.ultimo);
            this.ultimo.setNext(nuevo);
            this.ultimo = nuevo;
            this.cant++;
            return true;
        }
    }

    public Comentarios buscarComentario(String comentario){
        NodoLista aux = this.primero;
        while(aux != null){
            if(aux.getComentario().getTexto().equalsIgnoreCase(comentario)){
                return aux.getComentario();
            }
            aux = aux.getNext();
        }
        return null;
    }
```

3. **Estructura de datos ArbolAVL**: este árbol se ocupa para el guardado de foros donde son ordenados por la cantidad de comentarios, la elección es por la modalidad que se puede mostrar el árbol sea de mayor o menor y viceversa, porque el árbol AVL y no el ABB la idea también es implementar otro tipo de estructura donde este balanceado y sea lo mas optimo para el manejo de recorridos y búsqueda.

```
public class Arbol implements Serializable {
    private NodoArbol raiz;
    private int cant;

    public Arbol() { this.raiz = null; }

    public int obtenerFe(NodoArbol nodo){
        if(nodo == null){
            return -1;
        }else return nodo.getFe();
    }

    public NodoArbol rotacionIzquierda(NodoArbol nodo){
        NodoArbol aux = nodo.getHijoIzquierdo();
        nodo.setHijoIzquierdo(aux.getHijoDerecho());
        aux.setHijoDerecho(nodo);
        nodo.setFe(Math.max(obtenerFe(nodo.getHijoIzquierdo()),obtenerFe(nodo.getHijoDerecho()))+1);
        aux.setFe(Math.max(obtenerFe(aux.getHijoIzquierdo()), obtenerFe(aux.getHijoDerecho()))+1);
        return aux;
    }

    public NodoArbol rotacionDerecha(NodoArbol nodo){
        NodoArbol aux = nodo.getHijoDerecho();
        nodo.setHijoDerecho(aux.getHijoIzquierdo());
        aux.setHijoIzquierdo(nodo);
        nodo.setFe(Math.max(obtenerFe(nodo.getHijoIzquierdo()), obtenerFe(nodo.getHijoDerecho()))+1);
        aux.setFe(Math.max(obtenerFe(aux.getHijoIzquierdo()),obtenerFe(aux.getHijoDerecho()))+1);
        return aux;
    }

    public NodoArbol rotacionDobleIzquierda(NodoArbol nodo){
        NodoArbol temp;
        nodo.setHijoIzquierdo(rotacionDerecha(nodo.getHijoIzquierdo()));
        temp = rotacionIzquierda(nodo);
        return temp;
    }

    public NodoArbol rotacionDobleDerecha(NodoArbol nodo){
        NodoArbol temp;
        nodo.setHijoDerecho(rotacionIzquierda(nodo.getHijoDerecho()));
        temp = rotacionDerecha(nodo);
        return temp;
    }
```

Método para insertar árbol donde cada vez que se ingrese se tiene que actualizar su altura y hacer rotaciones dependiendo el caso.

```
public void insertar(Foro foro){
    NodoArbol nuevo = new NodoArbol(foro);
    if(this.raiz == null) this.raiz = nuevo;
    else raiz = insertarArbol(nuevo,this.raiz);
}

private NodoArbol insertarArbol(NodoArbol nuevo,NodoArbol raiz){
    NodoArbol nuevaRaiz = raiz;
    if(nuevo.getForo().getLc().getCant() < raiz.getForo().getLc().getCant()){
        if(raiz.getHijoIzquierdo() == null){
            raiz.setHijoIzquierdo(nuevo);
        }else{
            raiz.setHijoIzquierdo(insertarArbol(nuevo,raiz.getHijoIzquierdo()));
            if((obtenerFe(raiz.getHijoIzquierdo()) - obtenerFe(raiz.getHijoDerecho()) ) == 2){
                if(nuevo.getForo().getLc().getCant() < raiz.getHijoIzquierdo().getForo().getLc().getCant()){
                    nuevaRaiz = rotacionDobleIzquierda(raiz);
                }else nuevaRaiz = rotacionDobleIzquierda(raiz);
            }
        }
    }else{
        if(nuevo.getForo().getLc().getCant() >= raiz.getForo().getLc().getCant()){
            if(raiz.getHijoDerecho() == null){
                raiz.setHijoDerecho(nuevo);
            }else{
                raiz.setHijoDerecho(insertarArbol(nuevo, raiz.getHijoDerecho()));
                if((obtenerFe(raiz.getHijoDerecho()) - obtenerFe(raiz.getHijoIzquierdo())) == 2){
                    if(nuevo.getForo().getLc().getCant() > raiz.getHijoDerecho().getForo().getLc().getCant()){
                        nuevaRaiz = rotacionDerecha(raiz);
                    }else nuevaRaiz = rotacionDobleDerecha(raiz);
                }
            }
        }
    }

    //actualizar altura
    if((raiz.getHijoIzquierdo() == null) && (raiz.getHijoDerecho() != null)){
        raiz.setFe(raiz.getHijoDerecho().getFe()+1);
    }else if((raiz.getHijoDerecho() == null) && (raiz.getHijoIzquierdo() != null)){
        raiz.setFe(raiz.getHijoIzquierdo().getFe()+1);
    }else{
        raiz.setFe(Math.max(obtenerFe(raiz.getHijoIzquierdo()), obtenerFe(raiz.getHijoDerecho())) +1);
    }
    this.cant++;
    return nuevaRaiz;
```

4. **Algoritmo de búsqueda binaria**: se ocupa para la búsqueda de un foro en cuestión como el árbol es binario es el mejor algoritmo de búsqueda para este caso, siendo el más óptimo para resolver la búsqueda de un árbol binario.

```
public Foro buscarForo(int cant) { return buscarForo(cant,this.raiz); }
private Foro buscarForo(int cant,NodoArbol raiz){
    if(raiz == null) return null;
    else{
        if(raiz.getForo().getLc().getCant() == cant) return raiz.getForo();
        else{
            if(raiz.getForo().getLc().getCant() > cant) return buscarForo(cant,raiz.getHijoIzquierdo());
            else return buscarForo(cant,raiz.getHijoDerecho());
        }
    }

}
```

Resultados.

Al ir desarrollando me di cuenta de lo complejo que es ir adaptándose a nuevos entornos donde al comienzo cuesta un poco pero después es algo sencillo de aplicar, los resultados de mi propuesta fueron resueltos de manera eficaz sin ningún contratiempo y con un final de proyecto bastante bueno al haberlo desarrollado desde 0, con ganas de seguir aplicando y terminarlo con muchas funcionalidades más que falto aplicar e implementar.

Una buenas prácticas y consejos para el desarrollo de estos proyectos en Android.

- la idea es darse cosas básicas para ver las funcionalidades y después aplicarlas en el proyecto principal
- buscar documentación para el manejo de Activitys y formas de pasar información a través de ellas y contenedores de visualizaciones.
- También saber cómo funcionan las interfaces graficas de Android específicamente los archivos XML donde es como se vera la aplicación al final.
- Manera de manejar estructuras de datos, debido a que no es fácil manejarlas, ya que la mayoría de los datos en estas aplicaciones están asociadas a base de datos, donde se hacen llamadas para obtener información y guardarlas, cosas que no se implementó por tiempo en el desarrollo.

Para concluir este fue una de las experiencias en desarrollo más grande donde me da una base para el desarrollo de software en aplicaciones móviles, donde talvez en un futuro sea el campo que me dedique siendo que los smartphones cada día son más utilizados, y las aplicaciones para este entorno son más pedidas.

Repositorio GitHub de la aplicación.

https://github.com/EduardoPalma/ProyectoForo

Referencias.

- "*manejo de Activitys*" https://developer.android.com/guide/components/activities/intro-activities?hl=es
- "*Paso de datos entre Activitys*" https://es.stackoverflow.com/questions/36902/como-enviar-datos-entre-activities
- "*listas dinámicas Recyclerview*" https://developer.android.com/guide/topics/ui/layout/recyclerview?hl=es-419
- "*Botón de retroceso Android*" https://living-sun.com/es/android/45917-onkeydown-or-onbackpressed-android-back-button.html
- "*Menu Android Studio*" https://developer.android.com/guide/topics/ui/menus?hl=es-419#options-menu
- *"CardView"* https://developer.android.com/reference/android/support/v7/widget/CardView.html
- *"Pasar Objetos entre Activitys"* https://developer.android.com/guide/components/activities/parcelables-and-bundles?hl=es-419
- "*manejo de interfaces Android studio*" https://developer.android.com/guide/topics/ui/declaring-layout?hl=es-419