* proposito

app que permite contratar servicios (técnicos, ingenieros, abogados, etc.). mostrará el perfil de los usuarios con la información que se quiera mostrar, ya sea presupuestos e historia de trabajos.

Uno de los métodos más utilizados sea para subir o cargar información de la base de datos

private void accederalDB(String accion)  
{  
 StringRequest stringRequest = new StringRequest(Request.Method.*POST*, link + accion, new Response.Listener<String>()  
 {  
 @Override  
 public void onResponse(String response)  
 {  
 if(!response.isEmpty())  
 {  
 //Recibe la información de la base de datps   
 }  
  
 }  
 }, new Response.ErrorListener()  
 {  
 @Override  
 public void onErrorResponse(VolleyError error)  
 {  
 mensajes(error.getMessage());  
 }  
 })  
 {  
 @Nullable  
 @Override  
 protected Map<String, String> getParams() throws AuthFailureError  
 {  
 Map<String, String> parametros = new HashMap<String, String>();  
  
 //en este se emvian los datos requiere el servidor.

// para esto se utiliza este código:  
 parametros.put("Nombre\_Etiqueta", datos a enviar);  
 return parametros;  
 }  
 };  
 RequestQueue rq = Volley.*newRequestQueue*(this);  
 rq.add(stringRequest);  
}

EL sub-metodo onResponse recibe la información en formato Json. Para poderlo extraer se utiliza este método como ejemplo:

public void extraerdatos( String dato )  
{  
 try  
 {  
 JSONObject response = new JSONObject(dato);  
 JSONArray jsonArray = response.optJSONArray("datos");  
 JSONObject jsonObject = null;  
 jsonObject = jsonArray.getJSONObject(0);  
  
 cargar\_header( jsonObject.optString("id"),  
 jsonObject.optString("nombres"),  
 jsonObject.optString("apellidoP"),  
 jsonObject.optString("apellidoM"),  
 jsonObject.optString("profesion"),  
 jsonObject.optString("imagen"),  
 jsonObject.optString("RFC"),  
 jsonObject.optString("comprobado")  
 );  
 }  
 catch (JSONException e)  
 {  
 mensajes("Error de Jason: " + e.toString());  
 }  
}

Para que nos muestre los datos en lista, se utiliza un RecicleView. Para asignare un formato se utilizan dos clases (ListElement y ListAdapter):

El ListElemente guarda la información para poderla carga en el ListElement.

public class ListElement implements Serializable //ofertas  
{  
 public String ID, user, imagen, titulo, descripcion, info\_genera, profecion\_req, costo, vacantes;  
  
 public Boolean es\_usuario;  
  
 public ListElement(String ID, String user ,String imagen, String titulo, String descripcion,String profecion\_req, String info\_genera,String costo, String vacantes, Boolean es\_usuario)  
 {  
 this.ID = ID;  
 this.user = user;  
 this.imagen = imagen;  
 this.titulo = titulo;  
 this.descripcion = descripcion;  
 this.profecion\_req = profecion\_req;  
 this.info\_genera = info\_genera;  
 this.costo = costo;  
 this.vacantes = vacantes;  
 this.es\_usuario = es\_usuario;  
 }  
  
 public ListElement()  
 {  
  
 }  
  
 public String getCosto() {  
 return costo;  
 }  
  
 public void setCosto(String costo) {  
 this.costo = costo;  
 }  
  
 public String getVacantes() {  
 return vacantes;  
 }  
  
 public void setVacantes(String vacantes) {  
 this.vacantes = vacantes;  
 }  
  
 public String getID() {  
 return ID;  
 }  
  
 public void setID(String ID) {  
 this.ID = ID;  
 }  
  
 public String getUser() {  
 return user;  
 }  
  
 public void setUser(String user) {  
 this.user = user;  
 }  
  
 public String getInfo\_genera() {  
 return info\_genera;  
 }  
  
 public String getProfecion\_req() {  
 return profecion\_req;  
 }  
  
 public void setProfecion\_req(String profecion\_req) {  
 this.profecion\_req = profecion\_req;  
 }  
  
 public void setInfo\_genera(String info\_genera) {  
 this.info\_genera = info\_genera;  
 }  
  
 public String getImagen()  
 {  
 return imagen;  
 }  
  
 public void setImagen(String imagen)  
 {  
 this.imagen = imagen;  
 }  
  
 public String getTitulo()  
 {  
 return titulo;  
 }  
  
 public void setTitulo(String titulo)  
 {  
 this.titulo = titulo;  
 }  
  
 public String getDescripcion()  
 {  
 return descripcion;  
 }  
  
 public void setDescripcion(String descripcion)  
 {  
 this.descripcion = descripcion;  
 }  
  
 public Boolean getEs\_usuario() {  
 return es\_usuario;  
 }  
  
 public void setEs\_usuario(Boolean es\_usuario) {  
 this.es\_usuario = es\_usuario;  
 }  
}

El ListAdapter extrae la información, el formato y lo carga en el RecicleView.

mData carga la información del ListElement.

mInflate muestra el formato y la informacion en el RecicleView

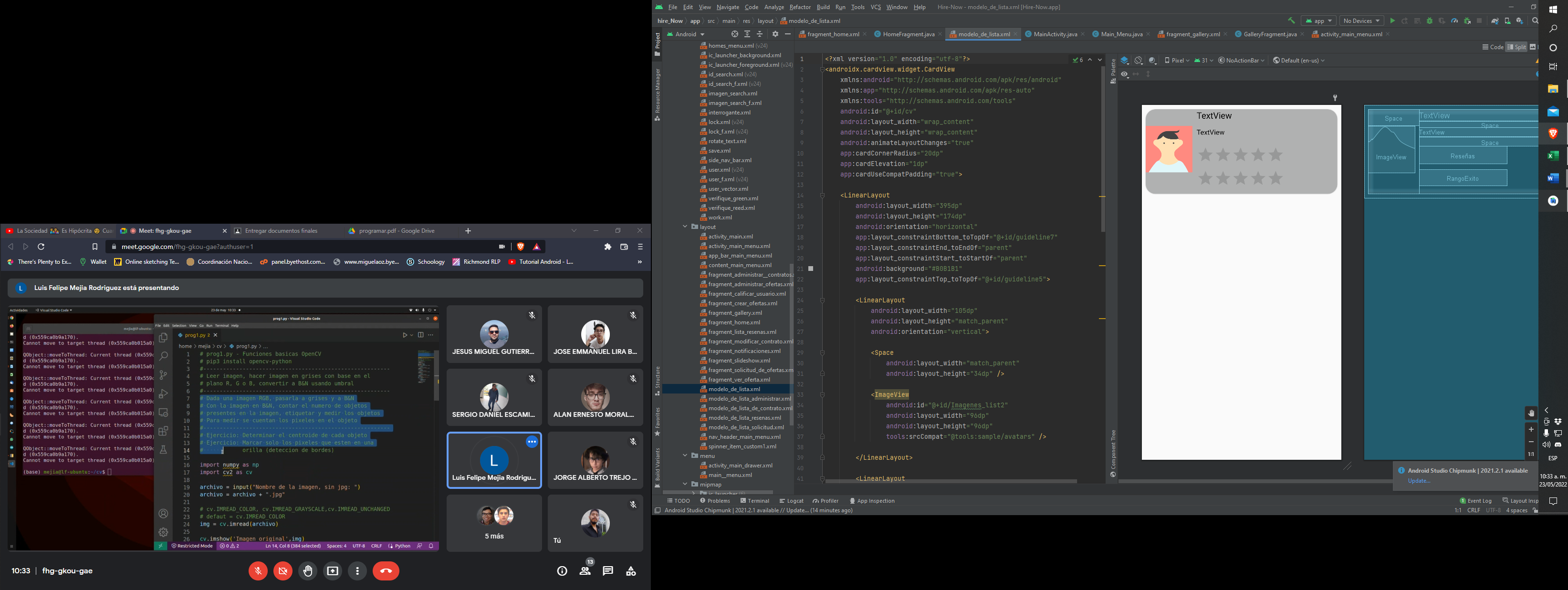
context la ubicación donde se mostrará la información.

Listener permite hacer clic en la lista para realizar una accion.

Skin\_list es el formato de la lista.

public class ListAdapter\_Solicitud extends RecyclerView.Adapter<ListAdapter\_Solicitud.ViewHolder>  
{  
 private List<ListElement\_Solicitud> mData;  
 private LayoutInflater mInflate;  
 private Context context;  
 final ListAdapter\_Solicitud.OnItemClickListener listener;  
 int skin\_list;  
  
 public interface OnItemClickListener  
 {  
 void onItemClick(ListElement\_Solicitud item);  
 }  
  
 public ListAdapter\_Solicitud(int skin\_list, List<ListElement\_Solicitud> itemList, Context context, ListAdapter\_Solicitud.OnItemClickListener listener)  
 {  
 this.mInflate = LayoutInflater.*from*(context);  
 this.context = context;  
 this.mData = itemList;  
 this.listener = listener;  
 this.skin\_list = skin\_list;  
 }  
  
 @Override  
 public int getItemCount()  
 {  
 return mData.size();  
 }  
  
 @Override  
 public ListAdapter\_Solicitud.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType)  
 {  
 View view = mInflate.inflate(skin\_list, null);  
 return new ListAdapter\_Solicitud.ViewHolder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(final ListAdapter\_Solicitud.ViewHolder holder, final int position)  
 {  
 holder.bindData(mData.get(position));  
 }  
  
 public void setItem(List<ListElement\_Solicitud> items)  
 {  
 mData = items;  
 }  
  
 public class ViewHolder extends RecyclerView.ViewHolder  
 {  
 ImageView imagen;  
 TextView id\_user\_notificacion, titulo, tipo;  
  
 ViewHolder(View itemView)  
 {  
 super(itemView);  
 }  
  
 void bindData(final ListElement\_Solicitud item)  
 {  
 imagen = itemView.findViewById(R.id.*Imagenes\_list2*);  
 titulo = itemView.findViewById(R.id.*titulo2*);  
 id\_user\_notificacion = itemView.findViewById(R.id.*usuario*);  
 tipo = itemView.findViewById(R.id.*tipo*);  
  
 titulo.setText(item.getTitulo());  
 id\_user\_notificacion.setText(item.getID\_user\_notificacion());  
 tipo.setText(item.getTipo());  
  
 Uri urlparse = Uri.*parse*(item.getImagen());  
 Glide.*with*(context).load(urlparse).error(R.drawable.*work*).into(imagen);  
  
 itemView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 listener.onItemClick(item);  
 }  
 });  
 }  
  
 public void mensajes(String mensaje)  
 {  
 Toast.*makeText*(context, mensaje, Toast.*LENGTH\_LONG*).show();  
 }  
  
 }  
}

Formato de la lista:



Este es el método el cual extrae los datos y los agrega al RecicleView.

public void extraerdatos( String dato )  
{   
 elementos = new ArrayList<>();//genera un array para la lista  
 try  
 {  
 JSONObject response = new JSONObject(dato);  
 JSONArray ofertas = response.getJSONArray("datos");  
 for(int i = 0; i < ofertas.length(); i++)  
 {  
 JSONObject jsonObject = null;  
 jsonObject = ofertas.getJSONObject(i);  
  
 elementos.add( new ListElement( //carga los datos  
jsonObject.optString("ID"), //String ID,  
 jsonObject.optString("id\_user"), //String user,  
 jsonObject.optString("Imagen"), // String imagen,  
 jsonObject.optString("Titulo"), //String titulo,  
 jsonObject.optString("descripcion"), // String descripcion,  
 jsonObject.optString("profecion\_requerida"), //String profecion\_req,  
 jsonObject.optString("Informacion\_general"), //String info\_genera  
 jsonObject.optString("pago"), //string costos  
 jsonObject.optString("vacantes"), //string vacantes  
 false)  
 );  
//A qui se carda el formato y la informacion al ListAdapter  
ListAdapter listAdapter =

new ListAdapter(R.layout.*modelo\_de\_lista*, //formato

elementos, //datos

getContext(),//ubicación

new ListAdapter.OnItemClickListener() //Clic

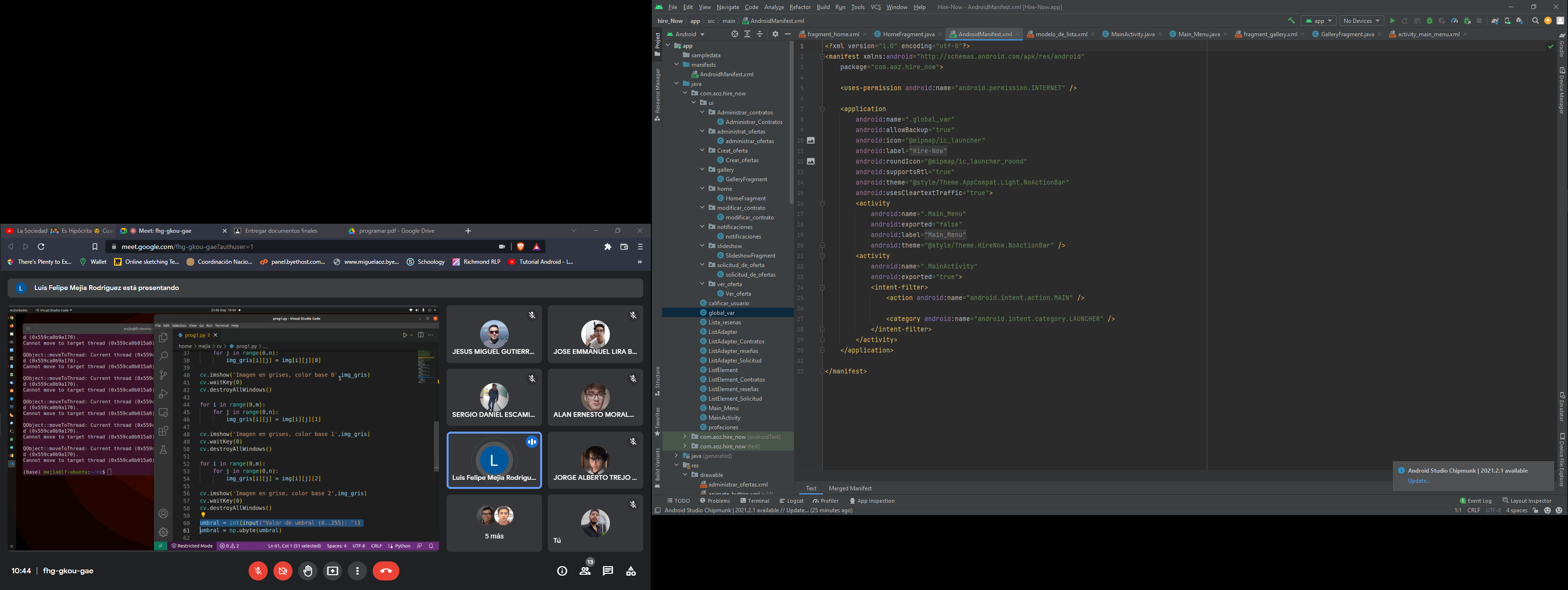
{  
 @Override  
 public void onItemClick(ListElement item) //si se da clic  
 {  
 abrirfragent\_veroferta(item);  
 }  
 });  
// Carga todo al RecicleView  
RecyclerView recyclerView =

(RecyclerView) rootView.findViewById(R.id.*Lista\_ofertas*); //nombre del RV  
 recyclerView.setHasFixedSize(true);  
 recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));  
 recyclerView.setAdapter(listAdapter); //carga el ListAdapter  
 }  
 }  
 catch (JSONException e) // si falla  
 {  
 //e.printStackTrace();  
 mensajes("Error de Jason: " + e.toString());  
 }  
}

Para facilitar el trabajo se creo esta clase para generar variables globales.

package com.aoz.hire\_now;  
  
import android.app.Application;  
  
public class global\_var extends Application {  
  
 public static String *user*, *Link*;  
  
  
 public static String getUser() {  
 return *user*;  
 }  
  
 public static void setUser(String user) {  
 global\_var.*user* = user;  
 }  
  
 public static String getLink() {  
 return *Link*;  
 }  
  
 public static void setLink(String link) {  
 *Link* = link;  
 }  
}

para que no nos de ningún error la llamamos en el manifest.



* Esquema lógico.

Ingreso/creación de cuenta

Carga de datos en la base de datos

Editar/mostrar perfil

Mostrar/hacer ofertas

Solicitar oferta/profesional

* Fórmulas y procesos complejos:

No se utilizaros.

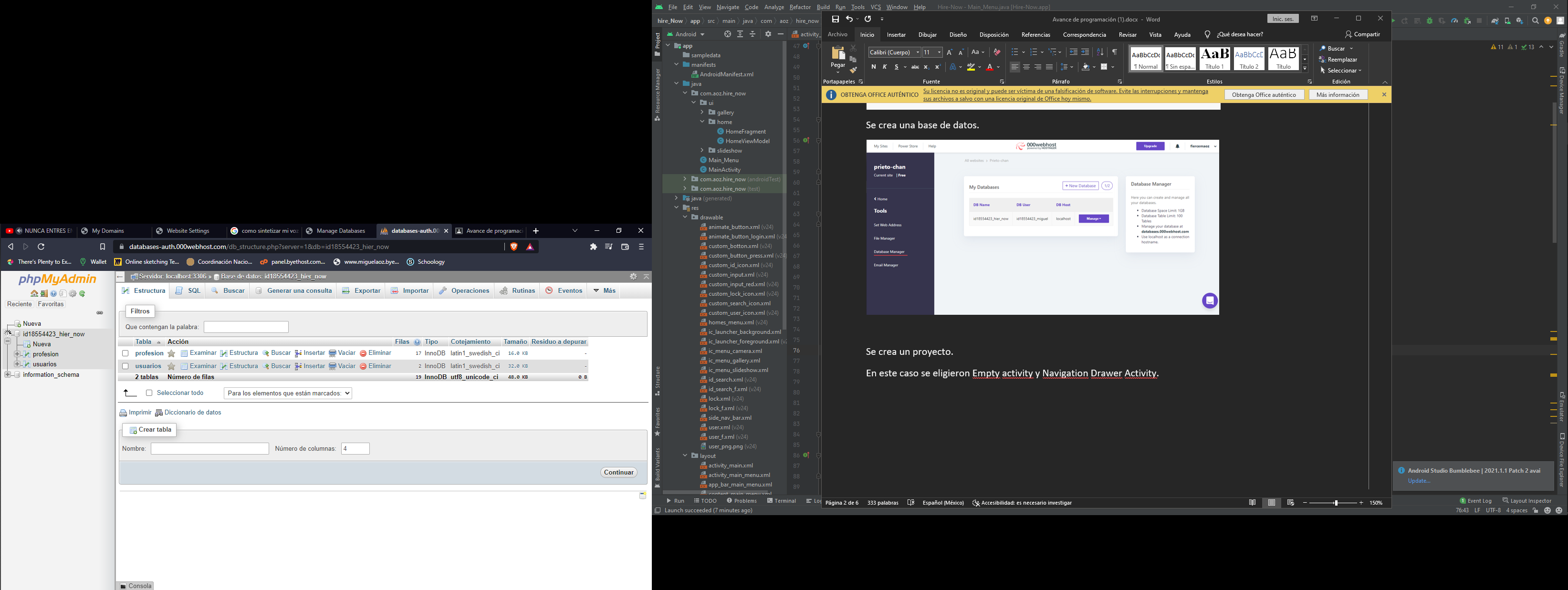
* Librerías requeridas

volley: permite las subida y bajada de la base de datos mediante la web.

Glide: permite cargar imágenes en la nube sin necesidad de descarga.

* Datos utilizados en las pruebas

Se utilizo una base de datos en la web para extraer datos como usuario, contraseña, profesión etc.



* Puntos débiles y futuras expansiones.

Débiles.

* + Falta que el usuario vea sus reseñas
  + Mejorar la estética de la app
  + Falta una actualización en tiempo real.

futuras expansiones

Se piensa añadir la opción de pago para la contratación de profesionales.

* Referencia de nuevos tipos definidos.

Ninguno por el momento.

* Referencia de constantes y variables globales.
  + Constantes
    - User String: nombre de la cuenta de usuario
    - Link String: enlace de la pagina para acceder a la base de datos.
  + Variables globales
    - Ninguna
* Referencia de procedimientos y funciones

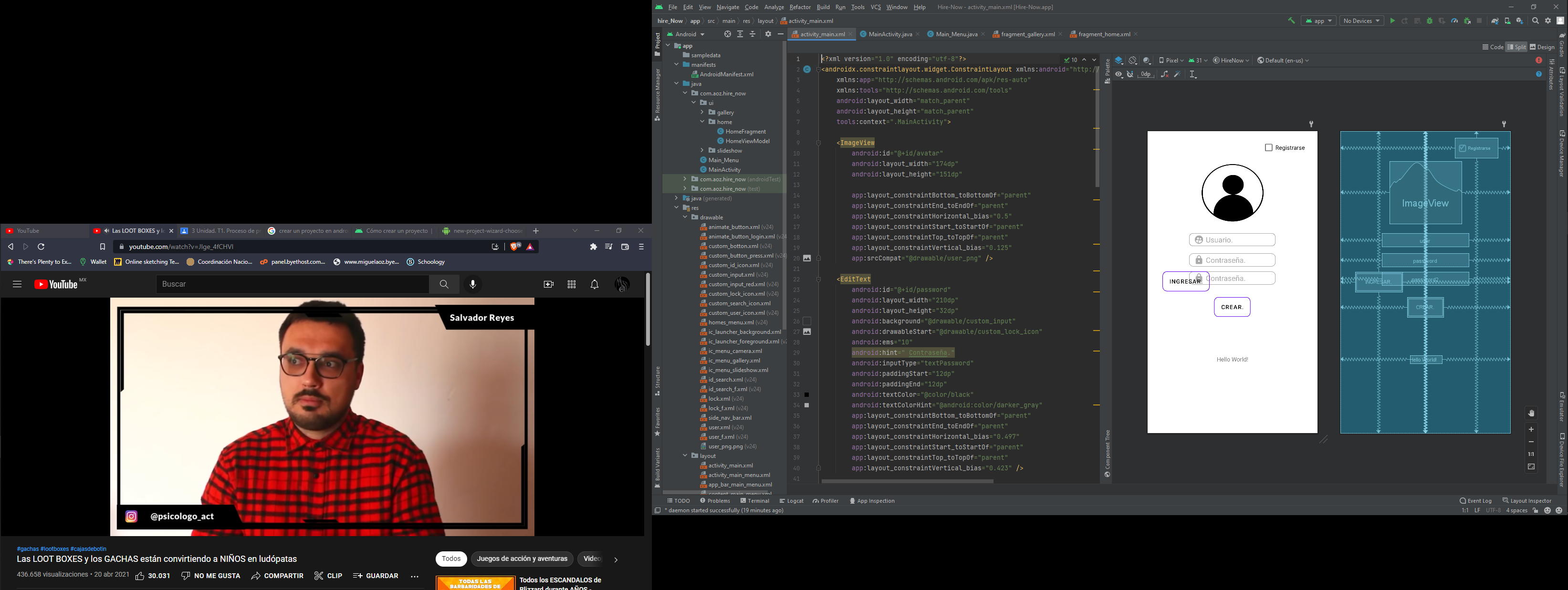
Rutina: abre la ventana home,

Función: muestra las ofertas disponibles y la posibilidad de crear una si el usuario esta verificado.

Parámetros: RecicleView muestra los datos en lista.

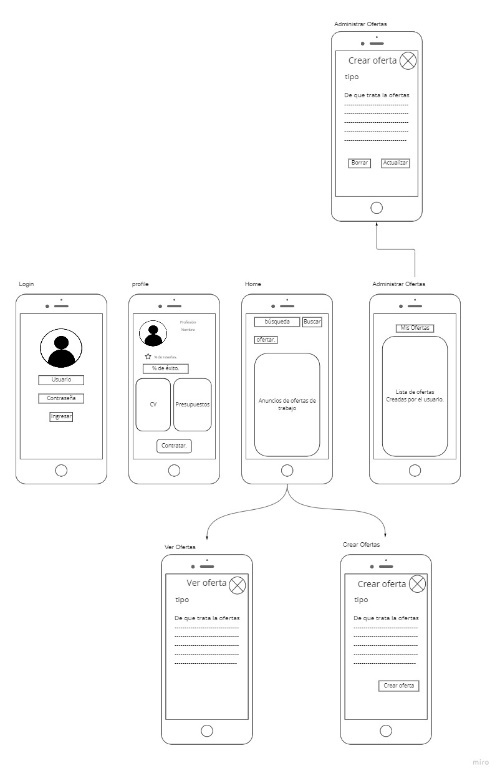
* Apéndices

A-1.Formato de los archivos y pantallas.



|  |  |  |
| --- | --- | --- |
| campo | tipo | tamaño |
| usuario | String | 255 |
| contraseña | String | 255 |
| Comprueba contraseña | String | 255 |

A-2.Diagramas.



A-3.Listado del programa fuente.

A-4 Características Técnicas

* + Celular Android de gamma baja
  + Sistema operativo Android 8
  + 10 Mb de espacio 100 Mb de ram.
  + Espacio del APK 5,233 Kb
  + Ejecuta el APK y usa.