UNIVERSIDAD LATINA DE COSTA RICA
POWERED BY Arizona State University

Manual Técnico

Curso:

BIS 14

Programación V

Profesor: Walter Guzmán Sánchez

Integrantes:

Marco Ney Fuertes Murillo

Kimberly Garcia Obando

Saquille Lackwood Blanco

Alvaro Montes de Oca Ramírez

Britany Morales Delgado

Xavier Salazar Mora

Steven Sibaja Araya

Evelyn Siezar Cambronero

II Cuatrimestre

Año 2022

Introducción

La siguiente aplicación está orientada al área estudiantil respecto a las becas de comedor estudiantiles, con la finalidad de hacer las funciones y registrar el uso de la beca diaria o mensualmente, el sistema ayuda a tener un mejor orden, ya sea a la hora de registrar el ingreso de los administradores o también de los estudiantes, por otra parte, podrá modificar su usuario y contraseña, también se podrá mostrar un historial donde dirá el día y la hora en que realizó uso de su beca.

Este sistema es de mucha ayuda para los centros educativos, ya que, permite un mejor control y ahorra mucho tiempo, cualquier centro educativo podría usar esta aplicación y es muy sencilla de usar.

Objetivo general

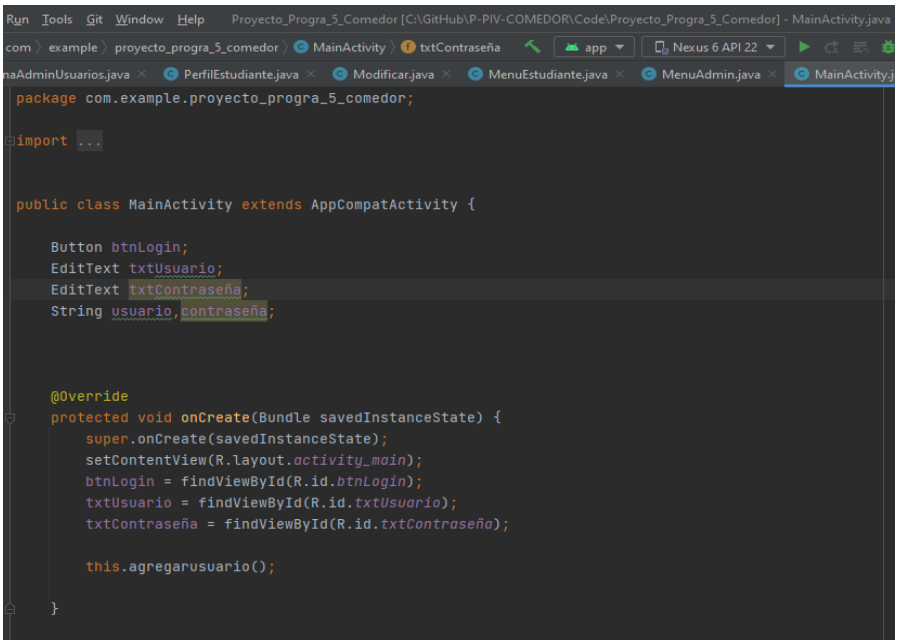
Determinar la asistencia de cada uno de los estudiantes del centro educativo la beca y registrar el uso diario o mensualmente, buscando fluidez y un mejor orden a la hora de ir al comedor.

Objetivos Específicos

- Facilitar el servicio de confirmación de beca de cada estudiante.
- Llevar un mejor control de beca de comedor en los estudiantes.
- Optimizar las funciones que se realizan en el centro educativo.

MainActivity

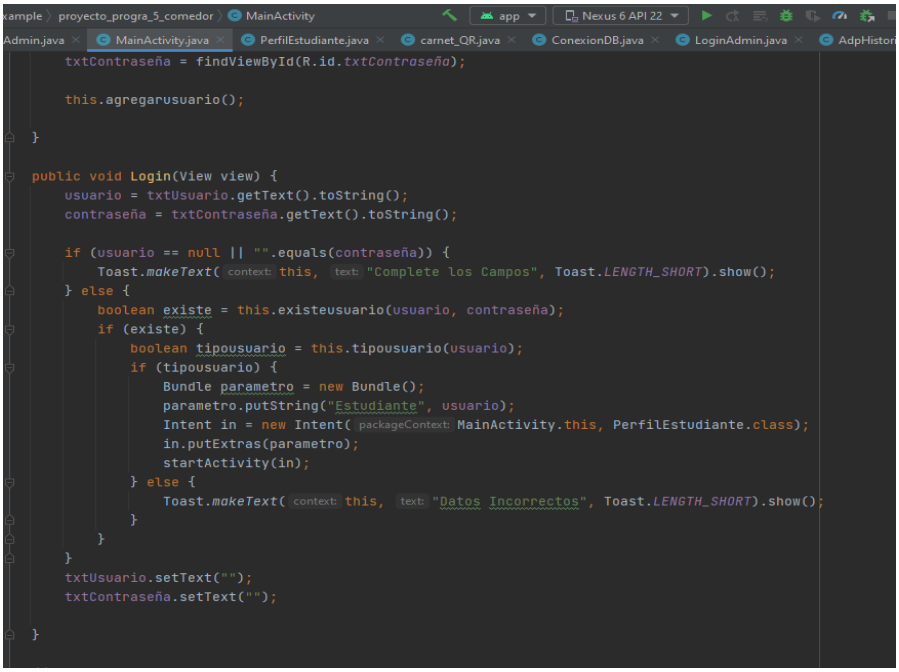
Primeramente, se crean las variables que seria Button btnLogin, un EditText txtusuario, EditText txtContraseña y un String usuario y contrasela.



```
package com.example.proyecto_progra_5_comedor;

import ...


public class MainActivity extends AppCompatActivity {

    Button btnLogin;
    EditText txtUsuario;
    EditText txtContraseña;
    String usuario,contraseña;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnLogin = findViewById(R.id.btnLogin);
        txtUsuario = findViewById(R.id.txtUsuario);
        txtContraseña = findViewById(R.id.txtContraseña);

        this.agregarusuario();

    }
```

Se mandan a llamar las vistas

Se crea un activity On click de Login, Se mandan a llamar las variables, Se crea un if donde el usuario es diferente a null va tener que llenar los campos, de lo contrario si existe va a hacer el tipo de usuario A que seria el estudiante, luego si el usuario manda a llamar los datos que están en la tabla de usuarios para confirmar los datos, Luego se crea un Intent de la pantalla Login admin a menú admin y manda a llamar los datos del usuario, y de lo contrario seria que los datos están incorrectos.

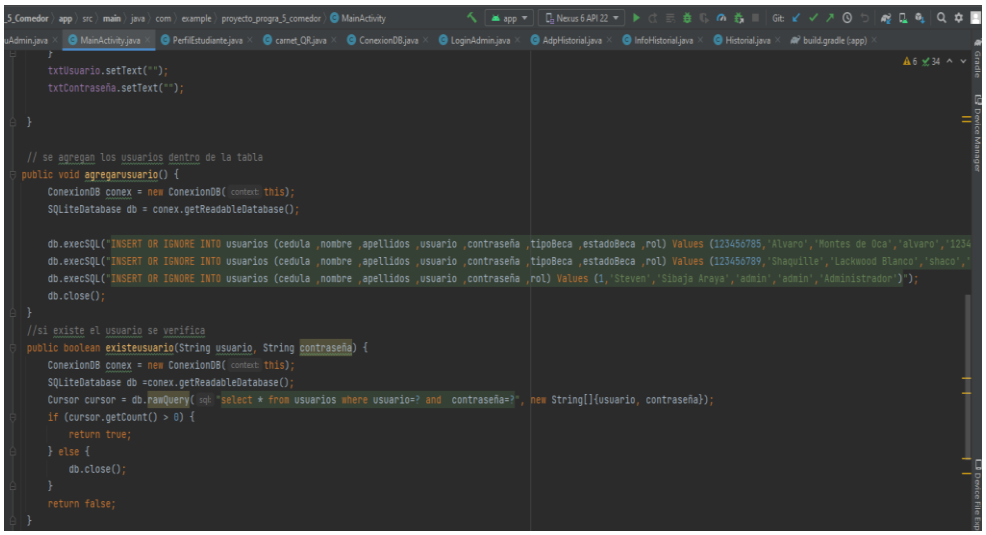


```
    txtContraseña = findViewById(R.id.txtContraseña);

    this.agregarusuario();

}

public void Login(View view) {
    usuario = txtUsuario.getText().toString();
    contraseña = txtContraseña.getText().toString();

    if (usuario == null || "".equals(contraseña)) {
        Toast.makeText( context: this,  text: "Complete los Campos", Toast.LENGTH_SHORT).show();
    } else {
        boolean existe = this.existeusuario(usuario, contraseña);
        if (existe) {
            boolean tipousuario = this.tipousuario(usuario);
            if (tipousuario) {
                Bundle parametro = new Bundle();
                parametro.putString("Estudiante", usuario);
                Intent in = new Intent( packageContext: MainActivity.this, PerfilEstudiante.class);
                in.putExtras(parametro);
                startActivity(in);
            } else {
                Toast.makeText( context: this,  text: "Datos Incorrectos", Toast.LENGTH_SHORT).show();
            }
        }
    }
    txtUsuario.setText("");
    txtContraseña.setText("");

}
```

Acá se mandan a limpian los datos

Se crean los métodos de agregar usuario y existe usuario que es para confirmar la contraseña, se manda a llamar la base de datos para que lean los datos que están en la tabla de usuario.

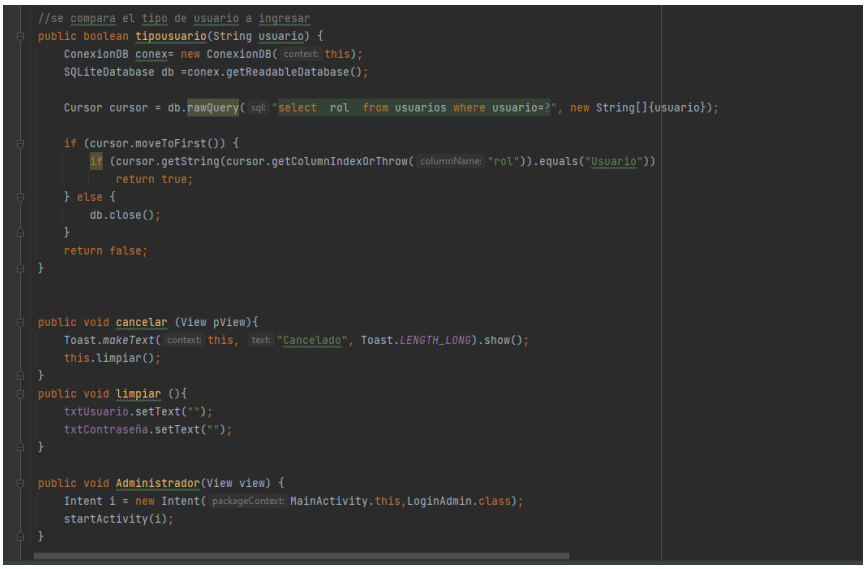
Con un cursor se selecciona el usuario y la contraseña para se validados



```
        txtUsuario.setText("");
        txtContraseña.setText("");

    }

    // se agregan los usuarios dentro de la tabla
    public void agregarusuario() {
        ConexionDB conex = new ConexionDB( context: this);
        SQLiteDatabase db = conex.getReadableDatabase();

        db.execSQL("INSERT OR IGNORE INTO usuarios (cedula ,nombre ,apellidos ,usuario ,contraseña ,tipoBeca ,estadoBeca ,rol) Values (123456785,'Alvaro','Montes de Oca','alvaro','1234
        db.execSQL("INSERT OR IGNORE INTO usuarios (cedula ,nombre ,apellidos ,usuario ,contraseña ,tipoBeca ,estadoBeca ,rol) Values (123456789,'Shaquille','Lackwood Blanco','shaco',
        db.execSQL("INSERT OR IGNORE INTO usuarios (cedula ,nombre ,apellidos ,usuario ,contraseña ,rol) Values (1,'Steven','Sibaja Araya','admin','admin','Administrador')");
        db.close();
    }
    //si existe el usuario se verifica
    public boolean existeusuario(String usuario, String contraseña) {
        ConexionDB conex = new ConexionDB( context: this);
        SQLiteDatabase db =conex.getReadableDatabase();
        Cursor cursor = db.rawQuery( sql: "select * from usuarios where usuario=? and  contraseña=?", new String[]{usuario, contraseña});
        if (cursor.getCount() > 0) {
            return true;
        } else {
            db.close();
        }
        return false;
    }
```

Se crea el método tipousuario, si es estudiante o administrador manda a llamar a la base de datos , trata de leer los datos que están en la tabla, luego selecciona la pila de rol en el que va a estar el administrador.

Luego el método cancelar que limpia los campos y limpiar para el método cancelar.



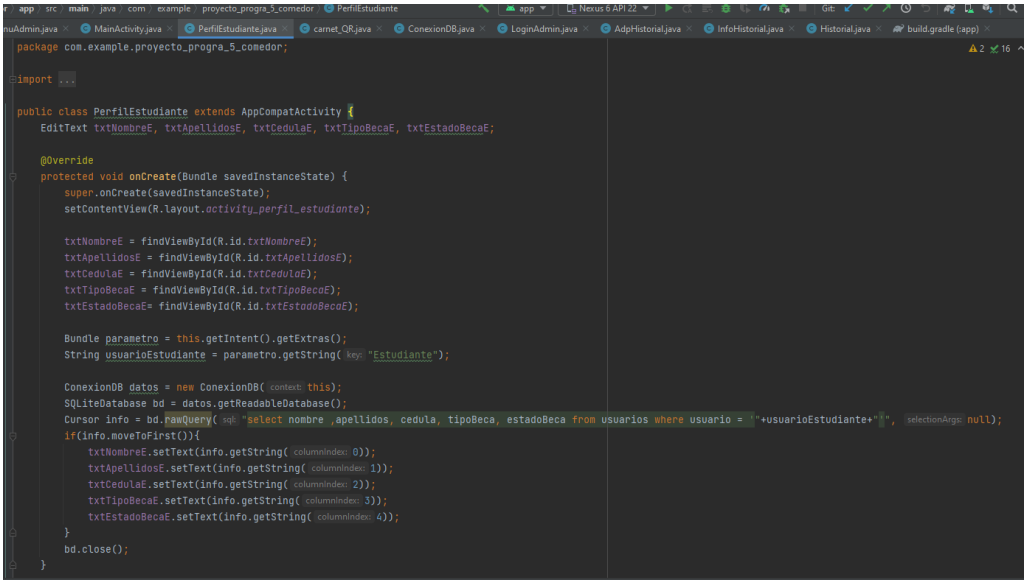
```
    //se compara el tipo de usuario a ingresar
    public boolean tipousuario(String usuario) {
        ConexionDB conex= new ConexionDB( context: this);
        SQLiteDatabase db =conex.getReadableDatabase();

        Cursor cursor = db.rawQuery( sql: "select  rol  from usuarios where usuario=?", new String[]{usuario});

        if (cursor.moveToFirst()) {
            if (cursor.getString(cursor.getColumnIndexOrThrow( columnName: "rol")).equals("Usuario"))
                return true;
        } else {
            db.close();
        }
        return false;
    }


    public void cancelar (View pView){
        Toast.makeText( context: this,  text: "Cancelado", Toast.LENGTH_LONG).show();
        this.limpiar();
    }
    public void limpiar (){
        txtUsuario.setText("");
        txtContraseña.setText("");
    }

    public void Administrador(View view) {
        Intent i = new Intent( packageContext: MainActivity.this,LoginAdmin.class);
        startActivity(i);
    }
```

Perfil estudiante

Primeramente, se establecen las variables de tipo EditText, seguidamente se mandan a llamar las vistas con el método de findViewById, luego se introduce un bundle para que mande a llamar los datos de estudiantes, por otra parte, se manda a llamar la base de datos para que lea la tabla de usuarios, para que mande a imprimir los datos.

Luego se hace un if para mandar a llamar los datos en ese orden.

Por último, se cierra la base de datos.



```java
package com.example.proyecto_progra_5_comedor;

import ...

public class PerfilEstudiante extends AppCompatActivity {
    EditText txtNombreE, txtApellidosE, txtCedulaE, txtTipoBecaE, txtEstadoBecaE;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_perfil_estudiante);

        txtNombreE = findViewById(R.id.txtNombreE);
        txtApellidosE = findViewById(R.id.txtApellidosE);
        txtCedulaE = findViewById(R.id.txtCedulaE);
        txtTipoBecaE = findViewById(R.id.txtTipoBecaE);
        txtEstadoBecaE= findViewById(R.id.txtEstadoBecaE);

        Bundle parametro = this.getIntent().getExtras();
        String usuarioEstudiante = parametro.getString( key: "Estudiante");

        ConexionDB datos = new ConexionDB( context: this);
        SQLiteDatabase bd = datos.getReadableDatabase();
        Cursor info = bd.rawQuery( sql: "select nombre ,apellidos, cedula, tipoBeca, estadoBeca from usuarios where usuario = '"+usuarioEstudiante+"'", selectionArgs: null);
        if(info.moveToFirst()){
            txtNombreE.setText(info.getString( columnIndex: 0));
            txtApellidosE.setText(info.getString( columnIndex: 1));
            txtCedulaE.setText(info.getString( columnIndex: 2));
            txtTipoBecaE.setText(info.getString( columnIndex: 3));
            txtEstadoBecaE.setText(info.getString( columnIndex: 4));
        }
        bd.close();
    }
```

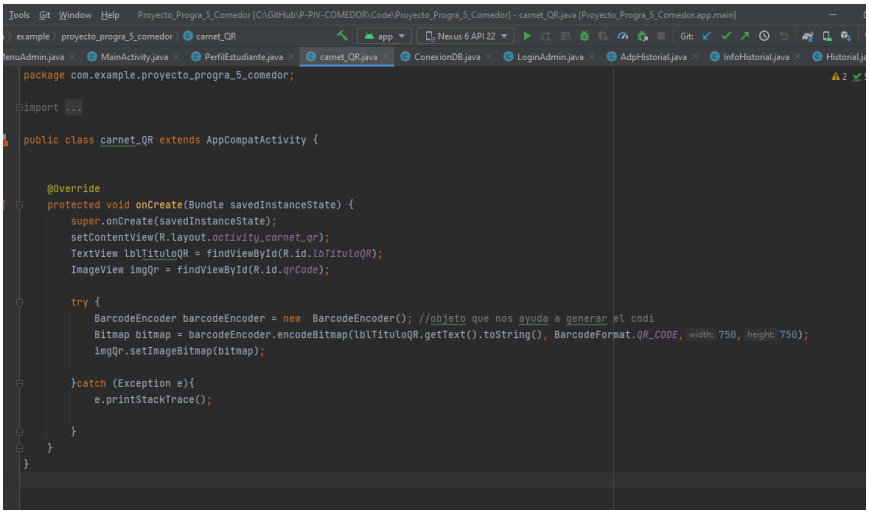
Se crea un método OnClick que se llama QR, donde manda a llamar a la ventana de carnet_QR.

```java
public void consultaEstudiante(View view) {
}

public void QR(View view) {
    Intent venQR = new Intent( packageContext: PerfilEstudiante.this,carnet_QR.class);
    startActivity(venQR);
}
}
```

Carnet QR

Acá solamente se hace el código para mandar a llamar el código_QR, se llama la vista del titulo y también se manda a llamar la vista del QR.

Se crea un try catch, para que pueda leer el QR y así pueda seguir probando si no lo quiere reconocer.

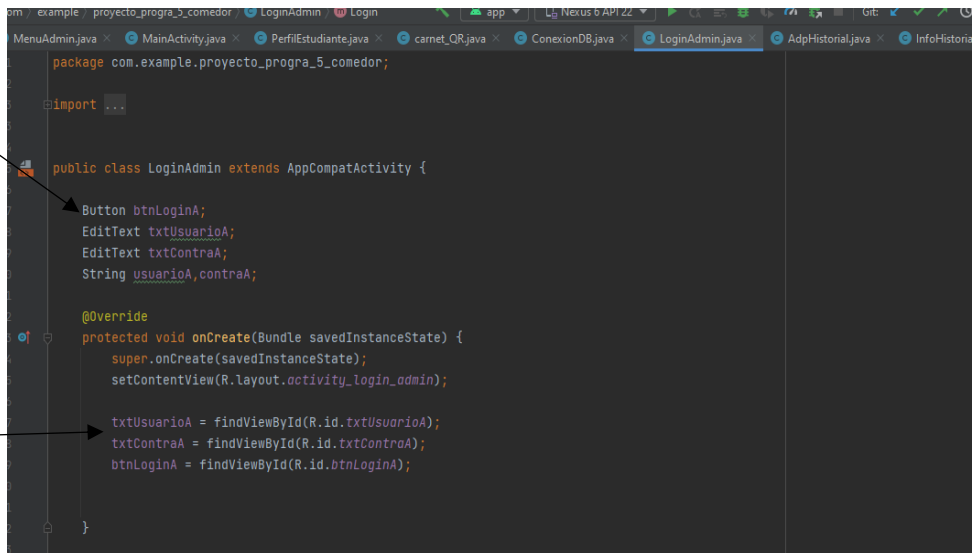


```
package com.example.proyecto_progra_5_comedor;

import ...

public class carnet_QR extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_carnet_qr);
        TextView lblTituloQR = findViewById(R.id.lbTituloQR);
        ImageView imgQr = findViewById(R.id.qrCode);

        try {
            BarcodeEncoder barcodeEncoder = new  BarcodeEncoder(); //objeto que nos ayuda a generar el codi
            Bitmap bitmap = barcodeEncoder.encodeBitmap(lblTituloQR.getText().toString(), BarcodeFormat.QR_CODE, width: 750, height: 750);
            imgQr.setImageBitmap(bitmap);

        }catch (Exception e){
            e.printStackTrace();

        }
    }
}
```

Login Admin

Primeramente, creamos las variables en este caso serían un Button btnLoginA, dos EditText txtUsuarioA, txtContraA y dos String usuarioA y contra.

Luego se manda a llamar las vistas.



```
package com.example.proyecto_progra_5_comedor;

import ...


public class LoginAdmin extends AppCompatActivity {

    Button btnLoginA;
    EditText txtUsuarioA;
    EditText txtContraA;
    String usuarioA,contraA;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login_admin);

        txtUsuarioA = findViewById(R.id.txtUsuarioA);
        txtContraA = findViewById(R.id.txtContraA);
        btnLoginA = findViewById(R.id.btnLoginA);


    }
```

Se hace un Activity OnClick del Login para cuando presione el botón de ingresar o de Login, se mandan a llamar las variables de getText a String.

Luego se hace un if en el que, si usuario es diferente a null o los espacios en blanco es igual a contraseña, va a tener que llenar los campos, de lo contrario si existe, entonces va a ser el tipo de usuarioA, que sería el estudiante.

```
public void Login(View view) {
    usuarioA = txtUsuarioA.getText().toString();
    contraA = txtContraA.getText().toString();

    if (usuarioA == null || "".equals(contraA)) {
        Toast.makeText( context: this,  text: "Complete los Campos", Toast.LENGTH_SHORT).show();
    } else {
        boolean existe = this.existeusuario(usuarioA, contraA);
        if (existe) {
            boolean tipousuario = this.tipousuario(usuarioA);
            if (tipousuario) {
                Bundle parametro = new Bundle();
                parametro.putString("Estudiante", usuarioA);
                Intent i = new Intent( packageContext: LoginAdmin.this, MenuAdmin.class);
                i.putExtras(parametro);
                startActivity(i);

            } else {
                Toast.makeText( context: this,  text: "Datos Incorrectos", Toast.LENGTH_SHORT).show();
            }
        }
        txtUsuarioA.setText("");
        txtContraA.setText("");
    }
}
```

Después se manda a llamar un bundle de parámetro que manda a llamar los datos que están en la tabla de usuarios para confirmar los datos, después se hace un intent para que mande a la pantalla de loginAdmin a menú admin, si no, les mostrará un mensaje donde dice que los datos son incorrectos.

Luego se crea el método booleano de existeusuario, que básicamente sería para confirmar el usuario y la contraseña. Se manda a llamar la Base de Datos, para que lea las variables que están en la tabla de usuarios y con un cursor se selecciona el usuario y la contraseña, para que los validen.

```
//si existe el usuario se verifica
public boolean existeusuario(String usuario, String contraseña) {
    ConexionDB conex = new ConexionDB( context: this);
    SQLiteDatabase db =conex.getReadableDatabase();
    Cursor cursor = db.rawQuery( sql: "select * from usuarios where usuario=? and  contraseña=?", new String[]{usuario, contraseña});
    if (cursor.getCount() > 0) {
        return true;
    } else {
        db.close();
    }
    return false;
}
//se compara el tipo de usuario a ingresar
public boolean tipousuario(String usuario) {
    ConexionDB conex = new ConexionDB( context: this);
    SQLiteDatabase db =conex.getReadableDatabase();

    Cursor cursor = db.rawQuery( sql: "select  rol  from usuarios where usuario=?", new String[]{usuario});

    if (cursor.moveToFirst()) {
        if (cursor.getString(cursor.getColumnIndexOrThrow( columnName: "rol")).equals("Administrador"))
            return true;
    } else {
        db.close();
    }
    return false;
}
```

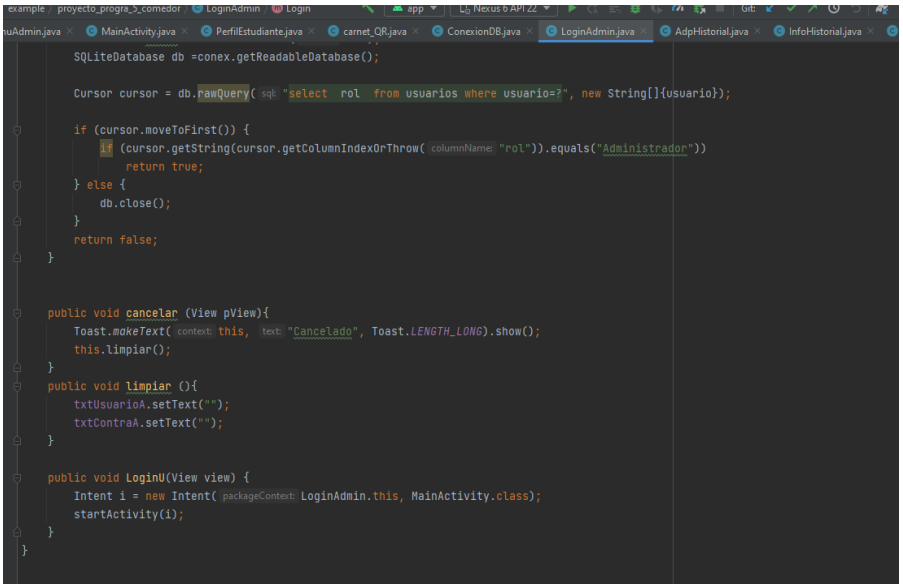
Por otra parte, tenemos el método tipo usuario, que valida si es Administrador o es un Estudiante.

El cursor selecciona la tabla de rol donde va a estar el administrador.

Tenemos un método cancelar que es para limpiar los campos

También un método limpiar que es básicamente lo mismo que el de cancelar, donde se manda a llamar el método cancelar.

El método LoginU es para mandar a ingresar al menú de estudiantes

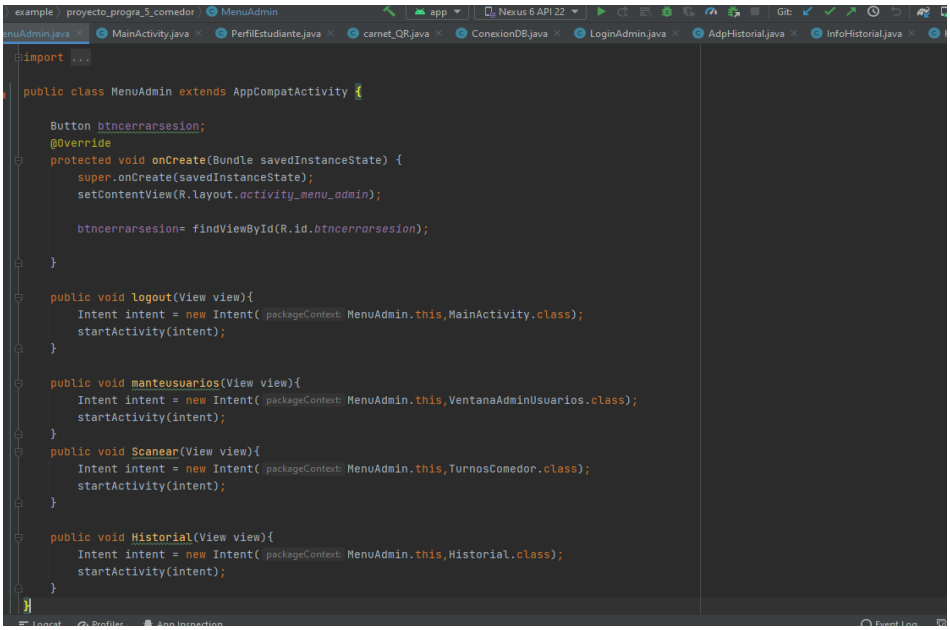


```
        SQLiteDatabase db =conex.getReadableDatabase();

        Cursor cursor = db.rawQuery( sql: "select  rol  from usuarios where usuario=?", new String[]{usuario});

        if (cursor.moveToFirst()) {
            if (cursor.getString(cursor.getColumnIndexOrThrow( columnName: "rol")).equals("Administrador"))
                return true;
        } else {
            db.close();
        }
        return false;
    }


    public void cancelar (View pView){
        Toast.makeText( context: this,  text: "Cancelado", Toast.LENGTH_LONG).show();
        this.limpiar();
    }
    public void limpiar (){
        txtUsuarioA.setText("");
        txtContraA.setText("");
    }

    public void LoginU(View view) {
        Intent i = new Intent( packageContext: LoginAdmin.this, MainActivity.class);
        startActivity(i);
    }
}
```

Menu Admin

Primero se crea la variable Button, luego se manda a llamar la vista.

Por último, se crean los OnClick para que mande a las otras ventanas, el logout sirve para que lo mande al MainActivity, el manteusuario lo manda a la ventanaAdminUsuario, el scanear lo manda a TurnosComedor y por último el historial lo redirige a la pantalla historial.



```
import ...

public class MenuAdmin extends AppCompatActivity {

    Button btncerrarsesion;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu_admin);

        btncerrarsesion= findViewById(R.id.btncerrarsesion);

    }

    public void logout(View view){
        Intent intent = new Intent( packageContext: MenuAdmin.this,MainActivity.class);
        startActivity(intent);
    }

    public void manteusuarios(View view){
        Intent intent = new Intent( packageContext: MenuAdmin.this,VentanaAdminUsuarios.class);
        startActivity(intent);
    }
    public void Scanear(View view){
        Intent intent = new Intent( packageContext: MenuAdmin.this,TurnosComedor.class);
        startActivity(intent);
    }

    public void Historial(View view){
        Intent intent = new Intent( packageContext: MenuAdmin.this,Historial.class);
        startActivity(intent);
    }
}
```

Modificar

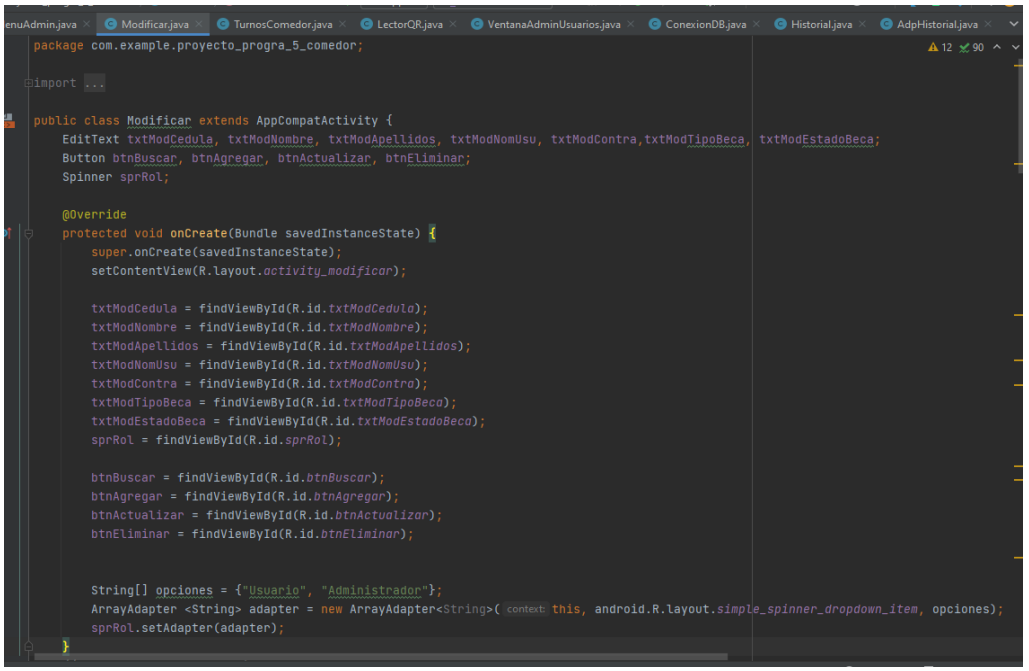
Se crea las variables EditText txtModCedula, txtModNombre, txtModApellidos, txtModNomUsu, txtModContra, txtModTipoBeca y txtModEstadoBeca.

Tipo Button btnBuscar, btnAgregar, btnActualizar, btnEliminar.

Luego una tipo Spinner sprRol.

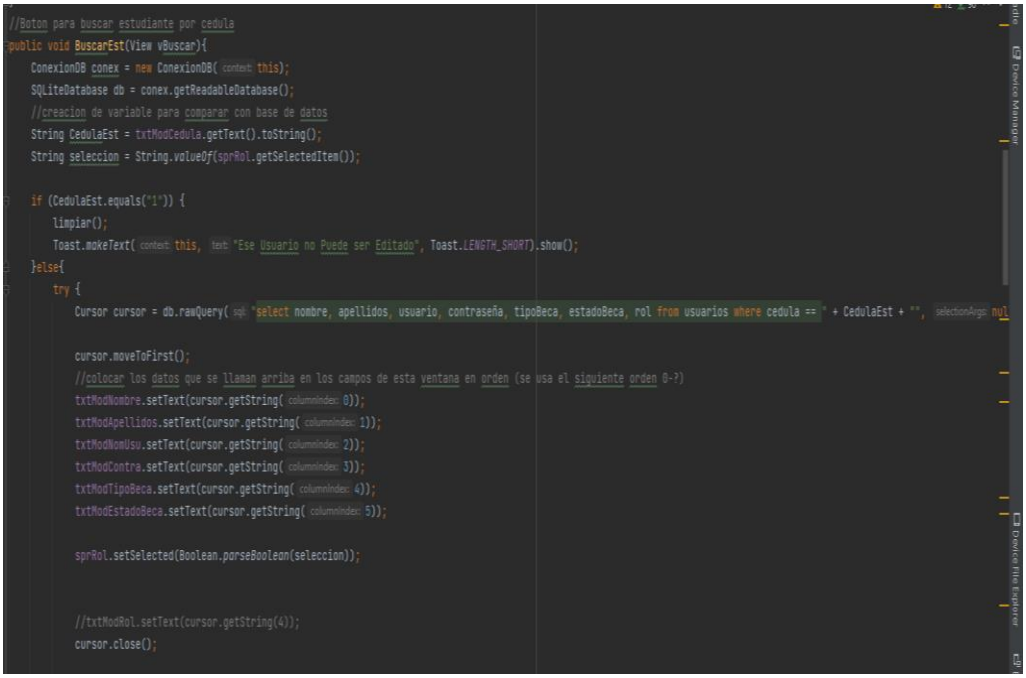
Después se mandan a llamar las vistas

Se crea un Array opciones en caso de sea usuario o administrador



```java
package com.example.proyecto_progra_5_comedor;

import ...

public class Modificar extends AppCompatActivity {
    EditText txtModCedula, txtModNombre, txtModApellidos, txtModNomUsu, txtModContra,txtModTipoBeca, txtModEstadoBeca;
    Button btnBuscar, btnAgregar, btnActualizar, btnEliminar;
    Spinner sprRol;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_modificar);

        txtModCedula = findViewById(R.id.txtModCedula);
        txtModNombre = findViewById(R.id.txtModNombre);
        txtModApellidos = findViewById(R.id.txtModApellidos);
        txtModNomUsu = findViewById(R.id.txtModNomUsu);
        txtModContra = findViewById(R.id.txtModContra);
        txtModTipoBeca = findViewById(R.id.txtModTipoBeca);
        txtModEstadoBeca = findViewById(R.id.txtModEstadoBeca);
        sprRol = findViewById(R.id.sprRol);

        btnBuscar = findViewById(R.id.btnBuscar);
        btnAgregar = findViewById(R.id.btnAgregar);
        btnActualizar = findViewById(R.id.btnActualizar);
        btnEliminar = findViewById(R.id.btnEliminar);


        String[] opciones = {"Usuario", "Administrador"};
        ArrayAdapter <String> adapter = new ArrayAdapter<String>( context: this, android.R.layout.simple_spinner_dropdown_item, opciones);
        sprRol.setAdapter(adapter);
    }
```

Se crea el método buscar estudiante para buscar el estudiante en la base de datos para que se lea en la tabla. Se crea un if que diga la cedula del estudiante tiene que ser igual a uno. En el else sea un cath para que mande a llamar los datos de la tabla a partir de la cedula



```java
//Boton para buscar estudiante por cedula
public void BuscarEst(View vBuscar){
    ConexionDB conex = new ConexionDB( context: this);
    SQLiteDatabase db = conex.getReadableDatabase();
    //creacion de variable para comparar con base de datos
    String CedulaEst = txtModCedula.getText().toString();
    String seleccion = String.valueOf(sprRol.getSelectedItem());

    if (CedulaEst.equals("1")) {
        limpiar();
        Toast.makeText( context: this,  text: "Ese Usuario no Puede ser Editado", Toast.LENGTH_SHORT).show();
    }else{
        try {
            Cursor cursor = db.rawQuery( sql: "select nombre, apellidos, usuario, contraseña, tipoBeca, estadoBeca, rol from usuarios where cedula == " + CedulaEst + "",  selectionArgs: nul

            cursor.moveToFirst();
            //colocar los datos que se llaman arriba en los campos de esta ventana en orden (se usa el siguiente orden 0-?)
            txtModNombre.setText(cursor.getString( columnIndex: 0));
            txtModApellidos.setText(cursor.getString( columnIndex: 1));
            txtModNomUsu.setText(cursor.getString( columnIndex: 2));
            txtModContra.setText(cursor.getString( columnIndex: 3));
            txtModTipoBeca.setText(cursor.getString( columnIndex: 4));
            txtModEstadoBeca.setText(cursor.getString( columnIndex: 5));

            sprRol.setSelected(Boolean.parseBoolean(seleccion));


            //txtModRol.setText(cursor.getString(4));
            cursor.close();
```

Se crea el catch para que notifique si el usuario existe o no

El método existe es para verificar si el usuario existe se llama a mandar a la base de datos para que se lea los datos y Seleccione la cedula.

Se crea para que lo que se digite en los campos de texto se añadan y se van a obtener con getText a toSting

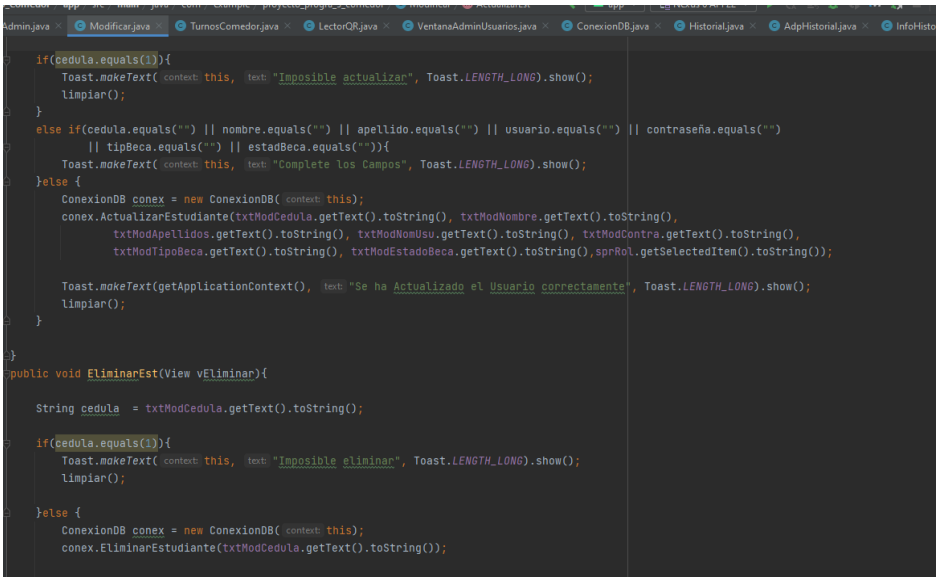
```
MenuAdmin.java | Modificar.java | TurnosComedor.java | LectorQR.java | VentanaAdminUsuarios.java | ConexionDB.java | Historial.java

            Toast.makeText( context: this, text: "Se ha Cargado el Usuario correctamente", Toast.LENGTH_LONG).show();
        }catch (Exception e){
            Toast.makeText(getApplicationContext(), text: "El Usuario no existe", Toast.LENGTH_LONG).show();
            limpiar();
        }
    }
}
private boolean existe(String cedula){
    ConexionDB conex =new ConexionDB( context: this);
    SQLiteDatabase db =conex.getReadableDatabase();
    Cursor cursor = db.rawQuery( sql: "select * from usuarios where cedula=?", new String[]{String.valueOf(cedula)});
    if (cursor.getCount() > 0) {
        return true;
    } else {
        db.close();
        return false;
    }
}
public void AgregarEst(View vAgregar){
    String CedulaEst = txtModCedula.getText().toString();
    String nombre  = txtModNombre.getText().toString();
    String apellido  = txtModApellidos.getText().toString();
    String usuario  = txtModNomUsu.getText().toString();
    String contraseña  = txtModContra.getText().toString();
    String tipBeca = txtModTipoBeca.getText().toString();
    String estadBeca = txtModEstadoBeca.getText().toString();
```

Se crea un if donde si existe la cedula del estudiante se envía un mensaje que ya existe y limpia los campos de lo contrario cedula es igual a 1 entonces no se puede ingresar y si los campos están en blanco se envía un mensaje de que completo los campos, luego se llama la conexión para agregar los datos de los estudiantes a la tabla

```
    if (existe(CedulaEst)) {
        Toast.makeText(getApplicationContext(), text: "El Usuario ya existe", Toast.LENGTH_LONG).show();
        limpiar();
    }else if(CedulaEst.equals(1)){
        Toast.makeText( context: this, text: "Imposible Agregar", Toast.LENGTH_LONG).show();
        limpiar();
    }
    else if(CedulaEst.equals("") || nombre.equals("") || apellido.equals("") || usuario.equals("") || contraseña.equals("")
            || tipBeca.equals("") || estadBeca.equals("")){
        Toast.makeText( context: this, text: "Complete los Campos", Toast.LENGTH_LONG).show();
    }else{
        ConexionDB conex = new ConexionDB( context: this);
        conex.AgregarEstudiante(txtModCedula.getText().toString(), txtModNombre.getText().toString(),
                txtModApellidos.getText().toString(), txtModNomUsu.getText().toString(), txtModContra.getText().toString(),
                txtModTipoBeca.getText().toString(), txtModEstadoBeca.getText().toString(),sprRol.getSelectedItem().toString());

        Toast.makeText(getApplicationContext(), text: "Se ha Agregado el Usuario correctamente", Toast.LENGTH_LONG).show();
        limpiar();
    }
}
public void ActualizarEst(View vActualizar){

    String cedula  = txtModCedula.getText().toString();
    String nombre  = txtModNombre.getText().toString();
    String apellido  = txtModApellidos.getText().toString();
    String usuario  = txtModNomUsu.getText().toString();
    String contraseña  = txtModContra.getText().toString();
    String tipBeca = txtModTipoBeca.getText().toString();
    String estadBeca = txtModEstadoBeca.getText().toString();
```

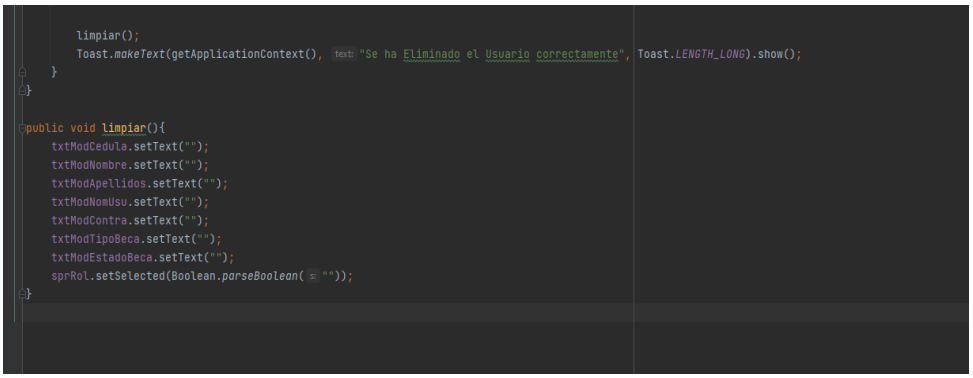
Se crea un método de Actualizar que manda a llamar los datos en los getText si la cedula a toString

Se crea un if donde si existe la cedula del estudiante se envía un mensaje que ya existe y limpia los campos de lo contrario cedula es igual a 1 entonces no se puede ingresar y si los campos están en blanco se envía un mensaje de que completo los campos, luego se llama la conexión para agregar los datos de los estudiantes a la tabla



Se crea el método eliminar

Si la variable cedula es 1 no se puede elimina porque es administrador, y un else que si se elimina la cedula se elimina el estudiante



Se crea el método limpiar el cual limpia los campos

Turno comedor

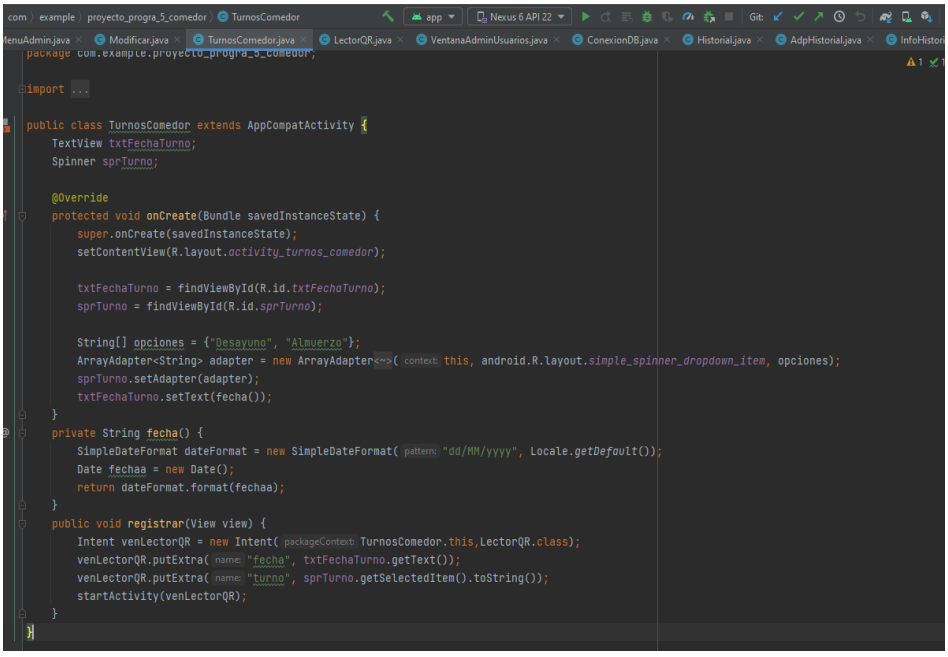
Se crean las variables de tipo TextView y de tipo Spinner.

Luego se mandan a llamar las vistas.

Se crea un array de opciones donde se encuentra la opción desayuno o almuerzo, para la hora de seleccionar una de las dos opciones, simplemente tome el valor que seleccionó.

Se establece la variable para establecer la fecha.

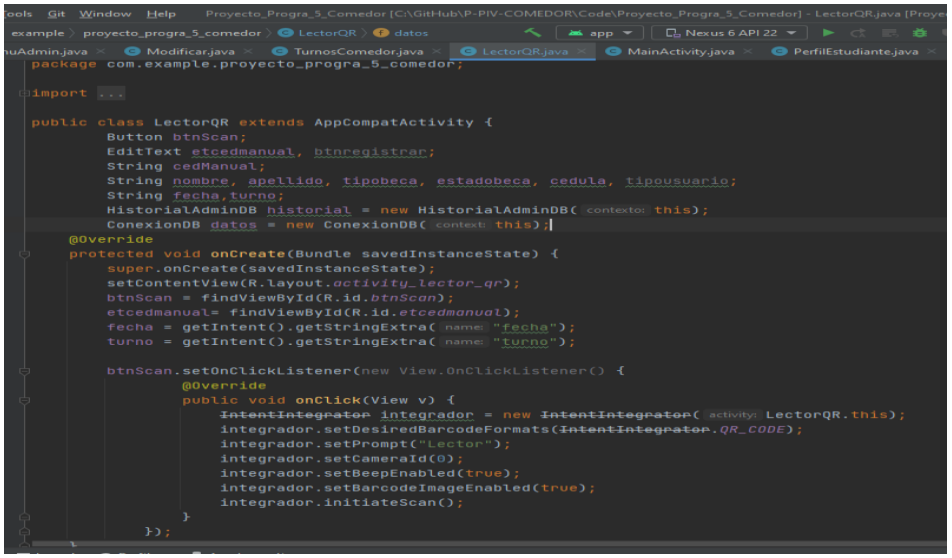
Y por ultimo se crea el método de registrar, para que quede guardado en el historial.



```java
package com.example.proyecto_progra_5_comedor;

import ...

public class TurnosComedor extends AppCompatActivity {
    TextView txtFechaTurno;
    Spinner sprTurno;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_turnos_comedor);

        txtFechaTurno = findViewById(R.id.txtFechaTurno);
        sprTurno = findViewById(R.id.sprTurno);

        String[] opciones = {"Desayuno", "Almuerzo"};
        ArrayAdapter<String> adapter = new ArrayAdapter<~>( context: this, android.R.layout.simple_spinner_dropdown_item, opciones);
        sprTurno.setAdapter(adapter);
        txtFechaTurno.setText(fecha());
    }
    private String fecha() {
        SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "dd/MM/yyyy", Locale.getDefault());
        Date fechaa = new Date();
        return dateFormat.format(fechaa);
    }
    public void registrar(View view) {
        Intent venLectorQR = new Intent( packageContext: TurnosComedor.this,LectorQR.class);
        venLectorQR.putExtra( name: "fecha", txtFechaTurno.getText());
        venLectorQR.putExtra( name: "turno", sprTurno.getSelectedItem().toString());
        startActivity(venLectorQR);
    }
}
```

Lector QR

Se establecen las variables y las vistas

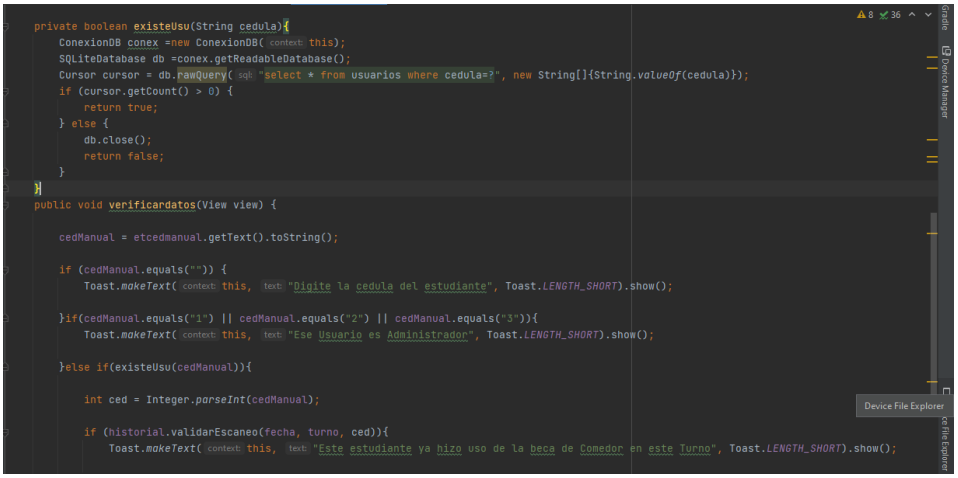


```
package com.example.proyecto_progra_5_comedor;

import ...

public class LectorQR extends AppCompatActivity {
        Button btnScan;
        EditText etcedmanual, btnregistrar;
        String cedManual;
        String nombre, apellido, tipobeca, estadobeca, cedula, tipousuario;
        String fecha,turno;
        HistorialAdminDB historial = new HistorialAdminDB( contexto: this);
        ConexionDB datos = new ConexionDB( context: this);
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lector_qr);
        btnScan = findViewById(R.id.btnScan);
        etcedmanual= findViewById(R.id.etcedmanual);
        fecha = getIntent().getStringExtra( name: "fecha");
        turno = getIntent().getStringExtra( name: "turno");

        btnScan.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    IntentIntegrator integrador = new IntentIntegrator( activity: LectorQR.this);
                    integrador.setDesiredBarcodeFormats(IntentIntegrator.QR_CODE);
                    integrador.setPrompt("Lector");
                    integrador.setCameraId(0);
                    integrador.setBeepEnabled(true);
                    integrador.setBarcodeImageEnabled(true);
                    integrador.initiateScan();
                }
            });
```

Se establece un set onClick para el botón de escanear y activar el lector de scanner.

Se crea un boolean porque si existe usuario a partir de la cedula manda a llamar la base de datos que están en la tabla de usuarios para que lea la cedula si ya paso no permite que ingrese la cedula y se verifica los datos.

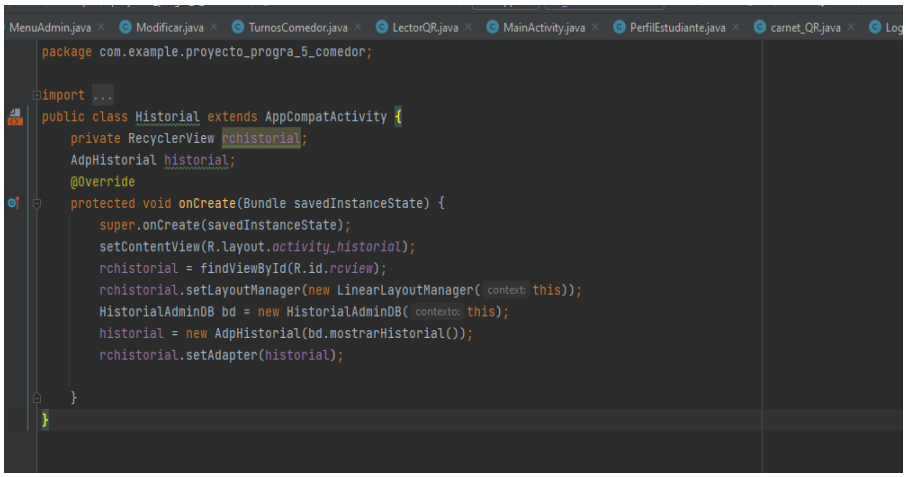
Se crea verificardatos donde si la cedula tiene espacios en blanco tendrá que digitar cedula estudiante, si cedula manual es igual a 1,2 o 3 indica que es administrador, luego si existe el usuario con cedula manual se crea una variable cedula y se manda un if con historial de verificar escaneo, se manda a validar escaneo de los datos del estudiante



```
private boolean existeUsu(String cedula){
    ConexionDB conex =new ConexionDB( context: this);
    SQLiteDatabase db =conex.getReadableDatabase();
    Cursor cursor = db.rawQuery( sql: "select * from usuarios where cedula=?", new String[]{String.valueOf(cedula)});
    if (cursor.getCount() > 0) {
        return true;
    } else {
        db.close();
        return false;
    }
}
public void verificardatos(View view) {

    cedManual = etcedmanual.getText().toString();

    if (cedManual.equals("")) {
        Toast.makeText( context: this, text: "Digite la cedula del estudiante", Toast.LENGTH_SHORT).show();

    }if(cedManual.equals("1") || cedManual.equals("2") || cedManual.equals("3")){
        Toast.makeText( context: this, text: "Ese Usuario es Administrador", Toast.LENGTH_SHORT).show();

    }else if(existeUsu(cedManual)){

        int ced = Integer.parseInt(cedManual);

        if (historial.validarEscaneo(fecha, turno, ced)){
            Toast.makeText( context: this, text: "Este estudiante ya hizo uso de la beca de Comedor en este Turno", Toast.LENGTH_SHORT).show();
```

Historial

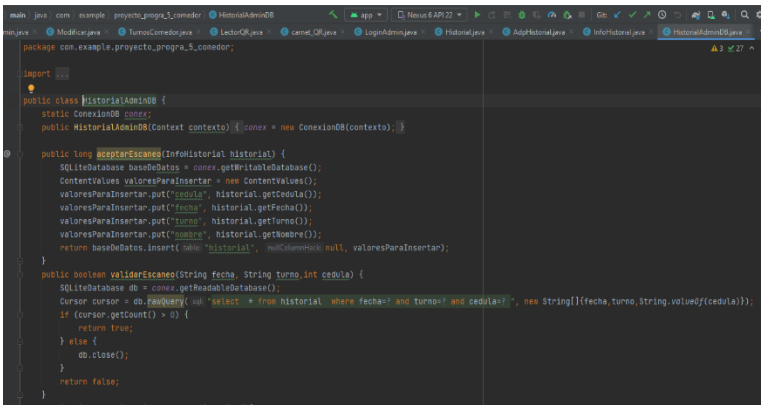
Agarra el rchistorial y lo acomoda en el historial a partir de lo que se llama en el historial admin base de datos que manda a llamar historial que es la lista que guarda todos los datos de los estudiantes.



HistorialAdminDB

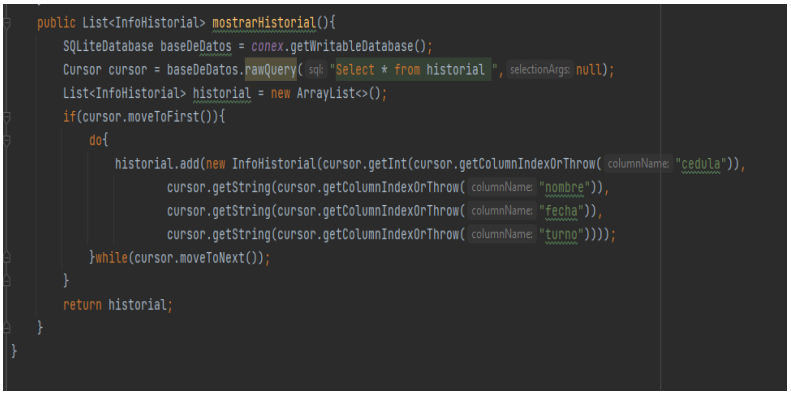
Se mandan a llamar la base de datos para que reescriba los datos del QR

Se validan los datos



```
package con.example.proyecto_progra_5_comedor;

import ...

public class HistorialAdminDB {
    static ConexionDB conex;
    public HistorialAdminDB(Context contexto) { conex = new ConexionDB(contexto); }

    public long aceptarEscaneo(InfoHistorial historial) {
        SQLiteDatabase baseDeDatos = conex.getWritableDatabase();
        ContentValues valoresParaInsertar = new ContentValues();
        valoresParaInsertar.put("cedula", historial.getCedula());
        valoresParaInsertar.put("fecha", historial.getFecha());
        valoresParaInsertar.put("turno", historial.getTurno());
        valoresParaInsertar.put("nombre", historial.getNombre());
        return baseDeDatos.insert( table: "historial", nullColumnHack: null, valoresParaInsertar);
    }
    public boolean validarEscaneo(String fecha, String turno,int cedula) {
        SQLiteDatabase db = conex.getReadableDatabase();
        Cursor cursor = db.rawQuery( sql: "Select  * from historial  where fecha=? and turno=? and cedula=? ", new String[]{fecha,turno,String.valueOf(cedula)});
        if (cursor.getCount() > 0) {
            return true;
        } else {
            db.close();
        }
        return false;
    }
```

Se manda a llamar la base de datos se seleccionan los datos y se acomodan en orden

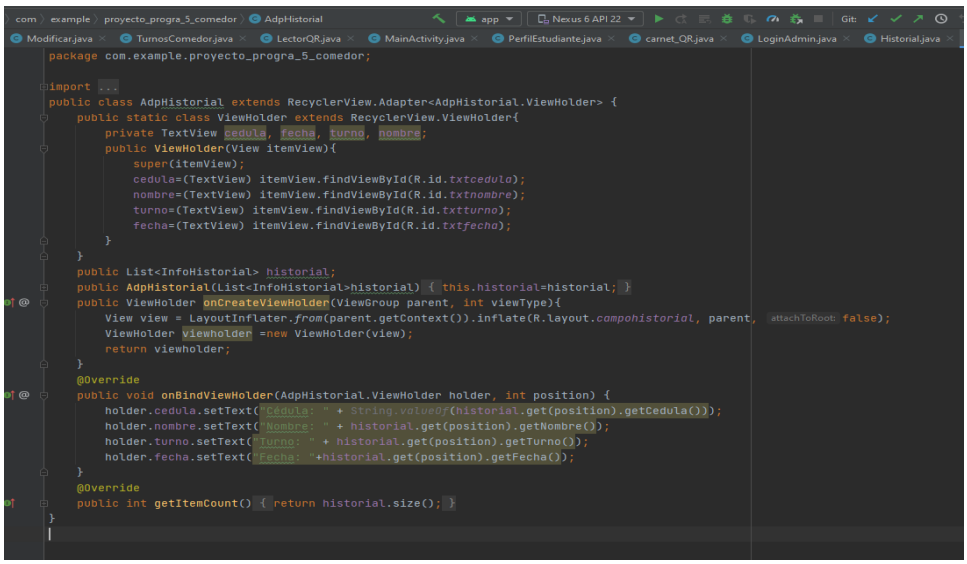


```
public List<InfoHistorial> mostrarHistorial(){
    SQLiteDatabase baseDeDatos = conex.getWritableDatabase();
    Cursor cursor = baseDeDatos.rawQuery( sql: "Select * from historial ", selectionArgs: null);
    List<InfoHistorial> historial = new ArrayList<>();
    if(cursor.moveToFirst()){
        do{
            historial.add(new InfoHistorial(cursor.getInt(cursor.getColumnIndexOrThrow( columnName: "cedula")),
                    cursor.getString(cursor.getColumnIndexOrThrow( columnName: "nombre")),
                    cursor.getString(cursor.getColumnIndexOrThrow( columnName: "fecha")),
                    cursor.getString(cursor.getColumnIndexOrThrow( columnName: "turno"))));
        }while(cursor.moveToNext());
    }
    return historial;
}
}
```

Adphistorial

Se establecen las variables y las vistas

Va a acomodar esos datos de infohistorial en el orden que se establece ahí para llamarlo en la pantalla campohistorial

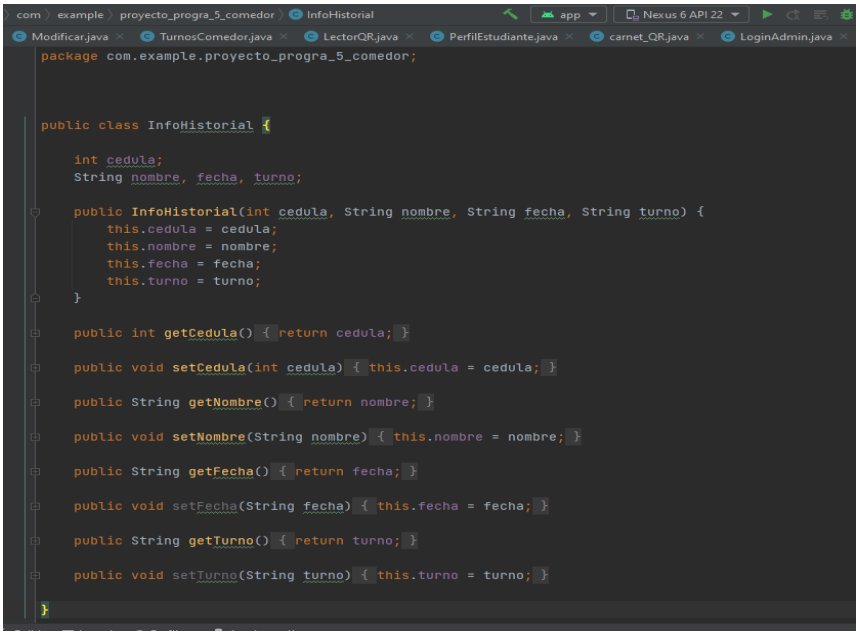


```java
package com.example.proyecto_progra_5_comedor;

import ...
public class AdpHistorial extends RecyclerView.Adapter<AdpHistorial.ViewHolder> {
    public static class ViewHolder extends RecyclerView.ViewHolder{
        private TextView cedula, fecha, turno, nombre;
        public ViewHolder(View itemView){
            super(itemView);
            cedula=(TextView) itemView.findViewById(R.id.txtcedula);
            nombre=(TextView) itemView.findViewById(R.id.txtnombre);
            turno=(TextView) itemView.findViewById(R.id.txtturno);
            fecha=(TextView) itemView.findViewById(R.id.txtfecha);
        }
    }
    public List<InfoHistorial> historial;
    public AdpHistorial(List<InfoHistorial>historial) { this.historial=historial; }
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType){
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.campohistorial, parent, attachToRoot: false);
        ViewHolder viewholder =new ViewHolder(view);
        return viewholder;
    }
    @Override
    public void onBindViewHolder(AdpHistorial.ViewHolder holder, int position) {
        holder.cedula.setText("Cédula: " + String.valueOf(historial.get(position).getCedula()));
        holder.nombre.setText("Nombre: " + historial.get(position).getNombre());
        holder.turno.setText("Turno: " + historial.get(position).getTurno());
        holder.fecha.setText("Fecha: "+historial.get(position).getFecha());
    }
    @Override
    public int getItemCount() { return historial.size(); }
}
```

Infohistorial

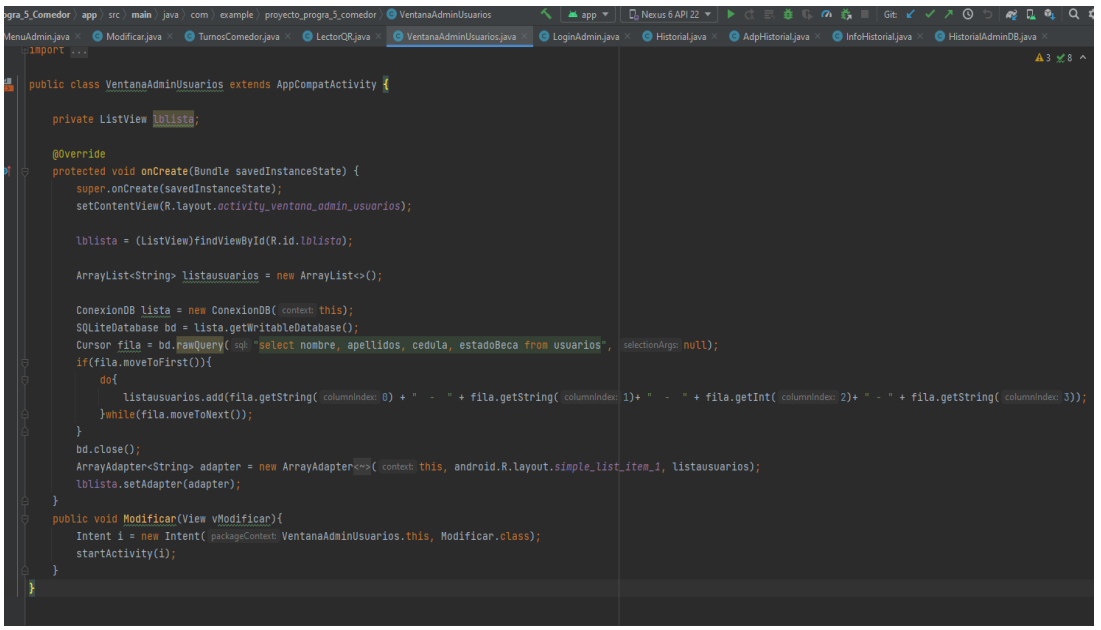
Se establecen las variables y las vista

Se crea infohistorial son los métodos get y set para los datos que se van a mandar a llamar en loscampos de campo



VentanaAdminUsuario

Aca se crea la lista donde se van a ver los datos de los estudiantes que están registrados en la app, se hace una conexión con la base de datos para que revise la tabla usuarios y agarre el nombre, apellido, cédula y estado de beca y los mande a mostrar en pantalla.

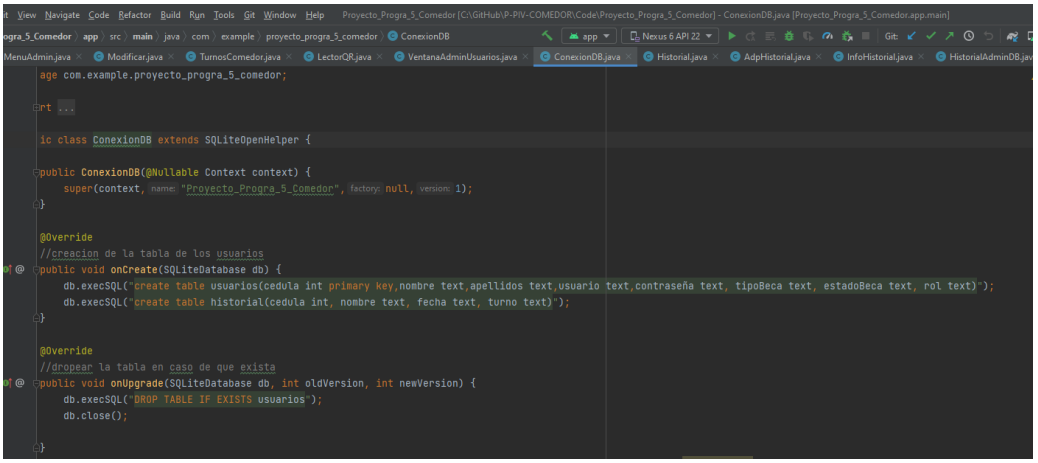


```
import ...

public class VentanaAdminUsuarios extends AppCompatActivity {

    private ListView lblista;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ventana_admin_usuarios);

        lblista = (ListView)findViewById(R.id.lblista);

        ArrayList<String> listausuarios = new ArrayList<>();

        ConexionDB lista = new ConexionDB( context: this);
        SQLiteDatabase bd = lista.getWritableDatabase();
        Cursor fila = bd.rawQuery( sql: "select nombre, apellidos, cedula, estadoBeca from usuarios",  selectionArgs: null);
        if(fila.moveToFirst()){
            do{
                listausuarios.add(fila.getString( columnIndex: 0) + "  -  " + fila.getString( columnIndex: 1)+ "  -  " + fila.getInt( columnIndex: 2)+ " - " + fila.getString( columnIndex: 3));
            }while(fila.moveToNext());
        }
        bd.close();
        ArrayAdapter<String> adapter = new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1, listausuarios);
        lblista.setAdapter(adapter);
    }
    public void Modificar(View vModificar){
        Intent i = new Intent( packageContext: VentanaAdminUsuarios.this, Modificar.class);
        startActivity(i);
    }
}
```

ConexiónBD

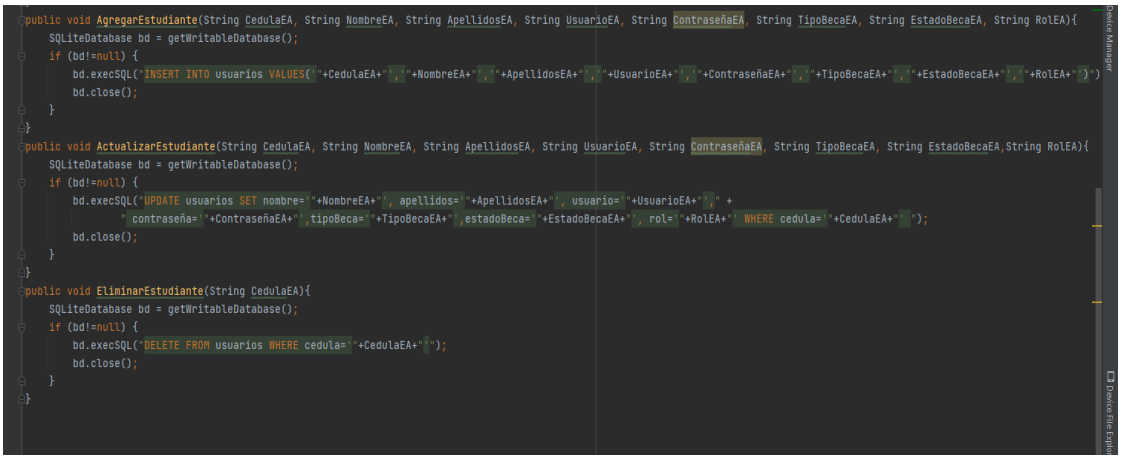
Se crea el constructor de la conexiónDB que este caso sería un context, se manda a llamar un super con el nombre del proyecto.

Se crea el oncearte con las tablas, luego se crea un onUpgrape



```
age com.example.proyecto_progra_5_comedor;

rt ...

ic class ConexionDB extends SQLiteOpenHelper {

public ConexionDB(@Nullable Context context) {
    super(context, name: "Proyecto_Progra_5_Comedor", factory: null, version: 1);
}

@Override
//creacion de la tabla de los usuarios
public void onCreate(SQLiteDatabase db) {
    db.execSQL("create table usuarios(cedula int primary key,nombre text,apellidos text,usuario text,contraseña text, tipoBeca text, estadoBeca text, rol text)");
    db.execSQL("create table historial(cedula int, nombre text, fecha text, turno text)");
}

@Override
//dropear la tabla en caso de que exista
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS usuarios");
    db.close();

}
```

Se crean tres métodos que serían para agregar, modificar o actualizar y eliminar los datos del estudiante.



```
public void AgregarEstudiante(String CedulaEA, String NombreEA, String ApellidosEA, String UsuarioEA, String ContraseñaEA, String TipoBecaEA, String EstadoBecaEA, String RolEA){
    SQLiteDatabase bd = getWritableDatabase();
    if (bd!=null) {
        bd.execSQL("INSERT INTO usuarios VALUES('"+CedulaEA+"','"+NombreEA+"','"+ApellidosEA+"','"+UsuarioEA+"','"+ContraseñaEA+"','"+TipoBecaEA+"','"+EstadoBecaEA+"','"+RolEA+"')");
        bd.close();
    }
}
public void ActualizarEstudiante(String CedulaEA, String NombreEA, String ApellidosEA, String UsuarioEA, String ContraseñaEA, String TipoBecaEA, String EstadoBecaEA,String RolEA){
    SQLiteDatabase bd = getWritableDatabase();
    if (bd!=null) {
        bd.execSQL("UPDATE usuarios SET nombre='"+NombreEA+"', apellidos='"+ApellidosEA+"', usuario='"+UsuarioEA+"'," +
                " contraseña='"+ContraseñaEA+"',tipoBeca='"+TipoBecaEA+"',estadoBeca='"+EstadoBecaEA+"', rol='"+RolEA+"' WHERE cedula='"+CedulaEA+"' ");
        bd.close();
    }
}
public void EliminarEstudiante(String CedulaEA){
    SQLiteDatabase bd = getWritableDatabase();
    if (bd!=null) {
        bd.execSQL("DELETE FROM usuarios WHERE cedula='"+CedulaEA+"'");
        bd.close();
    }
}
```