Desafío Practico 2

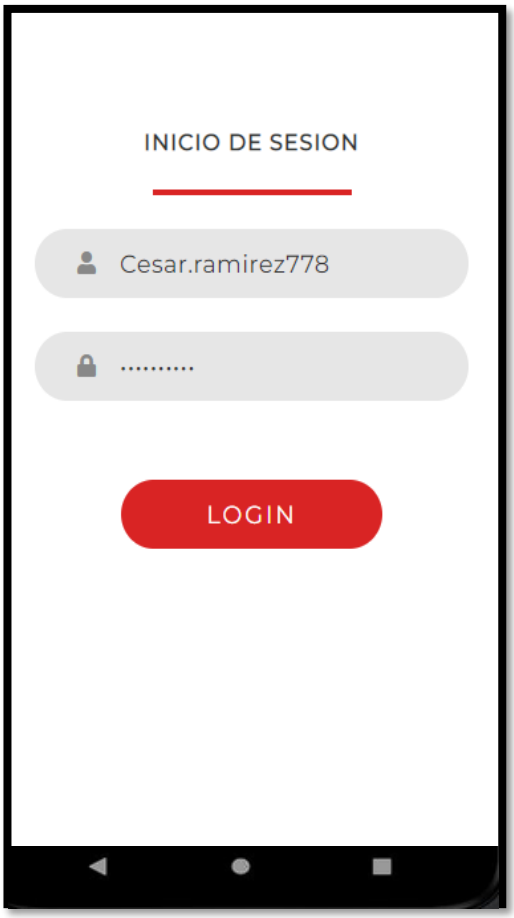
Desarrollo de Software para Móvil

Nelson Ernesto Muñoz Barahona MB192012

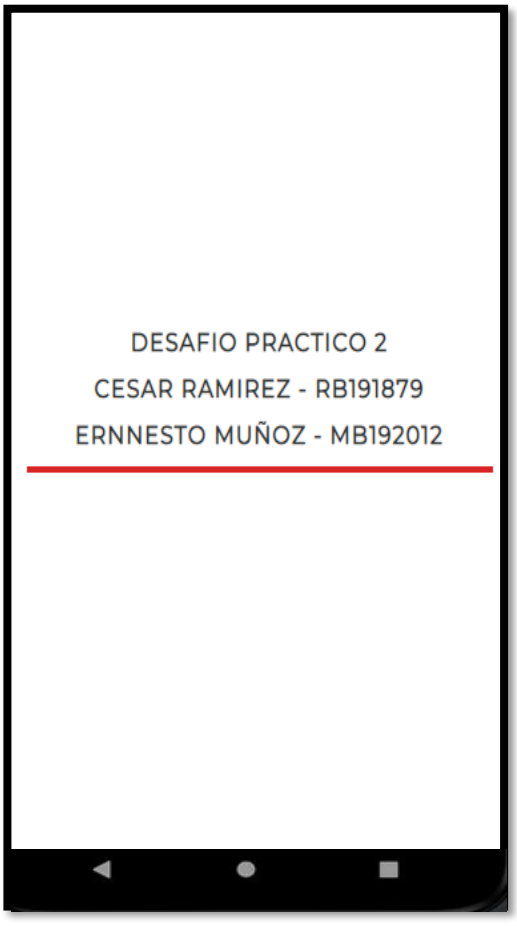
Cesar Antonio Ramírez Barahona RB19187

En este desafío se utilizo programación con metodología POO , lo cual se busca reutilizar los componentes y de esa manera agilizar el proceso de reutilización de código y haciendo mas rápida la manera de programar

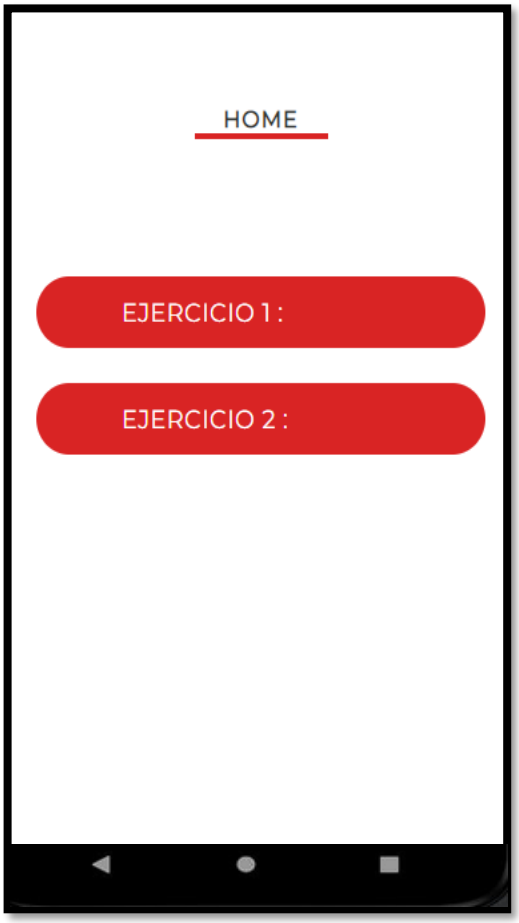
```kotlin
//METODO QUE VALIDA LA CUENTA DEL USUARIO
private fun loginUserAccount() {
    progressBar?.setVisibility(View.VISIBLE)
    //variables
    val email: String
    val password: String
    email = emailTV?.getText().toString()
    password = passwordTV?.getText().toString()
    //Validacion si esta vacio el txt Usuario
    if (TextUtils.isEmpty(email)) {
        Toast.makeText(applicationContext, "Please enter Username...", Toast.LENGTH_LONG)
            .show()
        return
    }
    //Validacion si esta vacio el txt contraseña
    if (TextUtils.isEmpty(password)) {
        Toast.makeText(applicationContext, "Please enter password!", Toast.LENGTH_LONG)
            .show()
        return
    }
    mAuth?.signInWithEmailAndPassword(email, password)
        ?.addOnCompleteListener { task ->
            // SI LA TAREA RESULTA CORRECTA, DEJARIA PODER ENTRAR AL MAIN SCREEN
            if (task.isSuccessful) {
                Toast.makeText(
                    applicationContext,
                    "Bienvenido",
                    Toast.LENGTH_LONG
                ).show()
                progressBar?.setVisibility(View.GONE)
                val intent = Intent(this@LoginActivity, DashboardActivity::class.java)
                startActivity(intent)  ^addOnCompleteListener
                // CASO CONTRARIO DEBERIA DAR ERROR EN UN TOAST
            } else {
                Toast.makeText(
                    getApplicationContext(),
                    "Inicio fallido intente nuevamente",
                    Toast.LENGTH_LONG
                ).show()  ^addOnCompleteListener
            }
        }
}
```



•PANTALLA DE BIENVENIDA

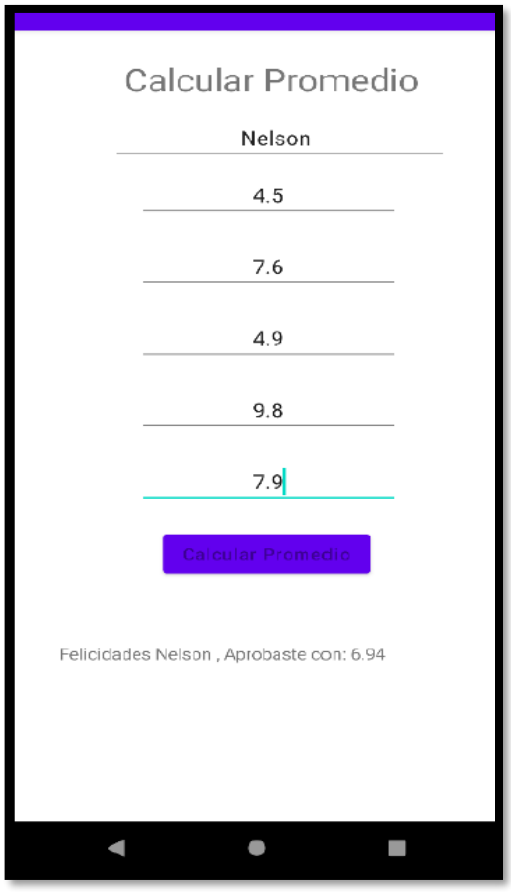


```
/// OPCIONES DE MENU segun ejercicio :
class MENU : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_dashboard)
    }
    // Evenetos para cuando se selecciona una opcion en el menu
    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        val id = item.itemId
        if (id == R.id.ejercicio1) {
            Toast.makeText(this, "Se seleccionó la primer opción", Toast.LENGTH_LONG).show()
            val intent = Intent(this, LoginActivity::class.java)
            startActivity(intent)
        }
        if (id == R.id.ejercicio2) {
            Toast.makeText(this, "Se seleccionó la segunda opción", Toast.LENGTH_LONG).show()
            val intent = Intent(this, RegistrationActivity::class.java)
            startActivity(intent)
        }
        if (id == R.id.ejercicio3) {
            Toast.makeText(this, "Se seleccionó la tercer opción", Toast.LENGTH_LONG).show()
            val intent = Intent(this, RegistrationActivity::class.java)
            startActivity(intent)
        }
        return super.onOptionsItemSelected(item)
    }
}
```



• EJERCICIO 1

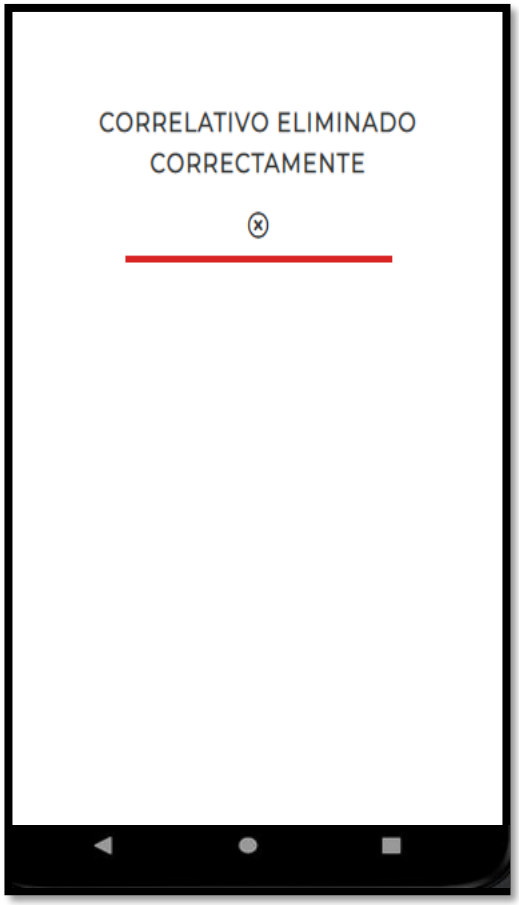
Para estos ejercicios reutilizamos los ejercicios del desafío 1 y agregamos su respectivo CRUD con un nuevo diseño ya que del código antiguo no se toco se hizo modificaciones a parte



• CRUD EJERCICIO 1:

No poseemos id ya que sacamos la información de local storages para agilizar los procesos sin las bd, asi que usamos correlativos en listas


```
//==========/ *ENCAPSULAMOS LOS VALORES EN VARIABLES PARA ASIGNARLES UN VALOR ///==========/
class Estudiante{
    var Nombre:String=""
    var Apellido:String=""
    var NotaPromedio:Double=0.0
    var Estado:boolean=""
//==========/ *ENCAPSULAMOS LOS VALORES EN VARIABLES PARA ASIGNARLES UN VALOR ///==========/
    fun EnlistarEstudiantes(nombre:String,nota1:Double,nota2:Double,nota3:Double,nota4:Double,nota5:Double){
       // validamos  si hay  valores asignados
    if(nombre,apellidom,notaPromedio,estado ==  ""){
        show.toast("No hay valores que mostrar")
    }
    //caso contrario tomamos los valores encapsulados y los mostramos en pantalla
    else{
        if(class.estudiantes.lenght)
        {
            Nombre = space.[opc]= nombre
            Apellido = space[opc]=apellido
            NotaPromedio = space[opc]=notapromedio
            Estado = space[opc]=estado
        } ^lambda
    }
```




```
//====/ *METODO que  Obtiene los valores encapsulados y elimina segun numero de correlativo del 1 al 10 ///==========/
public fun ELiminarEstudiante(nombre:String,apellido:string,promedio:double,estado:boolean)
{
    if(class.estudiantes.EliminarEstudiante==True)
    {
   // al capturar los datos a elimiar recorre un listado y los empuja a darles un valor nulo, para ser eliminados de la lista
        listOf(Nombre) nombre.toString[0]
        listOf(Apellido) apellido.push.toString[0]
        listOf(Promedio) promedio.push.toDouble[0]
        listOf(Estado) estado.push.to[empty]
        listView(reloaded)
    //Al ser eliminado manda a la vista de eliminado exitosamente segun correlativo.
        this.show(Eliminado) ^lambda
    }
    else(
        toast.show("No  se pudo eliminar el correlativo.")
         listView(reloaded)
    )

}
}
```


```
//====/ *METODO que  Obtiene los valores encapsulados y actualiza los datos segun el orden de ingreso//==========/
public fun ObtenerDatosEstudianteEditar(nombre:get,apellido:get,promedio:get,estado:get)
{
    if(class.estudiantes.EliminarEstudiante==True)
    {
        // al capturar los datos los obtiene con GET desde el encapsulado
        GET listOf(Nombre) nombre.toString[0]
        GET  listOf(Apellido) apellido.push.toString[0]
        GET listOf(Promedio) promedio.push.toDouble[0]
        GET listOf(Estado) estado.push.to[empty]
        GET listView(reloaded)
        //Al ser eliminado manda a la vista de eliminado exitosamente segun correlativo.
        this.show(views.editar) ^lambda
    }
public fun ObtenerDatosEstudianteEditar(nombre:get,apellido:get,promedio:get,estado:get)
    if()
    {
        // Asignamos nuevos Valores a los datos.
        SET listOf(Nombre) nombre.new[nombre]
        SET listOf(Apellido) apellido.new[apellido]
        SET listOf(Promedio) promedio.new[promedio]
        SET listOf(Estado) estado.new[estado]
        SET listView(reloaded)

        //Al ser editado los datas nos  manda a la vista de editado exitosamente segun correlativo.
        this.show(views.editar)
    }
    //si el dato no exite
    else{
        if(nombre==empty,apellido==empty,promedio==empty,estado==empty)
         toast.show("No  se pudo Editar el correlativo.")
         listView(reloaded)
    )
    }
}
```

• CRUD EJERCICIO 2:

De igual manera reutilizamos el ejerccio presentado en el desafio 1
Ya que estamos utilizando programación en POO obtenemos la ventaja de poder utilizar los mismos métodos y reciclar clases y eventos que funcionarían de la misma manera con diferentes datos.

Agregamos 2 variables para reutilizar las clases de encapsulamiento

```
//===========/ *ENCAPSULAMOS LOS VALORES EN VARIABLES PARA ASIGNARLES UN VALOR ///===========/
class Estudiante{
    var Nombre:String=""
    var Apellido:String=""
    var NotaPromedio:Double=0.0
    var Estado:boolean=""

    var salarioNeto
    var salario
```

Reutilizamos código agregando 2 variables y solo mandándolas
A llamar cuando sea necesario

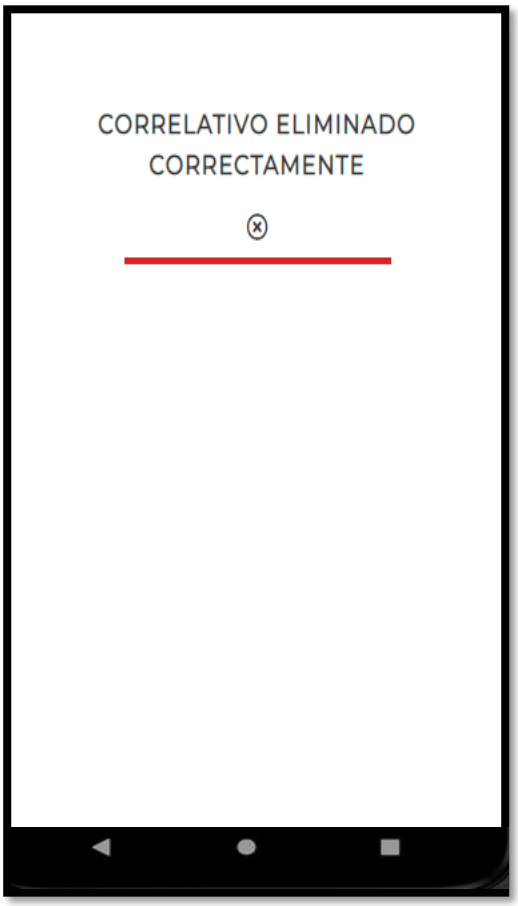
```
//===========/ *Enlistamos LOS datos EN VARIABLES PARA ASIGNARLES UN VALOR ///===========/
    fun EnlistarEmpleados(nombre:String,salario:int,Salneto:double){
       // validamos  si hay  valores asignados
    if(nombre,apellidom,notaPromedio,estado ==  ""){
        show.toast("No hay valores que mostrar")
    }
    //caso contrario tomamos los valores encapsulados y los mostramos en pantalla
    else{
        if(class.estudiantes.lenght)
        {
            Nombre = space.[opc]= nombre
            Apellido = space[opc]=apellido
            NotaPromedio = space[opc]=notapromedio
            Estado = space[opc]=estado

            salario=space[opc]=SALARIO
            Salneto=space[opc]=SalarioNeto

        } ^lambda
    }
```


```
//====/ *METODO que  Obtiene los valores encapsulados y actualiza los datos segun el orden de ingrerso//===========/
public fun ObtenerDatosEmpleadosEditar(nombre:String,salario:int,Salneto:double)
{
    if(class.estudiantes.EliminarEstudiante==True)
    {
        // al capturar los datos los obtiene con GET desde el encapsulado
        GET listOf(Nombre) nombre.toString[0]
        GET  listOf(Apellido) apellido.push.toString[0]
        GET listOf(Promedio) promedio.push.toDouble[0]
        GET listOf(Estado) estado.push.to[empty]
        GET listView(reloaded)
        //Al ser eliminado manda a la vista de eliminado exitosamente segun correlativo.
        this.show(views.editar) ^lambda
    }
public fun AsignarDatosEmpleadosEditar(nombre:get,salario:get,salneto:get)
    if()
    {
        // Asignamos nuevos Valores a los datos.
        SET listOf(Nombre) nombre.new[nombre]
        SET listOf(Salario) salario.new[apellido]
        SET listOf(salneto) SalarioNeto.new[salneto]
        //actualiza el listview
        SET listView(reloaded)

        //Al ser editado los datas nos  manda a la vista de editado exitosamente segun correlativo.
        this.show(views.editar)
    }
    //si el dato no exite
    else{
        if(nombre==empty,apellido==empty,promedio==empty,estado==empty)
         toast.show("No  se pudo Editar el correlativo.")
         listView(reloaded)
    )
    }
```




```
//====/ *METODO que  Obtiene los valores encapsulados y elimina segun numero de correlativo del 1 al 10 ///==========/
public fun ELiminarEmpleado(nombre:String,salario:int,Salneto:double)
{
    if(class.empleados.ELiminarEmpleado == True)
    {
   // al capturar los datos a elimiar recorre un listado y los empuja a darles un valor nulo, para ser eliminados de la lista
        listOf(Nombre) nombre.toString[0]
        listOf(salario) salario.push.toInt[0]
        listOf(salneto) sneto.push.toDouble[0]
        //Recarga el listado
        listView(reloaded)
    //Al ser eliminado manda a la vista de eliminado exitosamente segun correlativo.
        this.show(views.Eliminado)  ^lambda
    }
    else(
        toast.show("No  se pudo eliminar el correlativo.")
         listView(reloaded)
    )

}
}
```