

Proyecto Final App Moviles en condición de libre

Proyecto: control-proveedores

Alumno: Tomas Lencina

Carrera: Desarrollo web y App Moviles

Proceso de Desarrollo

La idea de este proyecto surge en la fase de planificación, tomando como base la propuesta para el desarrollo de la aplicación. Ante la diversidad de opciones que se presentan conforme a los requisitos establecidos, se plantean varias alternativas para su consideración. Sin embargo, mi elección se inclina hacia aquella que me permita abordar las consignas de la manera más eficaz y rápida posible, priorizando así la eficiencia en la ejecución del proyecto.

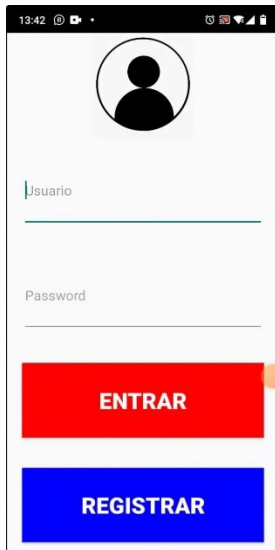
Entonces llego la idea a mi cabeza, desarrollar un proyecto el cual forme parte de un proceso común en las empresas. Específicamente me base en la célula que maneja las compras sobre materiales para diferentes proyectos de la organización, pensando en una app que sea de uso privado y forma parte de un sistema más grande dentro la misma empresa y este desarrollo maneje la parte de compras de insumos.

Pensé este desarrollo como una app a la cual solo tendrán acceso los empleados de una misma organización y que ellos serán los responsables de hacer la solicitud a los diferentes proveedores por medio de la app, con el fin de lograr evitar esperas innecesarias para los diferentes proyectos que se llevan a cabo dentro de la empresa.

La app estará conectada a una BD SQLite la cual almacenara los datos de los usuarios registrados, para poder darles acceso al momento del login. También almacenara los datos de todos aquellos pedidos que se soliciten levantar a los proveedores, indicando en cada pedido el proyecto al cual se le solicita la compra del insumo y especificando la cantidad. Estos pedidos se guardarán en la base de datos en la tabla que fue diseñada para manejar estos pedidos y dentro de la app abra una activity la cual muestre una lista de pedidos en curso para cada usuario.

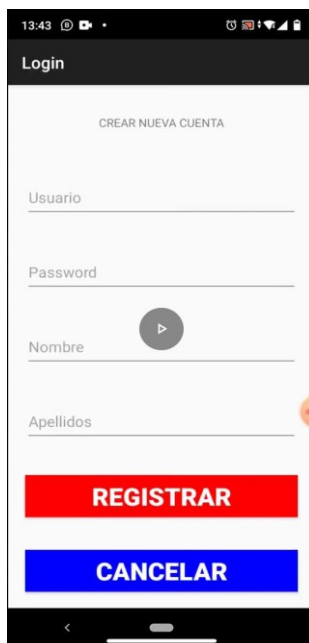
La base de datos cuenta con dos tablas, las cuales están conectadas por medio de una clave foránea, que en realidad es el ID de usuario que se asigna al registrar una cuenta, este id será el validador para todas las transacciones que se realicen en la app.

En cuanto a la planificación por parte del frontend pensé en una app sencilla, en la cual la pantalla de inicio sea el login. La idea es que ningún user no logeado pueda acceder a realizar pedidos. Dentro de la pantalla tendrás acceso a dos botones, el de registrar una cuenta y el entrar.



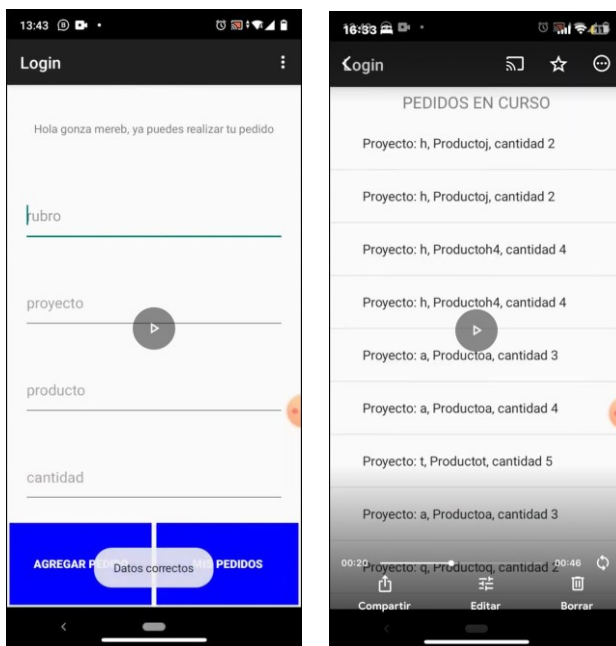
A mobile app login screen. At the top, there is a status bar with the time 13:42 and various icons. Below it is a circular profile icon placeholder. The form consists of two text input fields: 'Usuario' and 'Password'. Below the inputs are two buttons: a red 'ENTRAR' button and a blue 'REGISTRAR' button.

Otra vista que tiene la app es la que se encarga de registrar una cuenta, se accede a esta vista por medio del btn “REGISTRAR” que hay en la pantalla de login, y es una interfaz que contiene 4 campos para ingresar texto(usuario-contraseña-nombre-apellido), es fácil y rápido. La particularidad es que al momento de ingresar un user este debe tener la cuenta de mail de la empresa (Para este caso la cuenta deberá tener “@proveedores” en el user o no se le permitirá el ingreso), esta es una técnica de ciberseguridad para poder limitar el acceso de registro de cuentas a personas ajenas a la empresa. Otra particularidad también es que la contraseña deberá ser robusta y segura, de lo contrario no podrás registrarte (contraseña mayor a 8 caracteres, combinando mayúsculas-minúsculas-números)

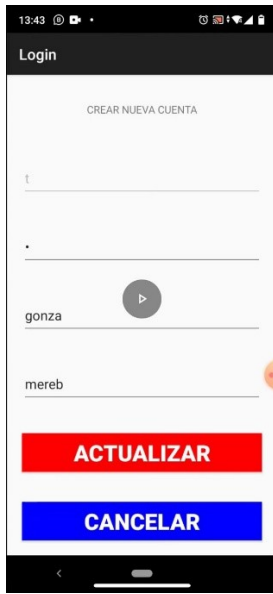


A mobile app registration screen. At the top, there is a status bar with the time 13:43 and various icons. Below it is a dark header with the text 'Login'. Under the header is a link 'CREAR NUEVA CUENTA'. The form consists of four text input fields: 'Usuario', 'Password', 'Nombre', and 'Apellidos'. There is a play button icon next to the 'Nombre' field. Below the inputs are two buttons: a red 'REGISTRAR' button and a blue 'CANCELAR' button. At the very bottom, there is a navigation bar with a back arrow and a home indicator.

Siguiendo con las vistas de la app, luego de logearte ingresas a la pantalla principal, aquí se gestionan los pedidos. En la parte superior de esta pantalla abra un mensaje de bienvenida para el dueño de la cuenta registrada, el cual contiene el nombre y el apellido que este user registro al momento de logearse. Debajo de este mensaje encontrara 4 campos de input-text para poder detallar cual es el proyecto-rubro-producto-cantidad para gestionar el pedido con toda la data necesaria para clasificar la compra y entregarla luego al equipo correcto. Luego de estos campos de texto tendremos dos botones, uno para agregar el pedido y otro para ver la lista de pedidos que solicito esta cuenta.

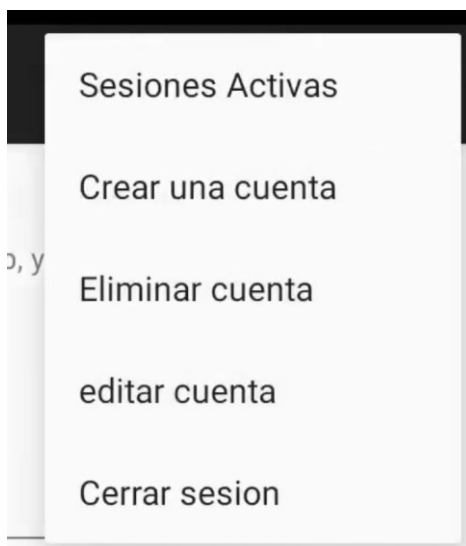


La aplicación cuenta con una vista para editar una cuenta ya registrada, la cual es una pantalla muy similar a la vista del registro que te permite editar/actualizar la contraseña-nombre-apellido del usuario, el único dato que no se puede cambiar es el mail-usuario registrado con la cuenta. A esta opción la encontraremos dentro del menú.

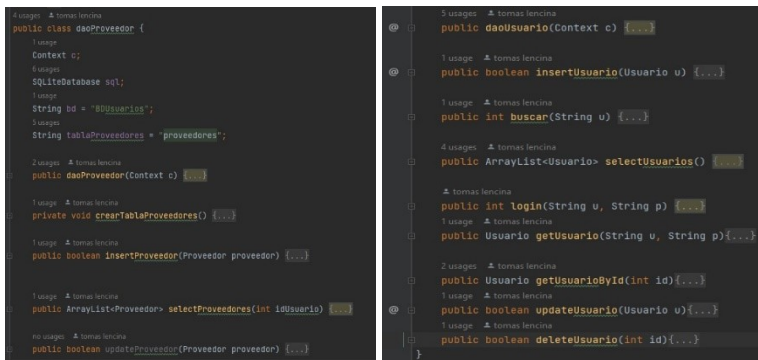


Dentro del menú de opciones de la app encontraras las opciones de:

- Ver sesiones activas (para controlar los últimos usuarios registrados que tuvieron actividad en la app)
- Crear Cuenta (podrás crear otra cuenta en caso de que te lo soliciten desde dentro de la app)
- Eliminar cuenta (Esta opción te permite eliminar tu cuenta permanentemente)
- Editar cuenta (podrás modificar contraseña-nombre-apellido)
- Cerrar sesión (volverías al login cerrando tu cuenta)



Luego de detallar los componentes de la estructura compartiré las clases las cuales se encargan de manejar y transportan los datos que se cargan en las diferentes vistas, para así mostrarlo en bd y mostrar cada dato del registro donde se lo necesite.



The image shows two side-by-side screenshots of Java code in an IDE. The left screenshot displays the code for a class named `daoProveedor`, which includes a `Context` parameter, an `SQLiteOpenHelper` instance, and methods for inserting, selecting, and updating providers. The right screenshot displays the code for a class named `daoUsuario`, which includes a `Context` parameter and methods for inserting, searching, selecting, logging in, getting, updating, and deleting users. Both classes use `SQLiteOpenHelper` to manage the database.

```
public class daoProveedor {  
    Context c;  
    SQLiteDatabase sq;  
    String bd = "BDUsuarios";  
    String tablaProveedores = "proveedores";  
  
    public daoProveedor(Context c) {  
        crearTablaProveedores();  
    }  
  
    public boolean insertProveedor(Proveedor proveedor) {  
        ArrayList<Proveedor> selectProveedores(int idUsuario);  
        boolean updateProveedor(Proveedor proveedor);  
    }  
}
```

```
@  
public daoUsuario(Context c) {  
    insertUsuario(Usuario u) {  
        buscar(String u) {  
            selectUsuarios() {  
                login(String u, String p) {  
                    getUsuario(String u, String p) {  
                        getUsuarioById(int id) {  
                            updateUser(Usuario u) {  
                                deleteUser(int id) {  
    }  
}
```

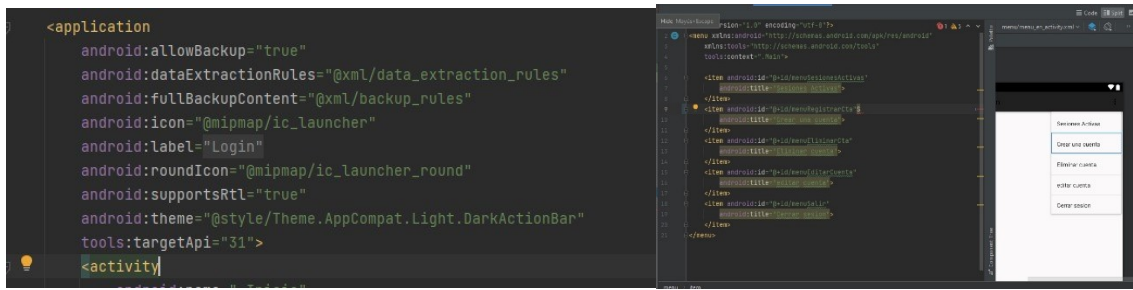
Dificultades que surgieron en el desarrollo:

El primer problema de desarrollo surgió cuando tuve que instalar Android studio en mi pc, me llevo aproximadamente 6 horas la instalación, y cuando quise empezar a desarrollar necesitaba instalar unas dependencias para poder utilizar el lenguaje java, porque la versión que instale traía como lenguaje por defecto a KOTLIN. Entonces investigué a través de diferentes foros y tutoriales hasta encontrar la solución. Y fue un comando el que me ayudó a instalar los paquetes necesarios en Android-studio para poder comenzar con el desarrollo de la app.

El segundo problema fue a la hora de emular las vistas de la app, me di cuenta que mi computadora no tenía los recursos necesarios para correr un emulador. Entonces debía buscar una alternativa para poder realizar el debug de lo que estaba desarrollando... Otra vez me puse a investigar opciones y encontré que Android studio te permite usar un dispositivo físico para emular y debuggear tu app. Como yo no contaba con un dispositivo Android un familiar me facilitó uno. Entonces una vez que configure el dispositivo, active los permisos de desarrollador y controle que la versión de Android del dispositivo era compatible con la versión en la que había desarrollado la app finalmente pude comenzar a realizar mis pruebas.

La tercera dificultad que se me presentó fue, que al momento de definir mi android-manifest sin darme cuenta seleccioné un tema el cual no me permitía visualizar el menú, por más que lo haya definido en los layouts la app no me los mostraba, investigando llegué a la conclusión que el error estaba en el tema. Entonces opté por elegir otro tema que me permitiera activar el menú y listo...

(android:theme="@style/Theme.AppCompat.Light.DarkActionBar")



La cuarta dificultad, la más estresante, caótica e importante de todas, fue que mi computadora decidió apagarse y no prenderse por aproximadamente 1 mes. La tuve que llevar a un service, realizaron una limpieza le agregue una RAM más grande y volvió a la vida. Este proble me detuvo y queda en evidencia con las fechas de los commit en GitHub, porque empecé con el proyecto y a realizar los primeros commits en enero y principios de febrero paso esto, y pude retomar el desarrollo a finales de febrero.

Conceptos de ciberseguridad aplicados en el proyecto

Contraseñas Seguras y Autenticación:

Concepto: Autenticación Robusta

Explicación: Implementar contraseñas seguras y complejas para las cuentas de usuario es esencial para garantizar la autenticación robusta. Utilizar combinaciones de letras, números y caracteres especiales ayuda a prevenir accesos no autorizados. Además, al requerir que los usuarios utilicen el correo electrónico de la empresa como parte de su identificación, se refuerza la autenticación y se evita el uso de cuentas personales no autorizadas.

```
// Función para validar requisitos de contraseña
1 usage  ▲ tomas lencina
private boolean passwordCumpleRequisitos(String password) {
    // Verificar que la contraseña contenga al menos una letra mayúscula, una letra minúscula y un número
    return password.matches( regex: "^(?=.*[A-Z])(?=.*[a-z])(?=.*\\d){8,}$");
}
```

Limitación de Intentos Fallidos

Concepto: Protección contra Ataques de Fuerza Bruta

Explicación: Al limitar el número de intentos de acceso fallidos en el proceso de inicio de sesión, se protege contra ataques de fuerza bruta. Este enfoque impide que los atacantes

prueben repetidamente combinaciones de contraseñas para obtener acceso no autorizado. La limitación de intentos fallidos ayuda a detectar y prevenir posibles amenazas antes de que tengan éxito.

```
public void onClick(View v) {
    if (v.getId() == R.id.btnEntrar) {
        String u = user.getText().toString();
        String p = pass.getText().toString();

        if (u.equals("") || p.equals("")) {
            Toast.makeText( context: this, text: "ERROR: Campos vacios", Toast.LENGTH_LONG).show();
        } else {
            if (System.currentTimeMillis() > tiempoBloqueo) {
                if (dao.login(u, p) == 1) {...}
            } else {
                // Login fallido
                intentosFallidos++;
                if (intentosFallidos >= 3) {
                    // Bloquear durante 5 minutos después de 3 intentos fallidos
                    tiempoBloqueo = System.currentTimeMillis() + 5 * 60 * 1000;
                    intentosFallidos = 0; // Reiniciar intentos fallidos
                    Toast.makeText( context: this, text: "Demasiados intentos fallidos. Bloqueado por 5 minutos.", T
                } else {
                    Toast.makeText( context: this, text: "Usuario y/o contraseña incorrectos", Toast.LENGTH_LONG).sh
                }
            }
        }
    } else {
        // Aún bloqueado
        Toast.makeText( context: this, text: "Cuenta bloqueada. Espera un momento antes de intentar de nuevo.", T
    }
}
```

Autoguardado Periódico de Bases de Datos:

Concepto: Resiliencia de Datos

Explicación: El autoguardado regular de bases de datos contribuye a la resiliencia de datos. En caso de un fallo del sistema, error humano o ataque, esta medida asegura que se disponga de copias actualizadas de la información. La resiliencia de datos es crucial para la continuidad del negocio y la rápida recuperación en situaciones adversas.

Una vez que tuve la app organizada, estructurada y desarrollada fue momento de empezar a testearla... Para ello desarrolle 3 casos de pruebas, me centre en el punto más importante de esta app, que es levantar pedidos a proveedores; y lo que se necesita para esto es que el usuario este logeado. Entonces me centre en hacer los casos de prueba comprobando logeo exitoso, logeo fallido y el levantamiento de pedidos:

Caso de prueba: Carga y visualización de pedidos
<p>OBJETIVO</p> <p>Verificar que la información del pedido se cargue correctamente en la vista después de realizar un pedido.</p>

Pasos a seguir:

- 1) **Logearse en la app**
- 2) **Acceso a la Pantalla de Pedido:**
- 3) **Ingresar datos para concretar un Pedido:**
-Llenar todos los campos requeridos del pedido(proyecto-producto-rubro-cantidad).
- 4) **Realizar el Pedido:**
Hacer clic en el botón agregar pedidos.
- 5) Revisar mensaje de pedido agregado
- 6) **Acceso a la Vista de Pedidos:**
Hacer clic en el botón agregar pedidos.
- 7) **Redireccion a la vista de pedidos**
- 8) Buscar el pedido recién cargado en la lista de pedidos.

Resultado esperado:

Luego de realizar un pedido, la información del pedido debería mostrarse correctamente en la vista de pedidos.

Los detalles del pedido, como el nombre del producto, la cantidad y la dirección, deberían coincidir exactamente con la información ingresada durante la creación del pedido.

Resultado obtenido:

Pedido registrado correctamente.

En la vista se observa toda la lista de pedidos de ese mismo usuario

Caso de prueba: Inicio de Sesión Fallido

OBJETIVO

Verificar que el sistema maneje correctamente los intentos de inicio de sesión con credenciales incorrectas y proporcione mensajes de error adecuados.

Pasos a seguir:

- 1) Acceder a la página de inicio de sesión desde la pantalla principal de la aplicación.
- 2) Introducción de Credenciales Incorrectas:
 - Ingresar un correo electrónico no registrado en la app
 - Ingresar una contraseña incorrecta (que no coincida con la contraseña registrada para ese usuario).
- 3) Reintentar el inicio de sesión.
- 4) verificar si los mensajes de error son iguales al resultado esperado
- 5) Reintentar el inicio de sesión
- 6) Verificar que después de 3 de intentos fallidos, el sistema bloquee temporalmente el acceso o muestre un mensaje indicando que el usuario ha sido bloqueado temporalmente por motivos de seguridad.

Resultado esperado:

El sistema debería manejar correctamente las credenciales incorrectas y mostrar un mensaje de error apropiado.

Después de varios intentos fallidos, el sistema debería implementar medidas de seguridad, como bloquear temporalmente el acceso.

Resultado obtenido:

Registro fallido exitosamente, mensajes de error adecuados para el caso de user/contraseña incorrecta, campos vacíos y bloqueo temporal luego de los 3 intentos fallidos de login

Caso de prueba: REGISTRO Y LOGIN EXITOSO**OBJETIVO**

Verificar que solo los usuarios con correos electrónicos de la empresa pueden registrarse(@proveedores) y que se requieran contraseñas robustas en el proceso de registro.

Pasos a seguir:

- 1) Acceder a la página de registro desde la pantalla principal de la aplicación.
- 2) Introducir de Datos de Registro:
 - Ingresar un correo electrónico que pertenezca a la empresa.
 - Ingresar una contraseña que cumpla con los requisitos de robustez (mínimo de 8 caracteres, al menos una letra mayúscula, al menos un número y al menos un carácter especial).
- 3) Presionar el botón “REGISTRAR” y verificar que se genera exitosamente el usuario
- 4) Ingresar en el login con el usuario y la contraseña creadas

Resultado esperado:

Lograr el registro de un nuevo empleado

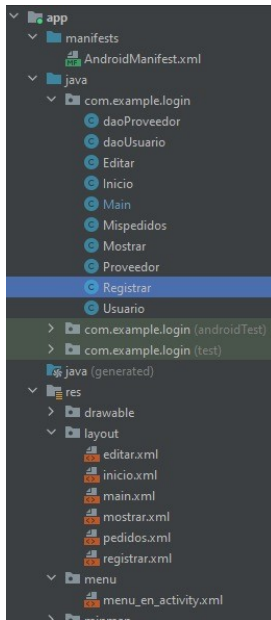
Poder ingresar a la app con el usuario y contraseña generada en el registro

Resultado obtenido:

Usuario registrado correctamente y redirección al login

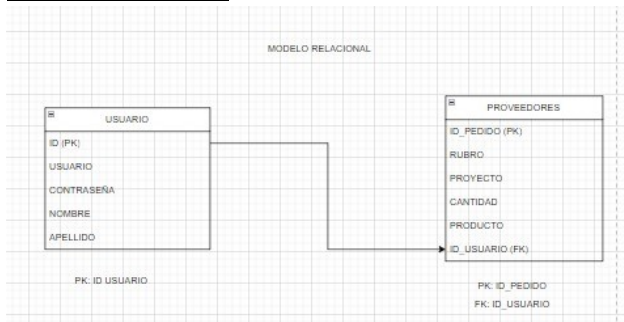
Ingreso con usuario y clave exitoso

Finalmente, mi proyecto adopto esta estructura, la cual consta de 10 clases, entre las cuales 2 de ellas manejan el control con la BD, otras gestionan los datos ingresados por los clientes, otras se encargan de darle actividad a todas las vistas que arme.

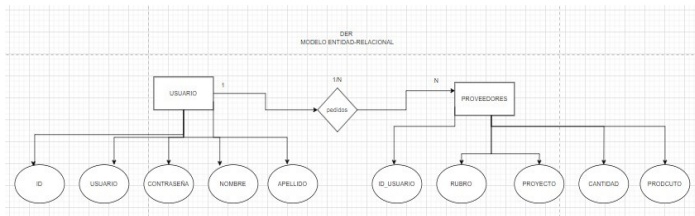


Diagramas:

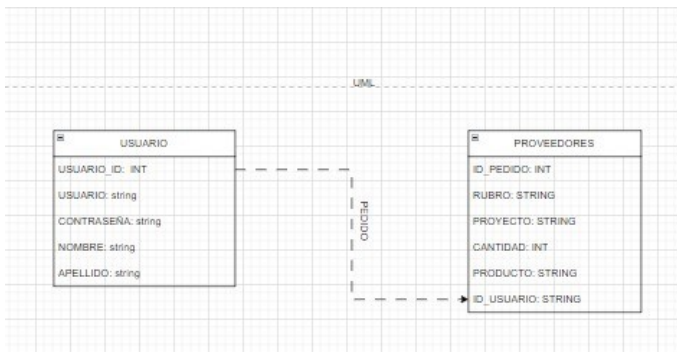
Modelo Relacional



Modelo entidad-relacion



Modelo UML



Conclusión: fue un desafío muy interesante que me llevo aproximadamente 2 meses, en los cuales me sorprendí por mi capacidad de buscar soluciones que al principio parecían imposibles de encontrarlas. También me amigue con java el cual era un lenguaje que me asustaba un poco la complejidad de su sintaxis.

Me llevo un buen proyecto a mi portfolio personal, el cual es sencillo, pero considero que tiene todo lo necesario para poder escalar en un futuro, me gustó mucho la experiencia