ILERNA

CFGS DESARROLLO DE APLICACIONES MULTIPLATAFORMA
Proyecto Final de Ciclo

TÍTULO DEL PROYECTO

Autor: José Manuel Flores Marzo y Antonio Guerrero García

Tutor: Javier Martin Rivero

Fecha de entrega:

Convocatoria: 2024 2025

Motivación (Ingles)

The motivation behind this project comes from my deep passion for the fitness world, which has grown stronger over time. I've always been curious to learn more about health, nutrition, and training. That passion sparked the idea for BuildYourBD—a fitness app designed to make the journey easier and more attractive, not only for enthusiasts like me, but especially for those who find it hard to stay consistent or get started.

While exploring different fitness apps, I noticed that most of them are either too focused on a single aspect—like workouts or calorie tracking—or overly complex. None of them seemed to offer a truly complete, user-friendly, and motivating experience. That gap inspired me to create an all-in-one solution.

BuildYourBD brings together essential fitness tools in one place: a food database for tracking calories and meals, personalized training routines, progress tracking features, hydration and workout reminders, and even community features to share recipes and workouts. It also includes dark mode and integration with APIs for gym locations and health data. My goal is to build not just another app, but a complete ecosystem that supports users in building long-lasting, healthy habits—helping them become the best version of themselves, both physically and mentally.

Diagrama de Clases

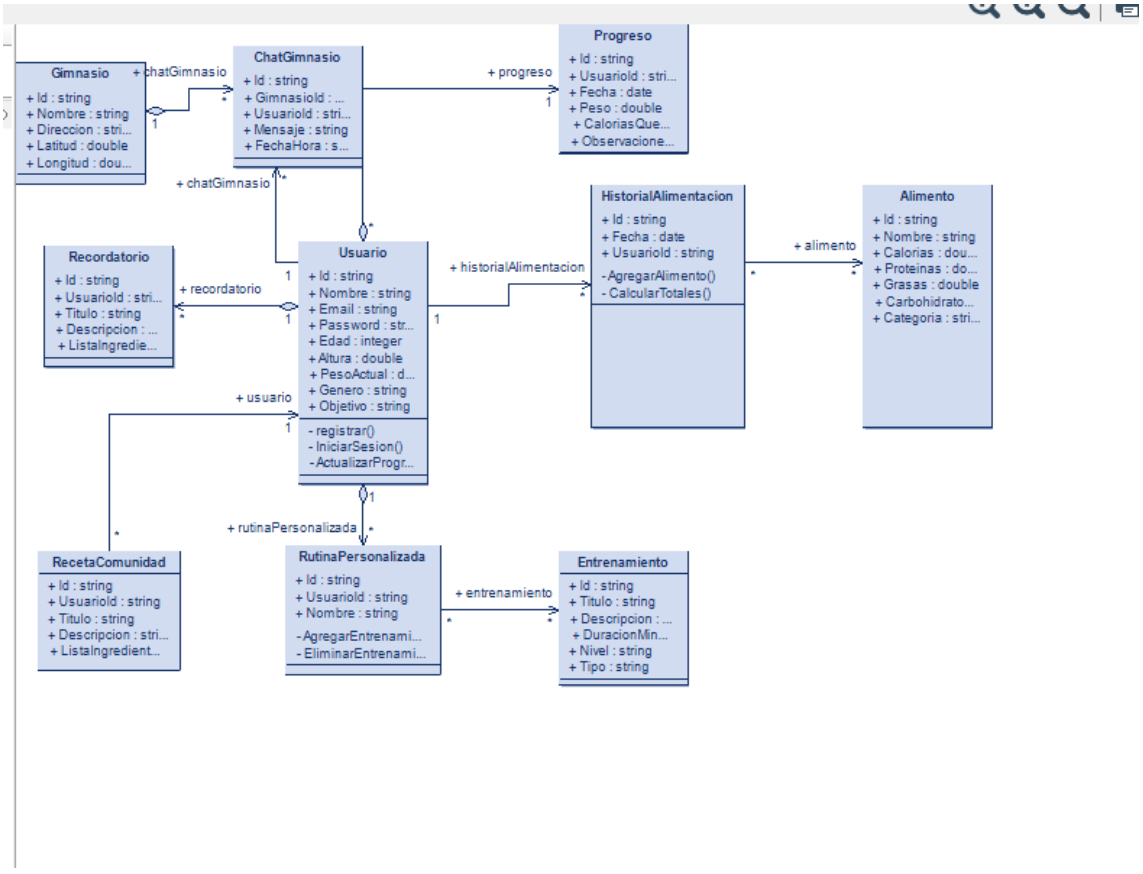
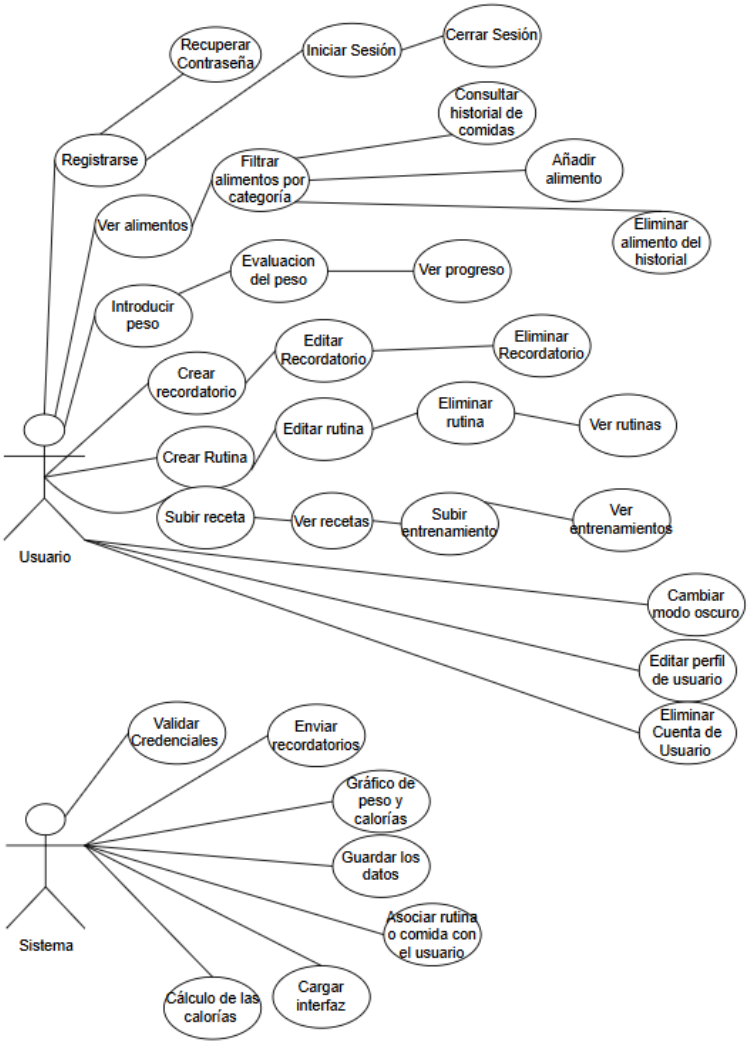


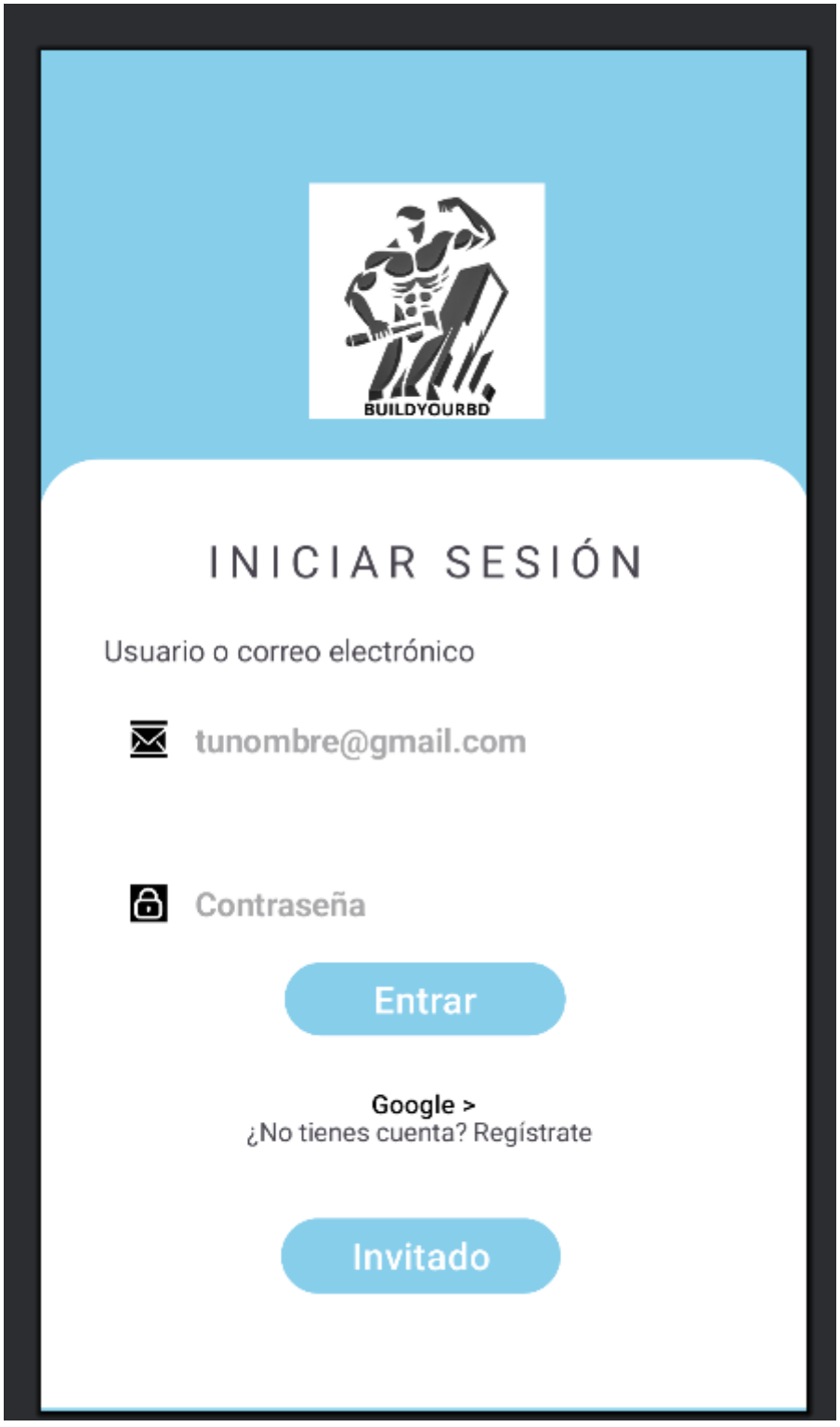
Diagrama de Casos de Uso



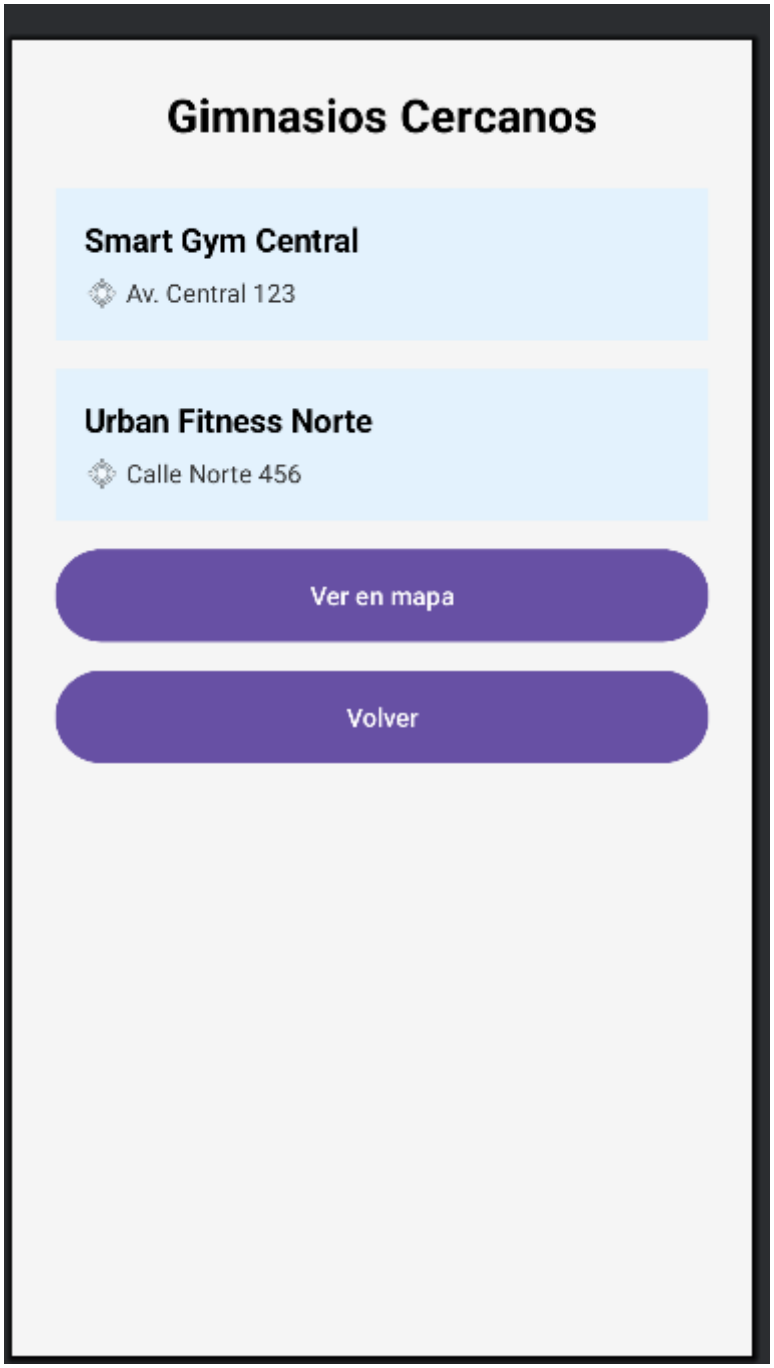
Prototipo

```
    // Lógica de inicio de sesión
    private fun login() {
        val email = loginEmail!!.text.toString()
        val password = loginPass!!.text.toString()

        if (email.isEmpty() || password.isEmpty()) {
            Toast.makeText( context: this,  text: "Por favor ingresa todos los campos", Toast.LENGTH_SHORT
            return
        }

        // Aquí se puede agregar la lógica para validar el email y la contraseña
        if (email == "usuario@dominio.com" && password == "123456") {
            Toast.makeText( context: this,  text: "Bienvenido", Toast.LENGTH_SHORT).show()
            // Redirigir a la pantalla principal o al dashboard
        } else {
            Toast.makeText( context: this,  text: "Credenciales incorrectas", Toast.LENGTH_SHORT).show()
        }
    }

    // Lógica para iniciar sesión como invitado
    private fun loginAsGuest() {
        Toast.makeText( context: this,  text: "Iniciando sesión como invitado", Toast.LENGTH_SHORT).show()
        // Aquí iría la lógica para iniciar sesión como invitado
    }

    // Lógica para iniciar sesión con Google
    private fun loginWithGoogle() {
        Toast.makeText( context: this,  text: "Iniciando sesión con Google", Toast.LENGTH_SHORT).show()
        // Aquí iría la lógica para la autenticación de Google
    }

    // Lógica para ir a la pantalla de registro
    private fun goToRegister() {
        Toast.makeText( context: this,  text: "Redirigiendo a registro", Toast.LENGTH_SHORT).show()
        // Aquí puedes redirigir a una actividad de registro
    }
}
```

En este caso la lógica para iniciar sesión en el que básicamente trabajamos el que no haya algún campo vacío o que se intente insertar algo fuera de los campos puestos en el código .



Con la página de inicio de sesión



Búsqueda de gimnasios según tu ubicación

En este caso necesitaremos la implementación de varias cosas

```
implementation 'com.google.android.gms:play-services-maps:18.1.0'
implementation 'com.google.android.gms:play-services-location:21.0.1'
```

La implementación para la configuración de Google maps, añadir estas dos líneas para los permisos de ubicación :

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Y no menos importante nuestra api key de Google maps.

```
package com.example.buildyourbd


import android.content.Context
import android.location.Location
import com.google.android.gms.location.LocationServices
import com.google.android.gms.tasks.OnSuccessListener

class Gimnasios(private val context: Context) {

    private val fusedLocationClient = LocationServices.getFusedLocationProviderClient(context)

    fun obtenerUbicacionUsuario(callback: (Double, Double) -> Unit) {
        if (androidx.core.content.ContextCompat.checkSelfPermission(
                context, android.Manifest.permission.ACCESS_FINE_LOCATION
            ) == android.content.pm.PackageManager.PERMISSION_GRANTED
        ) {
            fusedLocationClient.lastLocation.addOnSuccessListener { location: Location? ->
                location?.let {
                    callback(it.latitude, it.longitude)
                }
            }
        }
    }
}
```

Y este seria nuestro código para obtener la ubicación del usuario.

Pruebas valores límite

```
import org.junit.Assert
import org.junit.Test


class LoginTest {
    // Prueba para un correo mínimo de longitud válida (7 caracteres)
    @Test
    fun testValidEmailMinLength() {
        val email = "a@a.com" // Mínimo 7 caracteres
        val result = isValidEmail(email)
        Assert.assertTrue( message: "El correo debería ser válido", result)
    }

    // Prueba para un correo con 100 caracteres (máximo permitido)
    @Test
    fun testValidEmailMaxLength() {
        val email = "a".repeat( n: 94) + "@domain.com" // 100 caracteres
        val result = isValidEmail(email)
        Assert.assertTrue( message: "El correo debería ser válido", result)
    }

    // Método de validación simple para el correo
    private fun isValidEmail(email: String?): Boolean {
        return email != null && email.matches("^[\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$".toRegex())
    }
}


class LoginTest2 {
    // Prueba para una contraseña mínima válida (6 caracteres)
    @Test
    fun testValidPasswordMinLength() {
        val password = "123456" // Mínimo 6 caracteres
        val result = isValidPassword(password)
        Assert.assertTrue( message: "La contraseña debería ser válida", result)
    }
```

```kotlin
class LoginTest2 {
    // Prueba para una contraseña mínima válida (6 caracteres)
    @Test
    fun testValidPasswordMinLength() {
        val password = "123456" // Mínimo 6 caracteres
        val result = isValidPassword(password)
        Assert.assertTrue( message: "La contraseña debería ser válida", result)
    }

    // Prueba para una contraseña máxima válida (20 caracteres)
    @Test
    fun testValidPasswordMaxLength() {
        val password = "12345678901234567890" // Máximo 20 caracteres
        val result = isValidPassword(password)
        Assert.assertTrue( message: "La contraseña debería ser válida", result)
    }

    // Método de validación simple para la contraseña
    private fun isValidPassword(password: String?): Boolean {
        return password != null && password.length >= 6 && password.length <= 20
    }
}
```