Evaluación Sumativa 2 Dispositivos Móviles

Grupo 1

Sincronizar los proyectos individuales

Ejercicio 2

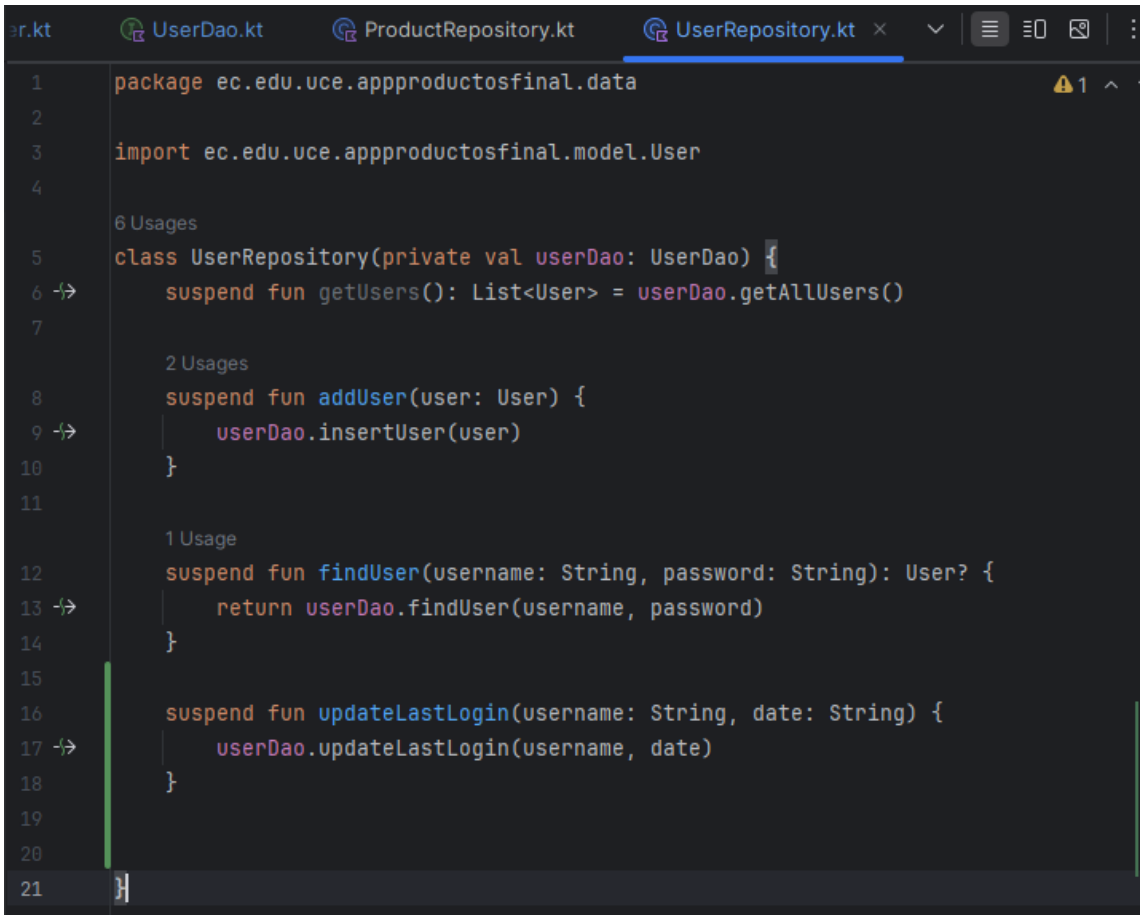
Creacion de lastlogin en User

```kotlin
package ec.edu.uce.appproductosfinal.model

import ...

27 Usages
@Entity(tableName = "users")
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val nombre: String,
    val apellido: String,
    val password: String, // Hash SHA-256
    val lastUpdated: Long = System.currentTimeMillis(),
    val lastLogin: String? = null
)
```

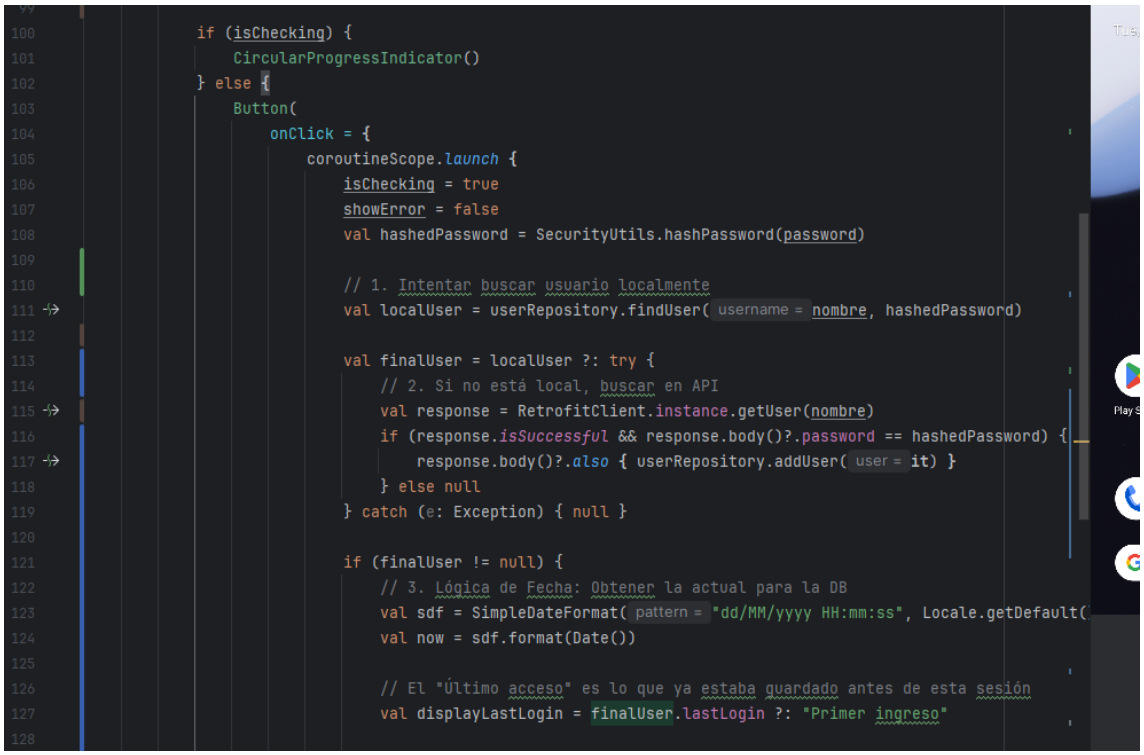
Se añadió la función updateLastLogin para actualizar solo la fecha cuando el usuario entre exitosamente

```kotlin
package ec.edu.uce.appproductosfinal.data

import ...

6 Usages  1 Implementation
@Dao
interface UserDao {
    1 Usage  1 Implementation
    @Query( value = "SELECT * FROM users")
    suspend fun getAllUsers(): List<User>

    1 Usage  1 Implementation
    @Insert
    suspend fun insertUser(user: User): Long

    1 Usage  1 Implementation
    @Query( value = "SELECT * FROM users WHERE LOWER(nombre) = LOWER(:username) AND
    suspend fun findUser(username: String, password: String): User?

    @Query( value = "UPDATE users SET lastLogin = :date WHERE nombre = :username")
    suspend fun updateLastLogin(username: String, date: String)
}
```

Exponemos la función de actualización y creamos una para obtener el usuario actual

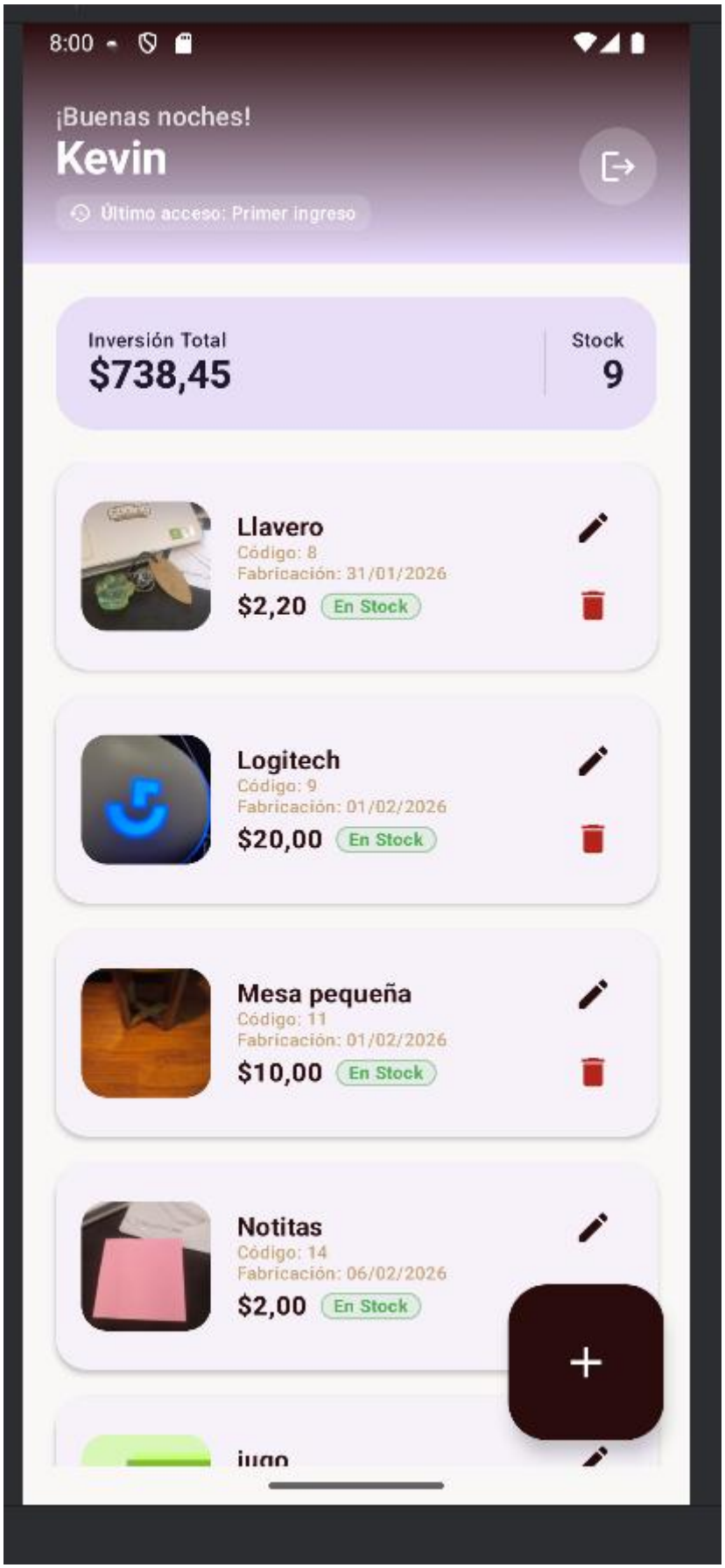


En LoginScreen.t Se incorporó el uso de las clases `SimpleDateFormat` y `Date` de Java para generar una estampa de tiempo en el momento exacto del clic.



```
if (isChecking) {
    CircularProgressIndicator()
} else {
    Button(
        onClick = {
            coroutineScope.launch {
                isChecking = true
                showError = false
                val hashedPassword = SecurityUtils.hashPassword(password)

                // 1. Intentar buscar usuario localmente
                val localUser = userRepository.findUser( username = nombre, hashedPassword)

                val finalUser = localUser ?: try {
                    // 2. Si no está local, buscar en API
                    val response = RetrofitClient.instance.getUser(nombre)
                    if (response.isSuccessful && response.body()?.password == hashedPassword) {
                        response.body()?.also { userRepository.addUser( user = it) }
                    } else null
                } catch (e: Exception) { null }

                if (finalUser != null) {
                    // 3. Lógica de Fecha: Obtener la actual para la DB
                    val sdf = SimpleDateFormat( pattern = "dd/MM/yyyy HH:mm:ss", Locale.getDefault(
                    val now = sdf.format(Date())

                    // El "Último acceso" es lo que ya estaba guardado antes de esta sesión
                    val displayLastLogin = finalUser.lastLogin ?: "Primer ingreso"
```

Prueba de la app



En la Base de Datos (User y UserDao)

Añadimos el campo `lastLogin: String?` a la tabla de usuarios, Creamos una nueva consulta `@Query` de tipo `UPDATE`. Esto permite que la aplicación modifique únicamente la fecha de acceso

Lógica de Login (`LoginScreen`)

Inmediatamente después, genera una nueva fecha con el momento actual (`SimpleDateFormat`) y la guarda en la base de datos para la próxima vez.

Navegación (MainActivity)

Rutas dinámicas: Modificamos la ruta de navegación de home/{userName} a home/{userName}/{lastLogin}.

Interfaz de Usuario (HomeScreen)

Encabezado (TopBar): Se rediseñó la parte superior para incluir un "Badge" o distintivo visual.