

Relatório de Desenvolvimento de Jogos para Plataformas Móveis

Prof. Lourenço Gomes

Índice

Estrutura do projeto / Protótipo do projeto.....	3
Lista de funcionalidades da <i>app</i>	5
Modelo de dados	5
Implementação do projeto.....	6
Tecnologias usadas	9
Dificuldades	10
Conclusão	10

Estrutura do projeto / Protótipo do projeto

Para melhor planeamento de como fazer este projeto, foi realizado um esquema básico de como o mesmo funcionaria:

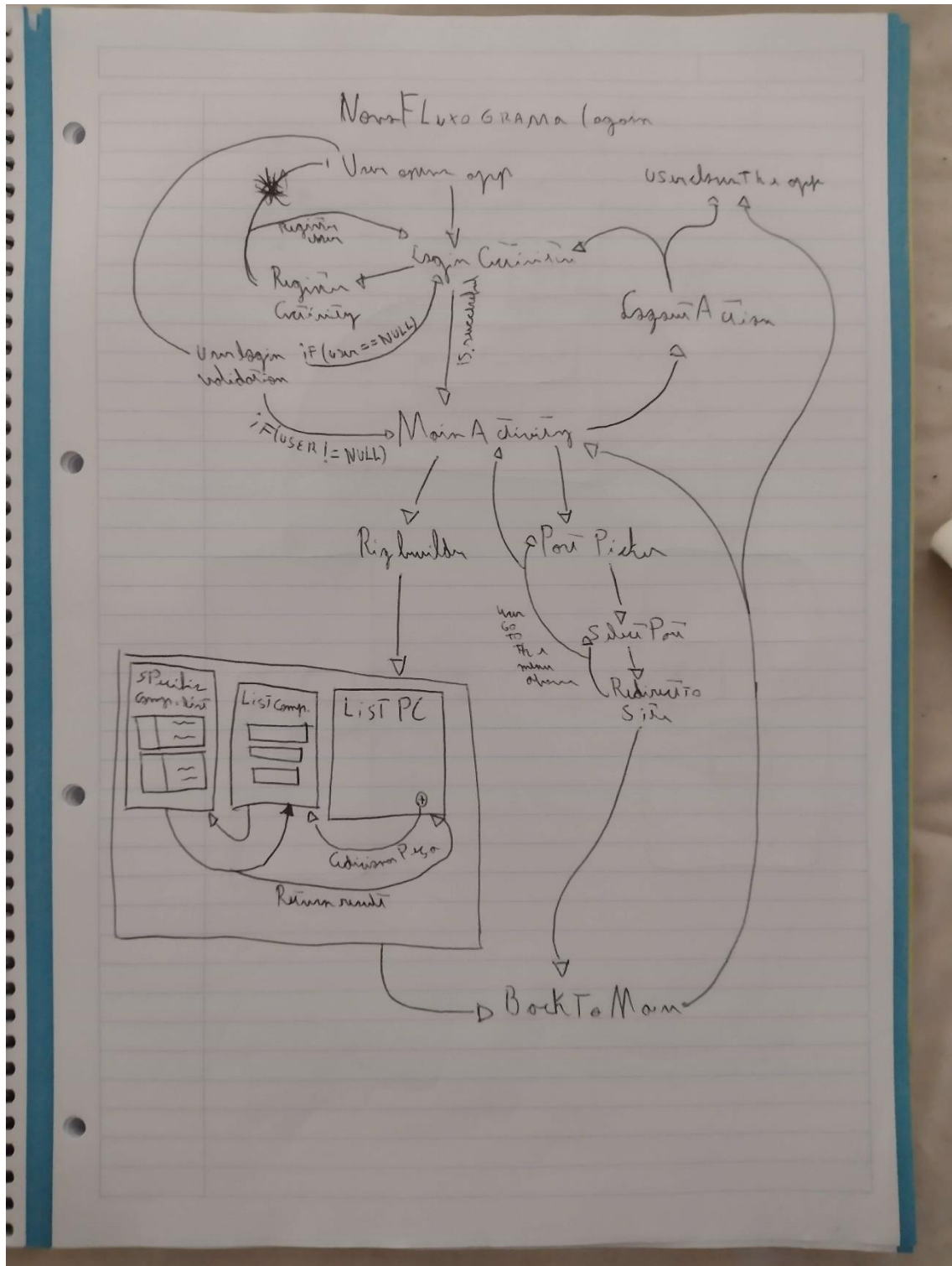


Figura 1 - Esquema representante do flow da app, e protótipo da mesma.

Na figura acima, está explicado que:

Após o utilizador abrir a *app*, irá aparecer o ecrã de *login*. Caso o utilizador não tenha uma conta, poderá criá-la na secção de “Registar”, e se o utilizador tiver já dado *login* numa sessão anterior, irá manter a sessão ativa automaticamente, e irá saltar diretamente para o ecrã principal da *app*.

Depois de dar *login*, irá aparecer o ecrã principal da *app*, onde esta irá pedir ao utilizador que selecione a peça que deseja ver.

Ao selecionar, ela irá pedir que selecione uma marca específica, e um modelo para a peça desejada.

No final, irão aparecer uma lista de URLs com a listagem da peça específica nas lojas de informática em Portugal mais famosas.

No menu principal, também aparece outra opção: o *Rig Builder*.

Nele, o utilizador poderá fazer uma configuração para criar um PC personalizado.

Lista de funcionalidades da app

O utilizador irá poder:

1. Fazer uma configuração para criar um PC personalizado;
2. Poder procurar uma peça específica, seleccionando o modelo e a marca da respetiva;
3. Utilizar a respetiva conta para guardar as configurações de PC feitas anteriormente.

Modelo de dados




 trabalho-de-mobile	 CPU_AMD	 Ryzen_9_7950X
+ Iniciar coleção	+ Adicionar documento	+ Iniciar coleção
CPU_AMD >	Ryzen_9_7950X >	+ Adicionar campo
CPU_INTEL	Ryzen™_5_5600X	Cores: 16
GPU_AMD	Ryzen™_5_7600X	Frequency: "5.7 GHz"
GPU_Intel	Ryzen™_7_5700G	Need discrete graphics card: false
GPU_NVidia	Ryzen™_7_5700X	Socket: "AM5"
Motherboard_AMD	Ryzen™_7_5800X	TBP: "170W"
Motherboard_INTEL	Ryzen™_7_7700X	Threads: 32
users	Ryzen™_9_5900X	
	Ryzen™_9_5950X	
	Ryzen™_9_7900X	

Figura 2 - Representação da base de dados utilizada.

Na figura acima, é possível observar a base de dados utilizada (mais especificamente, no Firebase).

Esta está agrupada por produtos, como possível de observar, e em cada produto, obtêm-se uma série de características que variam com o mesmo.

Implementação do projeto

```
class LoginActivity : AppCompatActivity() {
    private lateinit var email :EditText
    private lateinit var password:EditText
    private lateinit var loginButton :Button
    private lateinit var goToRegisterButton: Button

    private lateinit var auth:FirebaseAuth

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        email = findViewById(R.id.login_email_input)
        password = findViewById(R.id.login_password_input)
        loginButton = findViewById(R.id.login_button)
        goToRegisterButton = findViewById(R.id.login_register_button)

        auth = FirebaseAuth.getInstance()

        loginButton.setOnClickListener{ it: View!
            var txt_email : String = email.text.toString()
            var txt_password : String = password.text.toString()

            loginUser(txt_email, txt_password)
        }

        goToRegisterButton.setOnClickListener{ it: View!
            Toast.makeText( context: this, text: "A new soldier will join so", Toast.LENGTH_SHORT).show()
            startActivity(Intent( packageContext: this@LoginActivity,RegisterActivity::class.java))
            finish()
        }
    }
}
```

Figura 3 - Demonstração de parte do código utilizado.

```

class LoginActivity : AppCompatActivity() {
    private lateinit var email :EditText
    private lateinit var password:EditText
    private lateinit var loginButton :Button
    private lateinit var goToRegisterButton: Button

    private lateinit var auth:FirebaseAuth

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        email = findViewById(R.id.login_email_input)
        password = findViewById(R.id.login_password_input)
        loginButton = findViewById(R.id.login_button)
        goToRegisterButton = findViewById(R.id.login_register_button)

        auth = FirebaseAuth.getInstance()

        loginButton.setOnClickListener{ it: View!
            var txt_email : String = email.text.toString()
            var txt_password : String = password.text.toString()

            loginUser(txt_email, txt_password)
        }

        goToRegisterButton.setOnClickListener{ it: View!
            Toast.makeText( context: this, text: "A new soldier will join so", Toast.LENGTH_SHORT).show()
            startActivity(Intent( packageContext: this@LoginActivity, RegisterActivity::class.java))
            finish()
        }
    }
}

```

Figura 4 - Demonstração de parte do código utilizado (cont.)

```

private fun loginUser(txtEmail: String, txtPassword: String) {

    auth.signInWithEmailAndPassword(txtEmail,txtPassword).addOnCompleteListener{ it: Task<AuthResult>
        if(it.isSuccessful){
            Toast.makeText( context: this, text: "Login feito com sucesso!", Toast.LENGTH_SHORT).show()
            val intentToMain = Intent( packageContext: this@LoginActivity, MainActivity::class.java)
            startActivity(intentToMain)
        } else{
            Toast.makeText( context: this@LoginActivity, text: "Falha no login, ou a senha ou o e-mail estão incorretos", Toast.LENGTH_SHORT).show()
        }
    }
}

override fun onStart() {
    super.onStart()

    if(auth.currentUser != null){
        val intentToMain = Intent( packageContext: this@LoginActivity, MainActivity::class.java)
        startActivity(intentToMain)
    }
}
}

```

Figura 5 - Demonstração de parte do código utilizado (cont.)

```

class RegisterActivity : AppCompatActivity() {

    private lateinit var email: EditText
    private lateinit var password: EditText
    private lateinit var registerButton: Button
    private lateinit var firebaseAuth : FirebaseAuth
    private lateinit var returnButton: ImageView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_register)

        email = findViewById(R.id.register_email_input)
        password = findViewById(R.id.register_password_input)
        registerButton = findViewById(R.id.register_button)
        returnButton = findViewById(R.id.register_return_button)

        firebaseAuth = FirebaseAuth.getInstance()

        returnButton.setOnClickListener{ it: View!
            Toast.makeText( context: this, text: "Voltando ao inicio", Toast.LENGTH_SHORT).show()
            startActivity(Intent( packageContext: this@RegisterActivity, LoginActivity::class.java))
            finish()
        }

        registerButton.setOnClickListener{ it: View!
            var txt_email : String = email.text.toString()
            var txt_password : String = password.text.toString()

            if(TextUtils.isEmpty(txt_email) || TextUtils.isEmpty(txt_password)){
                Toast.makeText( context: this, text: "Credenciais vazias", Toast.LENGTH_SHORT).show()
            } else if(txt_password.length < 5){
                Toast.makeText( context: this, text: "Tamanho de palavra-passe insuficiente", Toast.LENGTH_SHORT).show()
            }
            else{
                registerUser(txt_email,txt_password)
            }
        }
    }
}

```

Figura 6 - Demonstração de parte do código utilizado (cont.)

```

private fun registerUser(txtEmail: String, txtPassword: String) {
    firebaseAuth.createUserWithEmailAndPassword(txtEmail,txtPassword).addOnCompleteListener(this){ task->
        if(task.isSuccessful){
            Toast.makeText( context: this, text: "Registro feito com sucesso!", Toast.LENGTH_SHORT).show()
            startActivity(Intent( packageContext: this@RegisterActivity, LoginActivity::class.java))
            finish()
        } else{
            Toast.makeText( context: this, text: "Falha no registro", Toast.LENGTH_SHORT).show()
        }
    }
}

```

Figura 7 - Demonstração de parte do código utilizado (cont.)

Nas figuras acima, poderemos observar o código utilizado na realização deste projeto. A implementação do projeto foi bastante complicada pois muitas coisas relacionadas ao firebase por vezes eram muito difíceis de serem encontradas.

Para além disso, ao longo da implementação do projeto, foi verificado que ele era muito mais complexo do que se imaginava, o que levou a refazer todos os menus a determinada altura e a reconstruir toda a lógica da aplicação.

Além disso, a complexidade geral do projeto era tanta que a maioria das funcionalidades ficou parcialmente implementada, sendo a funcionalidade mais bem implementada, o Rig Builder, entretanto, para este, ficaram faltando a adição de peças ao PC e a visualização das peças que o pc possuía.

A funcionalidade menos implementada foi o part picker, que ficou com maior parte dele sem ser implementado por conta do tamanho muito grande do projeto e da aplicação, neste caso, a parte mais bem implementada do part picker foi a seleção de uma cpu AMD, que neste caso, apesar de não conseguir exibir a lista de cpus a partir da firestore de maneira funcional, esta possui seu código parcialmente implementado.

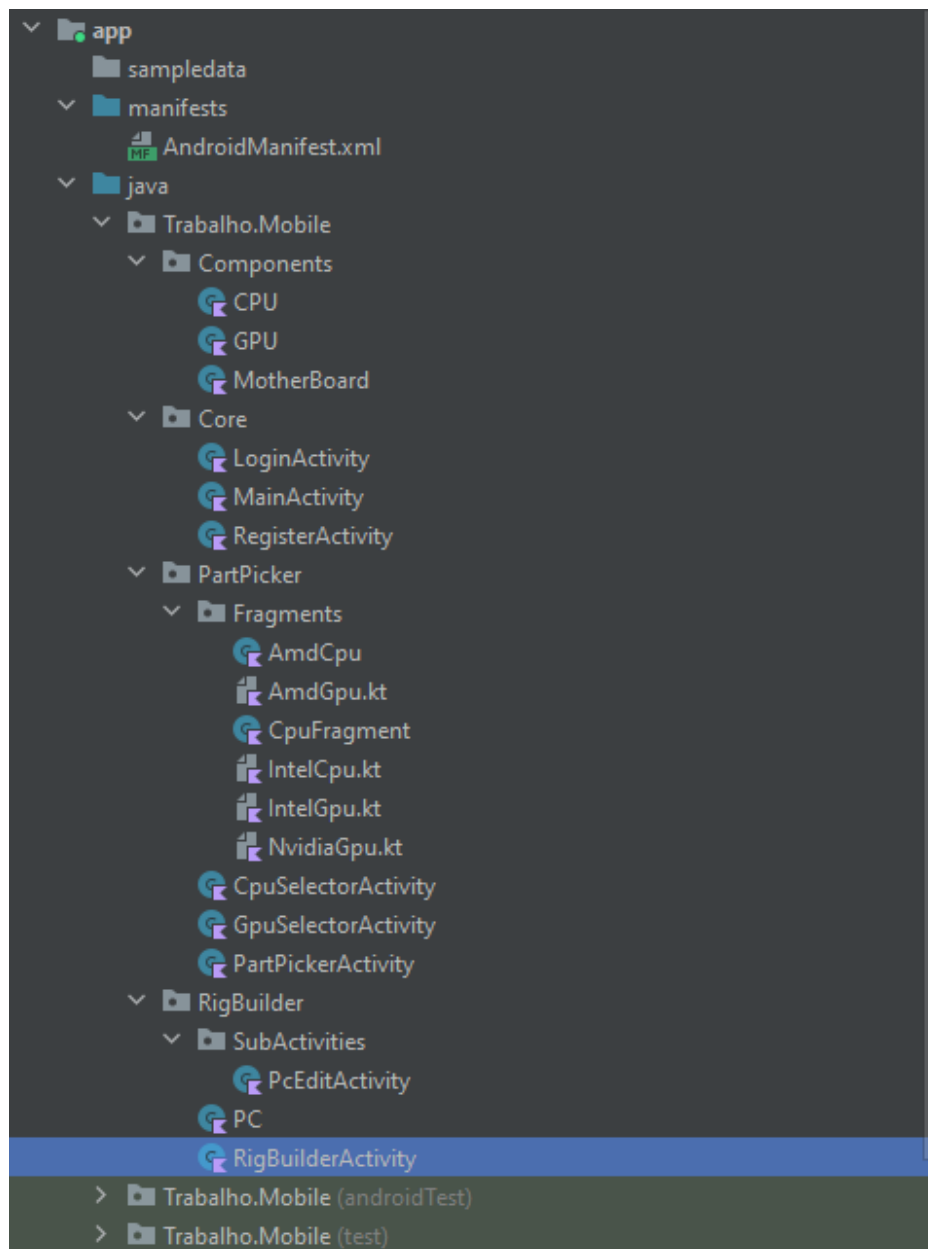


Figura 8 – Esquema do código do projeto ao fim da implementação.

Nota-se na figura acima, a quantidade de código e classes necessárias para esta implementação parcial da aplicação.

Para além disso tudo, foi possível implementar ao menos uma única notificação, que é ativada quando o usuário cria um computador na base de dados.

Tecnologias usadas

Para a realização deste projeto, foi necessário usar:

- Firebase;
- Firestore;
- Firebase Authentication;
- Os nossos PCs;
- Android Studio.

Dificuldades

As principais dificuldades encontradas na realização deste projeto foram:

1. A imensa quantidade de código que foi necessário usar corretamente no desenvolvimento deste projeto;
2. Problemas técnicos relacionados com o próprio Android Studio.
3. O tamanho do projeto em si que posteriormente percebeu-se que era grande demais para fazermos, ao fim a implementação final não foi a desejada, entretanto foi possível implementar uma aplicação funcional

Conclusão

Podemos concluir que, através da realização deste trabalho, obtivemos maior conhecimento ao nível de programação móvel, e da linguagem Kotlin em si.